

EnPRO 350  
Smart Specs  
Spring 2010  
FINAL REPORT

CONFIDENTIAL

## Table of Contents

<u>Value Proposition</u> .....	5
<u>Purpose and Objectives</u> .....	5
<u>Organization and Approach</u> .....	6
<u>Analysis and Findings</u> .....	6
<u>Conclusions and Recommendations</u> .....	7
<u>Summary of Appendices</u> .....	8
<u>Appendix 1: Team Information</u> .....	9
<u>Appendix 2: Expenses</u> .....	10
<u>Appendix 3: Business Plan</u> .....	11
Business Plan Table of Contents.....	12
Executive Summary.....	13
Company Description.....	15
Business Description	
Mission Statement	
Company Goals and Objectives	
Business Philosophy	
Business Model	
Target Market	
Industry Description	
Company Strengths	
Competitive Advantage	
Legal form of ownership: (LLC)	
Products and Services.....	17
Marketing Plan.....	18
Economics	
The Market	
Supply and Demand	
Market Trends	
Potential Growth	
Entry Barriers	
Features and Benefits	
Customers	
Paintball Field Owners and Teams	
Paintball Players	
Competition.....	21
Table 1: Competitive Analysis.....	22
Niche.....	23
Strategy.....	23
Promotion.....	23
Pricing.....	23
Place.....	23
Proposed Location	
Distribution Channels	
Sales Forecast	
Operational Plan.....	24

Production	
Location	
Personnel	
R&D	
Inventory	
Suppliers	
Financial Plan.....	26
Startup Costs	
Return on Investment	
Break Even Analysis	
5 Year Cash Flow	
1 <sup>st</sup> Year Outlook	
5 Year Cash Flow Projection	
Appendices.....	29
Appendix A	
Appendix B	
Appendix C	
Appendix D	
Appendix E	
<a href="#"><u>Appendix 4: Technical Report</u></a> .....	37
Technical Report Table of Contents.....	37
Descriptions.....	38
Arduino Duemilanove.....	38
XBee Pro.....	39
FV-M8 GPS.....	40
Digital Compass.....	41
Spartan-3E 1600E.....	42
Math and Simulation Software .....	43
Problem.....	44
Solutions Roadmap.....	44
Solution.....	44
Files for HUD Math.....	44
Files for Simulator.....	44
Defined Structures.....	46
Functions.....	47
<a href="#"><u>Appendix 5: Prototype Code</u></a> .....	49
Accelerometer Code .....	49
GPU Code.....	52
GPU example	
HUD.....	60
HUD Functions	
HUD Structures	
Input.....	65
Input.h.....	67
Matrix.....	69
Matrix.h	

<b>Matrix Structures</b>	
<b>DirectX Example.....</b>	<b>72</b>
<b>Simulator.....</b>	<b>76</b>
<b>Make File</b>	
<b>Movement Structures</b>	
<b>Open GL Structures</b>	
<b>Project</b>	
<b>GPS.....</b>	<b>89</b>
<b>Wireless.....</b>	<b>91</b>
<b>Accelerometer.....</b>	<b>93</b>
<b><u>Appendix 6: Projected Income Statements</u></b>	
<b>First Year</b>	
<b>Second Year</b>	
<b>Third Year</b>	
<b>Fourth Year</b>	
<b>Fifth Year</b>	
<b><u>Appendix 7: Hardware Data Sheets</u></b>	

CONFIDENTIAL

# Value Proposition

## Our Product

The 3F product provides the ability to spot team members anywhere on the paintball field. This function gives a player an advantage over his or her opponent, which is something players in the paintball market always strive to attain. The 3F product has a lot of potential upgrades for the future. Some of these possible upgrades include indoor usability, UAV simulators, ammunition meter, and many other possibilities. The functionality and durability of our product will make it superior to any competition.

## The Market

The paintball market is distinguished by two parts, scenario paintball and speedball. These two parts can be compared to each other like snowboarding is compared to skiing. The ages of paintball players are usually between 13 and 25; however, they can vary from 10 to 60. Of these players, scenario paintball tends to attract people of the ages 16-25. Scenario paintball players play games that try to mimic military combat. They will also spend the extra dollar not only for vanity, but for any advantage.

Scenario paintball players possess an annual income of \$48,200 on average. A beginner will typically spend \$600 for initial equipment, and usually ends spending over \$1,000 for a full set. A full set includes a paintball gun, ball hopper, mask, pads, extra barrels, and other accessories. Paintball players are always looking for the advantage and our willing to spend money, which makes our product perfect for the market.

## Value

The functionality of our product is a huge part of its value. It provides very useful information to a player, which will help eliminate friendly fire. The advantage given by our product is another key part of its value. The other type of value provided by our product has to do with the market. Our product could lead the way to a new market segment, which would involve all players using the 3F. Many product and field implementations could be added to make this segment more interesting and popular throughout many years to come. Our product not only gives value to the player, but it provides value for the sport of paintball.

## Purpose and Objectives

Our purpose at the beginning of the semester was to develop a system that provided hands-free networked navigation and tracking capabilities and also to assess the market opportunity.

The objectives the team had at the beginning of the semester were also very clear. The objectives were to develop a functional prototype to demonstrate the unique underlying concept being addressed by the IPRO, provide a comprehensive study of the applications for which the technology can be further developed and implemented and perform a preliminary business plan for the technology.

The main problem our project is addressing is fratricide. Friendly fire is something that has to be prevented in the field and our goal is to reduce the probability of fratricide. In order to prevent this we decided we needed a way to identify friendly forces. That is essentially what our product does. It identifies our teammates in the open field allowing us to make the proper choice when it

comes to firing a weapon. We decided that the best way for this to be used was to be mounted on the head, making it hands free. Hand held devices do not work the same because you need your hands for other more important things when you are out on the field.

## **Background**

GPS modules and tracking devices have influenced many advances in modern technology. The data GPS provides has replaced the map and compass, and can be crucial in many situations. Its current platform for use however requires the user to utilize a handheld device and direct his/her frame of view to a screen. In many situations there is a need to have eyes on and full awareness of surroundings, something that simply can't be done while holding and viewing a handheld device. To solve this dilemma, team Smart Specs has integrated ideas of GPS, heads-up displays, and datalink technology to form a design that will give the user full hands free capabilities. This device, worn like glasses, allows the user to track objects and even designate objects to be tracked, which are all shown through heads-up display. Through the lenses you see everything in front of you, but you also see a real-time head-up display of information. The information is tracking the GPS in your teammates Smart Specs and there are small arrows in your field of vision that point to where they are.

The team has decided that our product will serve the Paintball community. We had many other ideas of who could use our product, but after a semester of research, we have decided that the best way to go is Paintball fields for now.

## **Organization and Approach**

Our main problem at the beginning of this semester was to determine what would be our market. We had a wide variety of ideas, but we had to choose the one that we felt would achieve the most interest in our product. After research and talking to people, we decided that the best market would be Paintball fields.

The research was a very difficult process. We had to find a way of determining the demographics for paintball players. This was not an easy task since we couldn't find any source that had studied these numbers in the past. We had to find innovative ideas to get the information that was necessary to give our product a chance to grow.

We had a Technical Team and a Business Team. All of our research and steps taken during the semester are explained in Appendix 3 (Business Plan) and in Appendix 4 (Technical Report).

## **Analysis and Findings**

This semester involved a lot of research regarding paintball players. After analyzing all the information we were able to get, we came to a conclusion on what we wanted to do with our product. At first everyone on the team had the same idea of selling it in the future to individual consumers. We decided to take a different approach based on our findings. We came up with the idea to rent our product to paintball fields. This seemed to be our most solid idea; the one we felt would work the best for us.

We feel this was our most important discovery because now we know the direction we will go in. Our technical team has done a great job of constructing our prototype. The business team has

accomplished an important goal which was to find the most appropriate market for us. We described all of our business ideas thoroughly in Appendix 3.

## **Conclusions and Recommendations**

The prototype that was built matches very closely the ideas that Smart Specs envisions for the final product. We were not able to have a functional prototype by IPRO day, but we were very close to it. We had some problems at the end that prevented us from finishing our prototype. We have also created the first draft of a comprehensive Business Plan.

### Recommendations (Technical)

Design and demonstrate a Command Node and Headset with all the proposed functionality. This is the next step for our project because this is essentially how our product will work out in the paintball fields

Incorporate OLED (display). This does not require a backlight to operate and we would like to have this incorporated in the future.

Include a hands free radio. Make it easier for team members to communicate with each other.

Facilitate the miniaturization of the head set (Marketable Alpha Unit).

Conduct Alpha testing.

Refine the mounting and bracketing harnesses.

### (Business)

Confirm market interest for the paintball opportunity and define a go-to-market strategy.

Identify critical component suppliers and potential channels of distribution.

File for a provisional patent.

Acquire investors.

Prepare a convincing business plan and enter multiple business plan competitions to gather feedback from investment-savvy judges.

Investigate potential partners who can help with scale up and production.

# Summary of Appendices

## Appendix 1

**Team Information:** Appendix 1 lists all the people that were part of IPRO 350 Spring 2010.

## Appendix 2

**Expenses:** Appendix 2 lists all the equipment bought for the 2010 Spring semester. It consists of the prices per unit, with the sum of all expenses for the entire semester.

## Appendix 3

**Business Plan:** Appendix 3 consists of our business plan. All of our business strategies are explained in great detail.

## Appendix 4

**Technical Report:** Appendix 4 consists of all the hardware used in the production of our product type. We have a complete explanation of every single piece of hardware.

## Appendix 5

**Prototype Code:** Appendix 5 has all the code used in the production of our prototype.

## Appendix 6

**Projected Income Statements:** Appendix 6 has the projected income statements that are part of our business plan.

## Appendix 7

**Data Sheets:** Appendix 6 consists of the Data Sheets for all the hardware that makes up our prototype.



# Appendix 1: Team

**IPRO 350  
Spring 2010  
TEAM INFORMATION**

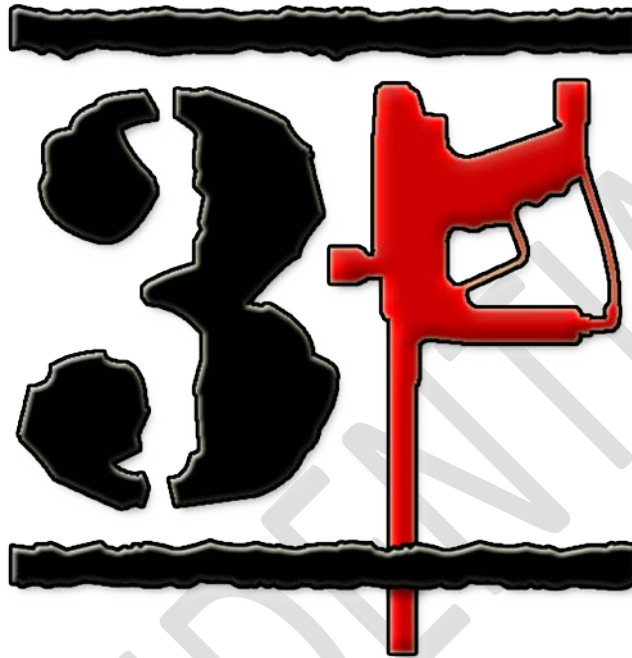
**Instructor:** Jim Braband [braband@iit.edu](mailto:braband@iit.edu)

Teague Algie	<a href="mailto:talgie@iit.edu">talgie@iit.edu</a>	Computer Science
Aaron Clovsky	<a href="mailto:aclovsky@iit.edu">aclovsky@iit.edu</a>	Computer Engineering
Phillip Hoylman	<a href="mailto:phoylman@iit.edu">phoylman@iit.edu</a>	Business
Sarah Hutchins	<a href="mailto:shutchil@iit.edu">shutchil@iit.edu</a>	Electrical/Computer
Konrad Kawa	<a href="mailto:kawakon@iit.edu">kawakon@iit.edu</a>	Electrical/Computer
Matthew Marks	<a href="mailto:mmarks2@iit.edu">mmarks2@iit.edu</a>	Electrical
Jose Miranda	<a href="mailto:jmirand2@iit.edu">jmirand2@iit.edu</a>	Electrical
Purvag Patel	<a href="mailto:ppate29@iit.edu">ppate29@iit.edu</a>	Mechanical
Jason Stogner	<a href="mailto:jstogner@iit.edu">jstogner@iit.edu</a>	INTM

## Appendix 2: Expenses

Component Name	Quantity	Unit Price	Total
Wii Motion Plus	1	19.99	19.99
SANAV FV-M8 (EB-85A) 5Hz GPS Engine Module	1	81.95	81.95
GPS Shield	2	16.95	33.90
8M SRAM	2	17.22	34.44
Breadboard extension for FPGA development board	1	29.99	29.99
EB-85A header for GPS shield	1	1.95	1.95
8-pin riser for Arduino	4	0.50	2.00
6-pin riser for Arduino	4	0.50	2.00
Arduino Duemilanove	1	29.00	29.00
MODULE 802.15.4 CHIP ANT - XBP24-ACI-001	3	32.00	96.00
Arduino XBee Shield Empty	1	24.00	24.00
9V Snap Connector	2	1.24	2.48
ANDpSi020TD-LED-KIT	1	125.00	125.00
RS-232 transceiver	1	4.19	4.19
1uF capacitor	5	1.40	7.00
200 ohm resistor	3	0.23	0.69
255 ohm resistor	2	0.16	0.32
523 ohm resistor	2	0.16	0.32
511 ohm resistor	4	0.16	0.64
1k ohm resistor	4	0.29	1.16
2k ohm resistor	4	0.16	0.64
3k ohm resistor	4	0.08	0.32
M3 x 14mm screws (box of 100)	1	5.51	5.51
5V line	1	11.37	11.37
12 rail with unwanted features	1	16.19	16.19
11.1V – 14Wh killer battery	1	29.99	29.99
Board	1	7.33	7.33
Polycarbonate Sheet 1/8" Thick, 12" X 12", Clear	1	6.59	6.59
Polycarbonate Sheet 1/16" Thick, 12" X 12", Clear	1	3.84	3.84
6mm Hex, 52mm Length, M3 Screw	8	3.40	27.20
Metric Class 4.6 Plain Steel Threaded Rod M3 Size	1	1.39	1.39
Metric Zinc-Pltd Threaded Hex 6mm Hex, 10mm	13	1.20	15.60
Architectural Anodized Al. 1/16" Thk, 3/8" X 3/8"	1	6.47	6.47
<b>TOTAL</b>			<b>\$ 629.46</b>

## Appendix 3: Business Plan



### **FRIENDLY FORCES FINDER**

*“See Your Team”*

#### **3F – Friendly Forces Finder**

3424 S State St

4<sup>th</sup> Floor, Room 4C-1

Chicago, IL 60616

815-325-2435

phoylman@iit.edu

## Table of Contents

I.	<a href="#">Table of Contents</a> .....	12
II.	<a href="#">Executive Summary</a> .....	13
III.	<a href="#">General Company Description</a> .....	15
IV.	<a href="#">Products and Services</a> .....	17
V.	<a href="#">Marketing Plan</a> .....	18
VI.	<a href="#">Operational Plan</a> .....	24
VII.	<a href="#">Financial Plan</a> .....	26
VIII.	<a href="#">Appendices</a> .....	29

CONFIDENTIAL

## **Executive Summary**

The necessity to locate entities that are not in line-of-sight is a detrimental paintball need. The current technology that exists, which facilitates the tracking of teammates while enhancing the layman's operational capability, is currently the governor of a professional's operational tempo. The aspect of current technologies which limit operational capability is that these devices such as GPS navigation systems, walky-talkies, maps and compasses are hand-held. Meaning they tie-up their users' hands while they fidget with such devices. To caveat off of that aspect, such devices also take your eyes and focus off of the over-all objective, thereby, increasing the risk of mission failure.

The purpose of Smart Specs LLC is to provide a hands-free device that can facilitate teammate tracking, data sharing and inter-team communication. The 3F project enables real-time location data to be displayed in the user's regular field of view, as well as enhance communication amongst the team through a built in hands-free radio. Smart Specs wirelessly provides viable information to multiple users who are on the same net, thus eliminating the need for other methods of communication.

Our strategy is to sell the 3F product to paintball fields across the country and to create a new paintball game scenario. We will start off by traveling across the country, starting in the Midwest, heading southeast, and finally making our way clockwise to the remainder of the country. This plan has been constructed for us to hit the hottest paintball spots first. The van we will be traveling in will be decked out with Smart Specs logos, and we will hit up as many paintball stores along the way in order to spread the word of our product.

The target market is paintball, specifically the scenario based paintball market. Players in this segment tend to be ages 16-30, and also have flexible incomes. These players will spend top dollar for vanity and any advantage. These players will typically spend \$1,020 on gear very quickly. Scenario paintball players have also developed many games, which include capture the flag, siege the base, hostage rescue, and many more. Our product will introduce them to a new scenario. At first they will be able to see each other, but the future of our product will be very broad. Our technology will be able to simulate UAVs (spot enemies for a period of time), object locator (see the flag), indoor blueprints, ammunition meter, and many other possibilities. The current and future functionality of our product will create a new, fun scenario that will make players feel like they are living in a video game.

Paintball fields are the primary target for our product. There are roughly 1,500 fields in the U.S., with the majority located in the Midwest and Southeast. We plan on visiting many of the top fields in order to get our game scenario started. 50 units will allow for a 25 man team scenario for paintball fields. 50 units will cost roughly \$37,500; however, by renting each unit for \$15 twice per day and 8 days per month, the entire \$37,500 will be recovered in 3 months.

Our operations will be on the road for the first 4 months, with our company being located in a family garage. Six months into the startup we will be moving into a 3,000 square foot facility, which will hold our office, R&D lab, and warehouse. At the start, we will have 100 units as safety stock. We will be distributing out of our warehouse, handling business and orders in the office, and doing R&D for future technological and product upgrades in the lab. We will have 3

employees, which will consist of two engineers and one business savvy leader. The first two years will net each employee \$24,000 annually in order to push our company into the correct direction.

The startup costs are roughly \$300,000. A little over \$60,000 of it will be needed 6 months down the line for our facility. The remainder of the investment will be needed immediately. \$67,500 will be need for initial product, which will consist of a starter 50 unit batch plus 100 units of safety stock. We will need \$5,000 for initial R&D, \$33,000 for marketing, \$35,000 for initial capital, \$36,000 for salaries, and an extra 20% for contingencies.

Our company provides a unique technology into an existing market, but we are creating a new segment. We are categorized as high risk, but we will have a return-on-investment of 40%. In other words, we will return 10 times our investment in 5 years. The breakeven has been calculated to be roughly 1 year and 3 months. This is the amount of time it will take for us to recover the first year losses from startup and fixed costs, and the variable costs of the first 15 months. Our 5 year estimates suggest our company to have made over a million dollars in revenue by year 3.

## **General Company Description**

### **Business Description:**

Smart Specs stands alone as the paintball industry's sole heads-up-display (HUD) provider. Smart Specs target customer is made up of paintball field owners and teams. We provide these clients with state of the art electronics designed to give them the edge over their opponents. Our flagship product 3F provides its user with real time hands-free teammate tracking and data sharing. 3F will revolutionize the way in which paintball is played and we plan to take full advantage of this revolution for years to come.

### **Mission Statement:**

Our mission is to provide our paintball clients with state of the art, useful technology, designed to give its user the advantage over their opponent in combat scenarios.

### **Company Goals and Objectives:**

Our goals include durability, reliable, and any other traits our products need in order to maintain good customer satisfaction. Our product augments modern paintball tactics and techniques giving our user an extreme advantage. In order to maintain that advantage we will release new accessories, software upgrades as well as design multiple scenarios in order to keep customer interest and facilitate our customers need. Our initial financial goals are to distribute our product to 50+ paintball fields within the United States, and continue to spread throughout the nation. We will strive to develop more than one optical device to ensure our customer base maintains growth in size and satisfaction.

### **Business Philosophy:**

"If we can't find a way, we'll make a way." Innovative and effective solutions that facilitate the needs of our clients stand atop all other priorities. Our business philosophy is rooted in the advancement of entertainment and customer satisfaction. We also recognize the fallible nature of equipment in extreme conditions which is why if any customer that is unhappy with our product will have tech support available as well as a mobile tech team that will visit our clients annually or upon request. We will also provide our customers with the ability to send in damaged units for repair or replacement. Our company aims to be a well-known and respected business in the paintball community, and perhaps other markets in the future.

### **Business Model:**

Our business model is to sell our product in bundles to paintball fields located around the nation. We will start with the most popular and weather friendly regions and work our way around the country. We will have a company van with our logo all over it, which will allow us to visit the fields we intend to do business. Our product will be attached to a thermal mask, and will have any necessary pieces of equipment for upgrades, repair, or use along with it. Product promotion will take place as we visit individual fields and stores during our trip around the nation.

### **Target Market:**

Our target market lies within the paintball industry, specifically scenario based paintball (also referred to as woods ball.) Scenario based paintball has a slightly higher population than speedball, which is the other paintball market. Woods ball recreates real-world battlefield

conditions as closely as possible. Scenario based teams usually have ranks assigned to each team member, they execute practiced maneuvers, they engage and defend their enemy and territory, all of which mirror military operations. Our product would provide a perfect fit within woods ball as it provides each user with real time data transfer, the ability to see their own teammates whether or not they happen to be within or out of line of sight. Woods ball players have expendable income. They are players who will spend top dollar to gain the advantage over their opponents. The current capability and future upgrades to our product fit perfectly within the woods ball market.

Fratricide regularly occurs in paintball and our product 3F addresses and dramatically reduces that occurrence. Our current product coupled with future upgrades enables us to create an entirely new market segment in the paintball industry where people would go to fields specifically for the use of 3F.

### **Industry Description:**

The paintball industry has been around for many years. In fact, the industry has grown 80% in the past 4 years. The potential for continued growth in the paintball industry is high, and with the introduction of 3F, we believe the paintball industry will experience a greater rate of growth. Our company can take advantage of the recent and potential growth by establishing our product as a trusted addition to the paintball community. With the introduction of new gaming scenarios, software upgrades and game enhancing accessories, our system will gain popularity for years to come.

### **Company Strengths:**

Our company's greatest strength is our Smart Specs team. The Smart Specs team is comprised of a breath of engineering, business, and industrial technology backgrounds assimilated into one team with the same philosophy: "if we can't find a way, we'll make a way." Our product is unique, innovative and will prove to be a dependable augmentation for any paintball team. Our product offers outstanding interchangeability between standard paintball masks, which prevents further cost incursion. Once we have established our product within the paintball industry, customers will know that the implementation of our product means success in the field.

### **Competitive Advantage:**

Without any direct competition currently in the paintball arena, our indirect competitors are hand-held devices such as radios, GPS, and the utilization of maps and compasses. Our initial prototype (to be beta tested) eliminates the need for hand-held radios, GPS devices, and extremely slow map and compass techniques. Our prototype has the capability to facilitate all three of those devices and best of all, it will do so in a hands-free manner enabling your hands to stay on your marker (paintball gun), and your head in the game. Our product will create never before possible game scenarios, as well as bring realistic combat communication to the paintball field. The ability to incorporate teammate identification, inter-team communication, and data sharing all in real time will make it tough for these indirect competitors to compete on the paintball field against teams using our product.



### **Legal form of ownership: Limited liability corporation (LLC)**

A LLC puts the liability on the company, rather than the individual employees. As a LLC, we can only have 75 employees and 100 investors, but there are benefits. Our company does not need more than 75 employees, and we would like to keep the liability on the company instead of on individuals. Single taxation is a key benefit, which means the company gets taxed; however, employee salaries do not get taxed.

### **Products and Services**

3F is a unique product intended for the paintball industry with the potential for many upgrades and uses. Our product securely attaches to the standard paralleled slot vents atop the paintball mask. 3F's minimal weight, low profile and impact resistant resin makes it an ideal piece of equipment for a strenuous sport like paintball. The 3F system locates team members throughout the paintball field by showing a red dot on the screen along with a distance. (See **Appendix A**) Our product integrates GPS with WiFi to provide real-time mapping of teammates which is displayed through a micro thin LCD screen. The HUD's create a net through a command node which is maintained by the team commander. This net referred to hence fourth as Battle Net facilitates the transfer of data packets between the HUD's.

Our product is superior to alternatives, because of its functionality, durability and hands-free nature. Our technology will combat fratricide, enable real-time data sharing, and enhance the operational capability of any team equipped with our product. Our product will be the first to create a game scenario that simulates an unmanned aerial vehicle video feed (Allows a player to see the other team) similar to that of modern day warfare and battle simulation video games.

Future additions to our product could include:

1. Wrist-top texting user interface device
2. Voice activated radio system
3. Ammunition meter
4. Full field UAV simulator
5. Overhead multi-map views
6. Software upgrades for optional color displays
7. Clear OLED displays
8. Protective mask inner lens displays
9. Weapon mounted range finder for spotting enemy locations
10. Indoor usability

# Marketing Plan

## Economics

### **The Market:**

The paintball industry currently consists of roughly 10 million players. The average player is between the ages of 13 and 25; however, some players will start as young as 10 and play as long as 60. Paintball is the number 1 growing sport in America and about \$720 million dollars was spent on paintball related products and services in 2008.

### **Supply and Demand:**

Fratricide and maneuverability are major concerns in the paintball market, and gaining the advantage is a primary reason why players will spend the extra dollar for equipment. The remaining percentage is motivated by vanity. Our product not only gives a team a tactical advantage, but it also provides a psychological advantage of making its user look more combat effective. Therefore, the demand for our product will be high enabling our ability to cover our initial investment by way of supplying that demand.

### **Market Trends:**

In the past 4 years, paintball has experienced an 80% growth. A major reason for this is the current economic crisis. An effect of the crisis results in people not spending money on vacations but, making up for the lost vacation time with local entertainment. Paintball proves to be a less expensive way to step out of everyday life with the added benefit of being much closer to home. Our product will enter a growing market and will prove to expand the market further also assisting Smart Specs in recouping our investment payoff very quickly. Once we have established our product in the industry and create our own game scenario, our product will continue to be successful through future inflations or recessions.

### **Potential Growth:**

Our product's unique functionality and the current growth and size of the paintball industry allows for a lot of potential growth for our company. Added features and technological upgrades will allow our company to be successful with our product at many paintball fields world-wide. At first, we plan on demonstrating through leasing our product to the top 10 fields located in what we have divided into five major regions of the United States. We plan to seed the market for what we have evaluated to be a viral demand.

### **Entry Barriers:**

There are several barriers to entry to consider. The first factor is dependability. The entire premise of success or failure teeters on our ability to increase our end user's operational capability. Without a dependable device the market will lose faith in our ability to provide a worthwhile product. Our second hurdle will be start-up cost. A project of this sophistication which integrates so many high-end technologies will have high up-front costs. We have estimated our initial start-up investment at \$300,000 to cover final R&D, manufacturing consulting, patent costs, marketing, salaries, start-up equipment, and other incidental expenses. We will experience weighty production costs initially because we will only be producing in batch which means a higher per unit manufacturing cost. However, our unit costs will fall as

production increases. Another major entry barrier is consumer acceptance and brand recognition. Our product will rely on people giving our equipment a try and spreading the word. So long as people are willing to try the 3F system, they will recognize the increased operational capability our product provides and will spread the word to other consumers. Another barrier to entry is the way in which we have assimilated our technology and the way in which it functions. The 3F system is a unique product; therefore, a patent will be needed to make sure our company has some intellectual property protection.

Overcoming our barriers can be done in several different ways. Our product will be beta tested in extreme conditions, and necessary measures will be taken to overcome any shortcomings we may find through testing. This will lead us towards a more dependable product helping us mitigate Battle Net and product failures. Founders and key employees will defer much of their salary to keep initial costs down and appeal to investors. Overcoming consumer acceptance should be one of our primary tasks as that is what our success hinges upon. To ensure acceptance we will provide a dependable product that will undoubtedly improve our users' operational readiness and enhance their capabilities. We heavily depend on our products reliability and unique ability to increase our users' effectiveness to foster word of mouth promotion. We have also decided to take a look into purchasing tradeshow booths during our initial market run to bolster our products identity and boost sales. A portion of our start-up capital will be spent on filing a patent.

### **Features and Benefits:**

Our product's main feature resides in its hands-free ability to identify teammates. 3F provides the ability through the integration of a micro thin LCD screen attached to the front of the mask over the user's non-dominant firing eye. The LCD has the appearance of a thin sheet of glass; however, red dots identifying the positions of teammates as well as an approximate distance to the other users appear on the screen. This feature helps eliminate fratricide on the paintball field, which has been identified as a very common error. Currently, most paintball field owners use an armband to determine who is on what team, and if members of the opposite team are wearing similar clothing and the identifying cloth is tied on the arm which is not in your line of sight you will have a distinct advantage over the opposite team. The feature also allows players to know how many people are left on their team. When someone is shot they can turn off their 3F and eliminate their red dot on other people's 3F systems.

The benefit of our product is the advantage it gives a user. Instead of having to inspect a person on a field to see if he or she is an enemy, a user can act on what we call "red dot reflex." Our product also gives the benefit of allowing people to feel as though they are in a video game and provides an experience into new game scenarios they never would have fathomed to be possible.

Our company will give after-sale services that include repairs, warranties, and refund-policies. We will only producing in batch to start, be we want to make sure they work properly in order to expand our game's popularity. Therefore, we will service any of our products that do not work properly, because we will only be distributing a small amount at the start. Another reason we'll initially produce in batch is because we want to get end user feedback to make any final adjustments before we launch the product for full scale manufacturing this is to avoid the release of a faulty product. A batch will consist of fifty 3F systems. We will also provide a warranty,

because every item in the paintball industry must be durable and last through many battles, as a beta product, we foresee a possible need for return. A refund policy will also be given. If any product does not satisfy a user or field, we will refund them and re-distribute our product. Since we will have a limited supply of demonstration units, we want to make sure our products are at fields that accommodate lots of players, which will increase our brand visibility in the paintball community.

## Customers

### **Paintball Field Owners & Teams:**

To best establish market research for these consumers one must analyze the people who comprise their customer base, the players themselves.

### **Paintball Players:**

Paintball players are usually between the ages of 13-25, with a minimum age of 10 and some players have been seen at the age of 60. Paintball players in scenario based games tend to be in the ages of 15-25; however, most woods ball players are in their 20s and spend a good amount of money on equipment. Scenario players will spend cash on equipment that gives them the greatest advantage, and they will spend money on vanity.

The first and most important piece of equipment is the paintball gun, also called a marker. The average marker goes for around \$500; however, they can range from \$150-\$1500. Many players will buy a circuit board for their marker, which allows them to switch between firing modes. These boards go for an average of \$100. Electronic hoppers will also be bought more often than not, because of ball breakage. The electronic hopper feeds the marker paintballs at a constant rate, which prevents paintballs from jamming and breaking in the gun. These hoppers range from \$65-\$150, with an average of \$65. Non-electronic hoppers go for \$20, which explains why the average hopper is at the low end of the spectrum. Masks, which are essential for our product, go for an average of \$40, with a range of \$20-\$100. The reason for the expense is the anti-fogging lenses in the masks. Fogging is a big problem, because of the amount of exhaling a player will do in battle. Barrels are the first piece of equipment many paintball players upgrade. Barrels go anywhere from \$35-\$200, most players end up spending \$200 on multiple barrels. Players will also spend \$100-\$300 on other miscellaneous items such as stocks, remotes, and pads. The average player usually starts with around \$600 worth of equipment, but can end up spending \$1,000 after a few times playing. Paintballs are also a major expense in the market. Most tournament teams go through 10-15 cases of paint a day, which is roughly \$5000-\$7500. Paintball players spend a lot of money, whether it is on equipment or for paint. (See **Appendix B** for a player's cost image)

Scenario based paintball has recently undergone an increase in players while speedball (The other market of paintball) has seen a drop in participants. We attribute this to a multitude of factors, since we are a nation at war we find more participants want to replicate real-world combat in a safe manner, we also recognize that our country is currently in an economic recession and the fact that people want to maximize the spending power of their dollar means that paintball players will spend more money on longer scenario based paintball bouts than speedball is due to the short duration of the games, cost of field rental, and cost of expensive

markers, without such expensive equipment it puts the paintball player at an egregious disadvantage. The variety offered through woods ball supersedes that of speedball making enabling the game to differ exponentially. Paintball equipment and game scenarios are mainly heard of via the internet or through word of mouth. Scenario playing fields have maintained popularity through recent years, and offer a great place for the 3F system to enter the industry.

Paintball fields are located all around America in what we have divided into five different regions. The regions are the Southeast, Midwest, Northeast, Northwest, and Southwest. They are also listed in order from the most fields to the least fields per region. (Southeast leading with 417, and Southwest with the least at 182) See **Appendix C** for a list of states for each region, top 10 fields in each region, and the top 10 states.

## Competition

Our product has no current direct competition, because of how unique and new the 3F technology is to the market. However, we do have indirect competitors such as radios or GPS devices. Many paintball teams use radios, which are typically \$40 per unit. Our product offers greater versatility and functionality when compared to our closest competitors. The problem created through the utilization of existing technologies is that they require the use of at least one hand and take your eyes off the battlefield to utilize the device rendering its user combat ineffective. Our product is superior to those other means of communication and positioning systems because 3F offers its user the full use of his other equipment as well as the ability to keep their eyes on the paintball field. The hands-free approach will give us an advantage over the indirect competition by way of functionality. A table comparing our product with other forms of indirect competition can be found in **Appendix D**. The following table rates our product versus radio and GPS on a scale of 1-3, 1 being the best and 3 being the worst. For the Importance to Customer column, 1 is very important and 3 not as important.

**Table 1: Competitive Analysis**

<b>FACTOR</b>	<b>3F Product</b>	<b>Strength</b>	<b>Weakness</b>	<b>Radio</b>	<b>GPS</b>	<b>Importance to Customer</b>
<b>Products</b>	1	Use of Hands	-	2	3	1
<b>Price</b>	3	Advanced Technology	Need Multiple Units	1	2	3
<b>Quality</b>	1	Advanced Technology	-	3	2	1
<b>Service</b>	1	Repairs and Warranty	-	3	2	1
<b>Reliability</b>	1	Good Battery Life	-	2	3	1
<b>Ease of use</b>	2	Easy to Learn	-	1	3	2
<b>Location</b>	1	At Fields	-	2	2	3
<b>Appearance</b>	1	Small and Appealing	-	2	3	2
<b>Advertising</b>	1	Fields and Online	None in Retail	2	2	2
<b>Image</b>	1	New Game Scenario	New to Market	2	2	2

## **Niche**

Our niche is the new game possibilities our product brings to the paintball market. Paintball fields that purchase our product can create new gaming scenarios, augment their sponsored teams, and facilitate a revolution in the paintball industry. Future additions to the functionality of the 3F will keep our consumer hooked to the product. Players will use our product to turn their average paintball games into real- life combat scenarios, as well as brings video games to life.

## **Strategy**

Our strategy is to develop fifty 3F units and take these fifty units around the country. We will start out in the Southeast Region where the weather permits the playing of paintball year round. We will hit the 20 most popular paintball fields over a two month period starting in January and ending in early March. We will lease the product for the day to the paintball field owners and collect our product up at the end of the day. As we leave each field we will take customer orders and facilitate the channeling of those orders to our manufacturer who will produce and send them to our company headquarters who will distribute the systems accordingly. In March the mobile demonstration team will hit the Southwest region traveling to the ten most popular paintball fields and will again lease the systems to field owners, collect orders, send them to the manufacturer and once produced distribute them to the fields. We will continue to hit a region a month until we have lightly saturated all five regions. We will have a website of which orders can be placed and a mobile demonstration team who thereafter can be requested to deliver the demonstration bundle for field owner evaluation. We intend on this method of initial entry to spread information about our product via word of mouth facilitating the advertisement and sale of enough 3F systems to cover our initial start-up loan in a reasonable amount of time. We intend on reaching the international market prior to our investment payoff benchmark. The Southeast region is the most paintball dense region therefore by releasing initially in the Southeast during a time where few other fields are open around the country we can maximize our brands visibility prior to the start of the nationwide paintball season.

## **Promotion**

We plan on promoting our product using several different methods. We will use a van with our logo on it to go around the country in order to sell our product to fields. We will also stop at paintball stores and hand out banners, posters, and brochures describing our product's functionality. We will stop at major paintball events such as tournaments or trade shows. Trade shows and tournaments will introduce us to bloggers, which will help spread our brand name on the internet. We will start in the Midwest and head to the Southeast, because those are the most popular paintball areas in America. We can also hold sponsored events at perspective paintball fields, which would feature our product and introduce it into the market.

## **Pricing**

Due to the sensitive nature of contractual agreements between Smart Specs LLC and our manufacturing counterparts we cannot disclose this information at this time. However, we will be pricing our product based off the total costs per unit and the value our product gives its user.

## **Place**

### **Proposed Location:**

We will have a distribution center located in Chicago, because that is where our founders currently reside. We will handle our orders online via our website and by phone. Fields and players will have our website's URL and distribution center's phone number readily available from banners, brochures, posters, and other forms of promotion we hand out. Our van will also have units of our product available. We will be selling our product as we travel across the country.

### **Distribution Channels:**

Our primary distribution channels will be online and by phone. We will take orders online and by phone, and send our units from our distribution center, or deliver them ourselves. We will be traveling across the country in our company van, which is a method of distribution and promotion.

### **Sales Forecast:**

After 4 months of initial promotion and sales, we expect to lease out 50 units to 32 different fields. That is 8 fields per month and each unit will be rented at \$25 apiece. After our entry we will sell 50 units a month the first few months, and then through popularity and progression our sales should increase by 100% very quickly. At the very minimum we will be able to cover our debts and startup expenses; however, we expect to have large growth during the first year or so.

## **Operational Plan**

### **Production**

Our intent is to initially produce 50 units, in batch, and do so locally within the Chicago land Region. We will do this for two reasons: it will enable us to have more control during the manufacturing process as well as to reduce logistic cost. However, we do intend to seek out a large scale manufacturer to better facilitate the production of larger quantities. We are not looking internationally for this as we will try to protect our product from being cloned as for as long as possible. The manufacturer we are speaking with as estimated the production costs of our 50 units to total \$450 per unit. This is an exceptional figure compared to our first quote of \$750 per unit.

### **Location**

Initially, we plan to run our company out of a family member's garage, however, we intend to move our company into building that can facilitate as a base of operations. We will hit that point around the six month mark. Regardless of the building we finally settle in we will have to make additional security enhancements to better facilitate our security needs. The facility that we choose to run our operations out of will have to be large enough to facilitate our office, research and development, maintenance, as well as warehouse operations. Some basic features of our facility will have to include office space for processing orders as well as personnel management and company conferences. Our R&D/Maintenance area must be large enough to encompass a work bench, multiple desktop computers, electronics testing equipment, storage for common



parts as well as all tools we will need to conduct maintenance on our equipment as well as to add any additional features we may need. (See **Appendix E** for a detailed picture of the facility)

Our warehouse must have the capacity to store 500 units of our product, have a shipping and receiving dock, environmental controls to protect our product from humidity and excessive heat climate control, and a packaging station so we can ship larger orders as well a shipping processing station to print labels and track shipments. As far as utilities, we will primarily need power and water. One great feature about our product it is small enough to utilize UPS, FED-EX, and DHL. Due to box size and weight we do not feel the need at this time to use heavy cargo third-party shippers.

### **Personnel**

Smart Specs will start off with three full time employees for the first five years. We will each make \$24,000 per year, which is enough to live on for the first 2 years. During the first four months, all three of us will be going on the road show trip in order to sell and promote our product. We can also take orders from the road, which eliminates the need for any of us to be home. After 6 months, we will be renting a warehouse and all three of us will be full time. Our three employees will be handling operations, finance, marketing, and anything else the company needs. We will also handle R&D for our product, which will be feasible with a couple of our employees being engineers.

### **R&D**

R&D will be handled in the facility next to the office. Our engineers will be creating prototypes of our product with new features. For instance, we plan on implementing a grenade like technology for our product, which will allow the user to turn off opposing teams' 3F products for a short period of time. Our team will use the facility and the money budgeted for R&D to make new technology such as this available for our product.

### **Inventory**

As for parts and units on hand, we intend on servicing all equipment in house. This means we will have to have casings, processor boards, OLED displays, and cables on hand at all times. We expect to maintain a ratio of 5% of replacement parts to parts sold.

For the first six months of business we will not be storing any compete units on hand. Rather we will have them shipped directly from the factory to the consumer, however, after the 6 month period and in order to keep up with demand, we expect to move into our permanent base of operations. We plan on storing an initial 100 units as safety stock.

Once our business picks up around the 6 month mark we will move into our new facility which will enable us to stock our products in house. We foresee a 60-90 day lead-time therefore to keep our customers happy we will need to build up a small surplus to satiate initial demand. We expect the initial demand to be no greater than 200 units for the first six months. However, after the first year between the months of October and March we will have to triple our inventory to prep for high sales between April and September.

## **Suppliers**

We have one main supplier who we purchase the parts for our 3F unit from. Newegg a computer parts supplier based in City of Industry, California. Their website is [www.newegg.com](http://www.newegg.com). Newegg has been supplying computer parts for just under a decade and currently service 13,000,000 individuals and businesses. Newegg generally has 5-7 business days lead-time for part delivery and their customer service is unparalleled with 24 hours of availability.

## **Financial Plan**

### **Startup Costs**

The initial investment we need to cover our startup expenses and capitalization is \$300,000. This will be enough to push us into the market, until we start making revenue of our own. We need all but \$60,000 of the \$300,000 immediately, but we have everything we plan to purchase figured out. The \$60,000 we do not need right away is for the facility we plan on renting 6 months into the startup. Therefore, we will need all \$300,000 before 6 months.

Our initial expenses include the initial product development costs of roughly \$67,500, \$135,000 in capital purchases, \$33,000 for marketing, \$5,000 for R&D, and 20% for contingencies which would put us at around \$300,000 for startup. The startup costs are shown after the projected income statements for the first 5 years at the end of the financial analysis.

### **Return on Investment:**

Our product is a high risk investment, but there is an existing market that can generate a good ROI. We are making a new product using relatively understood technology, and we plan on creating our own little market segment in the paintball market. We estimate at minimum a 40% return-on-investment for our product, which will return investors 10 times their money back in 5-7 years.

Our exit strategy would be to minimize our losses and sell off capital equipment in order to pay off our debt. However, our company would be a strategic purchase for popular paintball equipment companies like Tippmann or Angel. Tippmann is huge in the scenario paintball segment, which would make it easier for them to create a new segment featuring our product's functionality. Angel is more popular in Speedball, but they have the reputation and funds to create a new market segment with our product.

For the paintball fields we have estimated a 3 month period to cover the costs of our product. 50 units of our product will cost \$37,500. If fields rent the units for \$15 apiece and rent each unit twice per day, they will be making \$1,500 per day. Since paintball fields tend to do business on Saturdays and Sundays it will take 3 months to completely cover the costs of our product. (8 days \* 3 months \* \$1,500 = \$36,000)

### **Break Even Analysis**

Our break-even point has been calculated to be 1 year and 3 months after startup. The break-even has been calculated using our projected 5 year cash flow and 5 year income statements, which follow this section. Our 1<sup>st</sup> year income statement projects us to be \$95,035 in the negative. Our contribution margin per 50 units is \$15,000. At the start of year 2, we will be \$125,035 in the

negative due to marketing and R&D costs. However, after 3 months in the 2<sup>nd</sup> year, we will have sold 12 batches of 50 units which would completely cover the remaining \$125,035 of debt, plus the \$6,000 for rent and \$29,550 for G&A for the first three months of year two. Therefore, after 1 year and 3 months, each 50 unit batch of our product will start giving us a profit.

### 5 Year Cash Flow Projection / 5 Year Income Statement

The following consists of the 1<sup>st</sup> year outlook, 5 year cash flow projection, followed by the projected income statements for the first 5 years and details of the startup costs.

#### 1<sup>st</sup> Year Outlook

Quarters	1st	2nd	3rd	4th	Total
Sales	\$37,500	\$112,500	\$225,000	\$225,000	\$600,000
CGS	\$22,500	(\$67,500)	\$135,000	\$135,000	\$225,000
Gross Margin	\$15,000	\$45,000	\$90,000	\$90,000	\$240,000
Expenses:					
G&A	(\$98,600)	(\$15,850)	(\$35,550)	(\$35,550)	(\$185,550)
Marketing	(\$36,040)	(\$36,040)	\$0	\$0	(\$72,080)
R&D	(\$77,405)	\$0	\$0	\$0	(\$77,405)
Total Expenses	(\$212,045)	(\$51,890)	(\$35,550)	(\$35,550)	(\$335,035)
Net Income	(\$197,045)	(\$6,890)	\$54,450	\$54,450	(\$95,035)

#### 5 Year Cash Flow Projection

Year	1	2	3	4	5	Total
	Smart Specs Income Statement As of February 1, 2013	Smart Specs Income Statement As of February 1, 2014	Smart Specs Income Statement As of February 1, 2015	Smart Specs Income Statement As of February 1, 2016	Smart Specs Income Statement As of February 1, 2017	
Sales	\$600,000	\$1,800,000	\$1,987,500	\$2,287,500	\$2,400,000	\$9,075,000
CGS	(\$360,000)	(\$1,080,000)	(\$1,192,500)	(\$1,372,500)	(\$1,440,000)	(\$5,445,000)
Gross Margin	\$240,000	\$720,000	\$795,000	\$915,000	\$960,000	\$3,630,000
G&A	(\$185,550)	(\$142,200)	(\$214,200)	(\$286,200)	(\$358,200)	(\$1,186,350)
Marketing	(\$72,080)	(\$25,000)	(\$25,000)	(\$25,000)	(\$25,000)	(\$172,080)
R&D	(\$77,405)	(\$20,000)	(\$20,000)	(\$20,000)	(\$20,000)	(\$157,405)
Net Income	(\$95,035)	\$532,800	\$535,800	\$583,800	\$556,800	\$2,114,165

**Projected Income Statements**

**(Please See Appendix 6)**

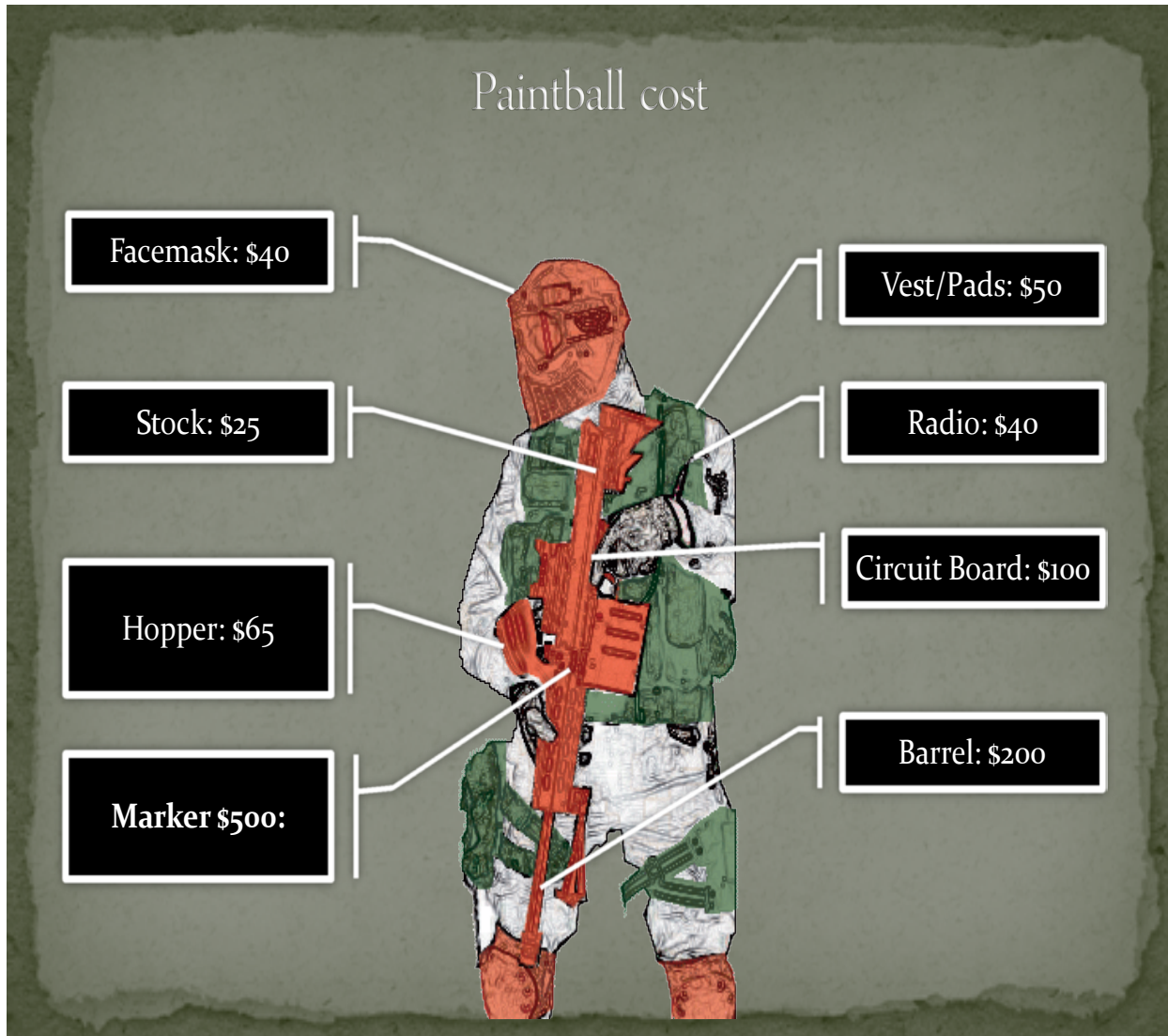
CONFIDENTIAL

## Appendices

### Appendix A:



**Appendix B:**



## **Appendix C:**

### **Paintball Areas:**

#### **Area 1:**

**States:** Arkansas, Louisiana, Mississippi, Alabama, Georgia, Florida, Tennessee, Kentucky, West Virginia, Virginia, North Carolina, South Carolina  
(417 fields)

**Top 10:** (Field name, state)

Orlando Paintball, Florida  
Nitro Paintball, Georgia  
Rocky Creek Paintball, Florida  
Hogback Mountain, Virginia  
Skyline Paintball, Virginia  
Paintball Atlanta, Georgia  
Orbital Paintball, Florida  
Jungle Games Paintball, Florida  
Ruff n Tuff Paintball, Florida  
Hurricane Paintball Park, Florida

#### **Area 2:**

**States:** Arizona, New Mexico, Texas, Oklahoma, Hawaii  
(182 fields)

**Top 10:** (Field name, state)

Fun on the run Paintball, Texas  
Predator Paintball, Texas  
Twisted Texas Paintball Games, Texas  
DFW Adventure Park, Texas  
Cow town Paintball, Arizona  
Xdrenalin Paintball, Texas  
D-day Adventure Park, Oklahoma  
Official Paintball Games of Texas, Texas  
Texas Paintball, Texas  
Survival Games of Texas, Texas

#### **Area 3:**

**States:** Alaska, California, Nevada, Utah, Colorado, Wyoming, Idaho, Oregon, Washington, Montana  
(220 fields)

**Top 10:** (Field name, state)

SC Village, California  
Hollywood Sports Park, California  
Conquest Paintball, California  
American Canyon Paintball Jungle, California  
Jungle Island, California  
Camp Pendleton Paintball Park, California  
Action Paintball Games, California  
California Paintball Park, California  
Doodlebug Sports, Washington  
Portland Paintball, Oregon

**Area 4:**

**States:** North Dakota, Minnesota, Wisconsin, Michigan, Ohio, Indiana, Illinois, Iowa, Missouri, South Dakota, Nebraska, Kansas  
(415 fields)

**Top 10:** (Field name, state)

CPX Sports, Illinois (Very close)  
Badlandz Paintball, Illinois (Very close)  
Fox River Games Paintball, Illinois (We interviewed)  
I-70 Paintball, Ohio  
Bushwalkers Paintball, Missouri  
Jaegers Paintball, Missouri  
Fort Knox USA, Indiana  
Rush Paintball, Ohio  
Silver Spur Splat Paintball, Indiana  
Hell Survivors, Michigan

**Area 5:**

**States:** Vermont, New Hampshire, Maine, Maryland, Rhode Island, Connecticut, New Jersey, Delaware, Massachusetts, Pennsylvania, New York, Washington D.C.  
(268 fields)

**Top 10:** (Field name, state)

Skirmish USA, Pennsylvania  
Fox 4 Paintball, Massachusetts  
Matt's Outback, Connecticut  
Hornet's Nest Paintball, New York  
Fireball Paintball, New Jersey  
ABC Paintball, New Jersey  
Liberty Paintball, New York  
Friendly Fire Paintball, Massachusetts  
Boston Paintball, Massachusetts



Outdoor Adventure Bowie, Maryland

**Top 10 States:**

Texas, Florida, New York, California, Ohio, Michigan, Pennsylvania, Georgia, Illinois, Wisconsin

CONFIDENTIAL

Appendix D:

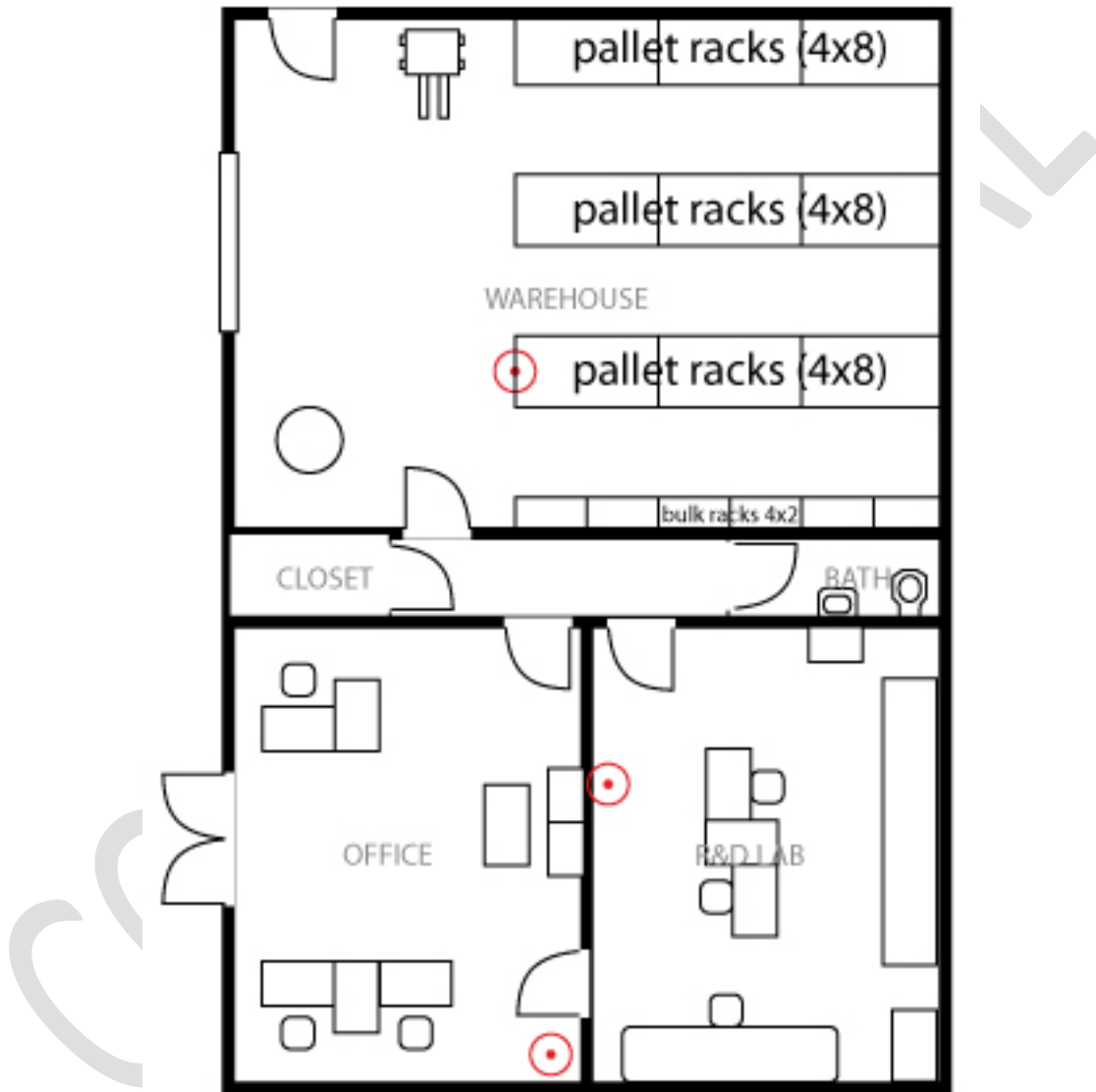
	Smart Specs (3F)	Hand Held GPS	Map and Compass	Walkie Talkie
Hands-Free	X			X
Location Identification	X	X	X	
Real Time Communication	X			X
Portable	X	X	X	X
Easy to Use	X	X		X
Friendly Forces Identification	X			



(Hand Held GPS)

Appendix E:

# **SMART' SPECS**



Page left blank intentionally.

CONFIDENTIAL

# Appendix 4: Technical Team Semester Report

## Table of Contents

### Descriptions

[Arduino Duemilanove](#)

Zigbee Shield for the Arduino Board

[XBEE Pro \(60mW\)](#)

SFE GPS Shield for Arduino

[SANAV FV-M8 \(EB-85A\) 5Hz GPS Engine Module](#)

[HMC6343 3-Axis Digital compass/accelerometer](#)

[Spartan-3E 1600E](#)

### Code and Documentation

Math and Simulation Software Documentation

### Datasheets (attached)

Arduino Duemilanove

Zigbee Shield for the Arduino Board

XBEE Pro (60mw)

SFE GPS Shield for Arduino

SANAV FV-M8 (EB-85A) 5Hz GPS Engine Module

HMC6343 3-Axis Digital compass/accelerometer

Static RAM

Spartan-3E 1600E

## Descriptions

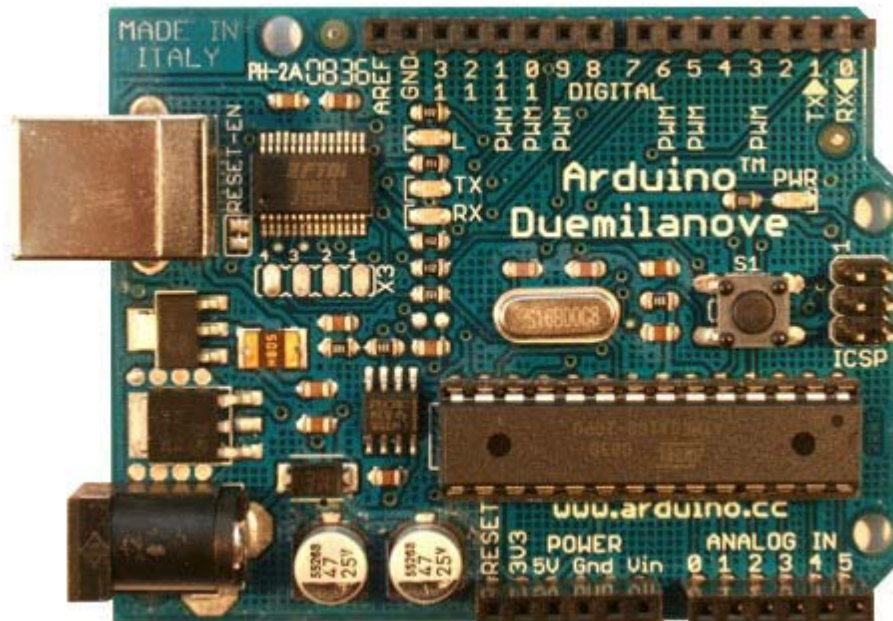
### Arduino Duemilanove

Arduino is a physical computing platform based on a simple open hardware design for a single-board microcontroller, with embedded I/O support and a standard programming language. The Arduino programming language is based on Wiring and is essentially C/C++. Arduino is a microcontroller module with USB connection. It is intended for artists, designers and hobbyists.

Arduino can be used to develop stand-alone interactive objects or can be connected to software on your computer like Flash, Processing, MaxMSP. Arduino board is easy to use in almost any electronic or robotics project. The Arduino Duemilanove is essentially a robot "brain" able to send and receive signals and then take action based on calculations and logic. For more information, see the attached datasheet.

Features:

- 13 digital pins
- 6 analog pins
- Regulated 5V and 3.3V output pins
- Three power options: USB, Wall adapter or Vin/GND pin
- Tx/Rx serial communication pins
- Mini LED connected to pin 13 and status LEDs



## **XBee PRO**

The XBee 60mW Chip Antenna is the wireless unit used for the prototype. Wireless communication is the transfer of information over a distance without the use of wires. The distances involved may be short a few meters as in remote control or long, thousands or millions of kilometers for radio communications. When the context is clear, the term is often shortened to "wireless". This module of wireless supports simple communication point and multi-point networks. This series is 2.40 GHz communication modules take the 802.15.4 stack and lays a very useful serial command set on top of it, making it very popular, very reliable and simple for communication between microcontrollers, computer, and systems.

XBee 802.15.4 features:

- 60 mW chip antenna
- 60 mW output
- 250kbps communication speed
- 3.3V @ 55mA power consumption
- 1500 meter (1 mi) range (in ideal circumstances, but really, these do have good range)
- 6 10-bit ADC input pins
- 8 digital I/O pins
- 128-bit encryption
- Industrial temperature rating (-40° C to 85° C)
- FCC certified for USA, Canada, Australia, & Europe.



## **FV-M8 GPS**

Global Positioning System (GPS) receiver is a device that can tell you exactly where you are on Earth at any moment with clear view of the sky by using three-dimensional location (latitude, longitude, and altitude), provided by GPS satellites from space. GPS is a group of 27 satellites but only 24 in operation and other three as backup in case one fails. Each of these 3,000 to 4,000-pound solar-powered satellites circles the globe at about 12,000 miles (19,300 km), making two complete rotations every day. The orbits are arranged so that at anytime, anywhere on Earth, there are at least four satellites "visible" in the sky.

### Features (FV-M8)

Frequency: 1575.42 MHz; C/A code

Channels: 32 parallel channels

Sensitivity: -158 dBm

Main power input: 3.3V ~ 5V DC input

Current consumption: up to 63mA

Operating temperature: -30C to +80C

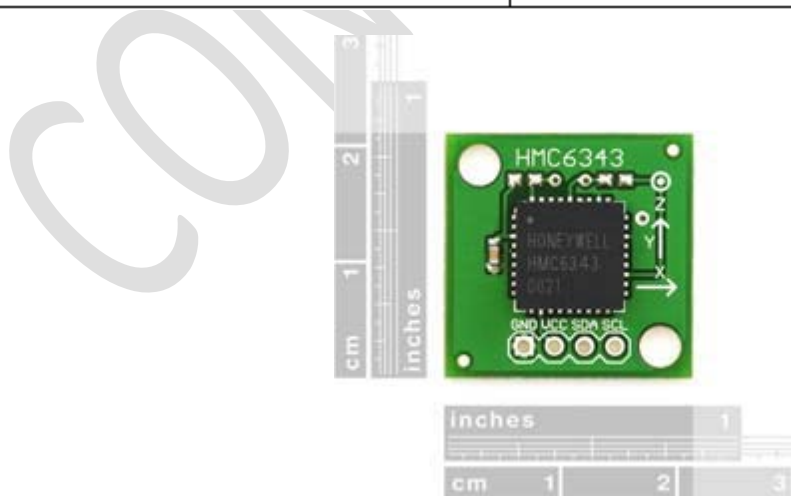




## Digital Compass

Compass Module with Tilt Compensation is the latest IC module in compass arena. These modules have 3-axis magnetic sensor and 3-axis accelerometer, analog and digital support circuits and integrated with programmable in PIC core running all the calculation. By combining all those packages the direction of the compass will remain the same even when the board is tilted. This compass is used in devices such as binoculars, cameras, night vision optics, laser ranger finders, antenna positioning, and other industrial compassing applications.

Features	Benefits
Compass with heading/tilt outputs	A complete compass solution including compass firmware
3-axis MR sensors, accelerometers and a microprocessor in a single package	A digital compass solution with heading and tilt angle outputs in a chip-scale package
Compass algorithms	For computation of heading, and magnetic calibration for hard-iron
9 x 9 x 1.9mm LCC Surface Mount Package	Small size, easy to assemble and compatible with high speed
Low voltage operations	Compatible with battery powered applications
EEPROM memory	To store compass data for processor routines
Digital Serial Data Interface	I2C Interface, easy to use 2-wire communication for heading output
Moderate Precision Outputs	Typical 2° Heading Accuracy with 1° Pitch and Roll Accuracy
Lead Free Package Construction	Complies with RoHS environmental standards
Flexible Mounting	Can be mounted on horizontal or vertical circuit boards



## Spartan-3E 1600E

A comprehensive development kit of hardware, design tools, IP and pre-verified reference designs can rapidly accelerate your embedded development. The Spartan®-3E 1600E Edition of the MicroBlaze™ Development Kit includes the SP3E1600E development board, Platform Studio embedded tool suite and ISE® design software, supporting MicroBlaze software processor design. This kit is RoHS compliant and also includes the power adapters for US, UK, and Europe.

### Features:

Xilinx Devices: XC3S1600E-4FG320C FPGA, XC9572XL CPLD

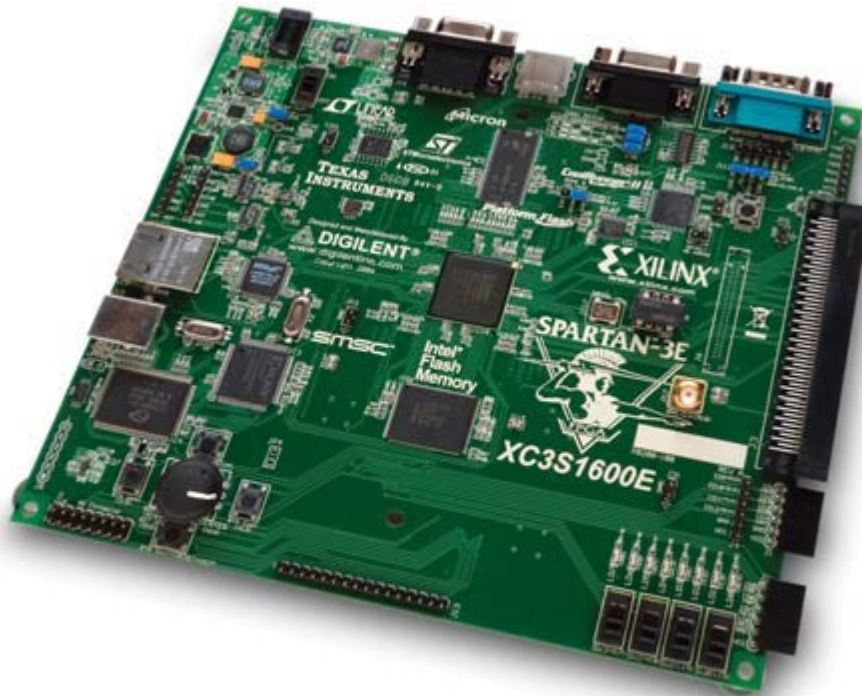
Flexible Spartan-3E SP3E16 Development Board

Complete Platform Studio embedded tool suite

ISE WebPACK™ FPGA design software

Pre-Verified Reference Designs

Documentation, USB A/B download cable, UART and Ethernet cables and power supply



# Math and Simulation Software Documentation

- I. [Problem](#)
- II. [Solutions Roadmap](#)
- III. [Solution](#)
- IV. Files
  - 1. matrix.cpp
  - 2. matrix.h
  - 3. matrix\_structures.h
  - 4. hud.cpp
  - 5. hud.h
  - 6. hud\_structures.h
  - 7. math.h
  - 8. opengl\_structure.h
  - 9. project.cpp
- V. [Structures](#)
  - 1. [COMPASS](#)
  - 2. [GPSOBJ](#)
  - 3. [SCREEN](#)
  - 4. [DRAW COORDS](#)
  - 5. [MATRIX3](#)
  - 6. [CMATRIX](#)
  - 7. [CHANGES](#)
  - 8. [CAMERA](#)
  - 9. [WINDOW](#)
- VI. Functions
  - 1. matMultiply (MATRIX3)
  - 2. matMultiply (CMATRIX)
  - 3. matSubtract
  - 4. hudMakeCMatrix
  - 5. hudMakeTransform
  - 6. hudFloatCoords
  - 7. hudIsInView
  - 8. hudScreenCalibration

## Problem

Given the GPS coordinates of the user, a user's ally, the user's compass readings, and the screen calibration, place a marker on the user's screen where the ally is.

Given variables: GPS coordinates user, GPS coordinates ally, Compass readings user

Needed variables: Screen coordinates user

## Solutions Roadmap

First attempted solution: Using various trigonometric functions for transforming 3D points into a 2D space as calculated by Teague. When the math was placed into the simulation, it did not work at all and attempts to figure out why produced no results.

Second attempt: Using vector projection to find the coordinates projected onto the screen. This resulted in similar results to the above attempt and was discarded after a day.

Third attempt: Matrix transforms for 3D projection. The issues with this attempt were based on the simulator – the GPS x coordinate and Compass heading angle had to be reversed.

## Solution

Matrix transformations. Given the GPS coordinates of the user as a column matrix  $c$  and the GPS coordinates of the user's ally as a column matrix  $a$ , and the heading, pitch, and roll of the user's viewpoint, one uses the following equation to get the column matrix  $d$ :

$$\begin{bmatrix} dx \\ dy \\ dz \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 \cos \theta_x - \sin \theta_x & 0 & 1 & 0 \\ 0 \sin \theta_x & \cos \theta_x & -\sin \theta_y & 0 \end{bmatrix} \begin{bmatrix} \cos \theta_y & 0 & \sin \theta_y \\ \cos \theta_z - \sin \theta_z & 0 \\ \sin \theta_z & \cos \theta_z & 0 \end{bmatrix} \begin{bmatrix} ax \\ ay \\ az \end{bmatrix} = \begin{bmatrix} cx \\ cy \\ cz \end{bmatrix}$$

We then use matrix  $d$  to calculate the coordinates of the marker for  $a$  ally using the distance of the viewer from the screen  $e$ :

$$\begin{aligned} bx &= (dx - ex)(ez / dz) \\ by &= (dy - ey)(ez / dz) \end{aligned}$$

Note: depending on how the program doing the drawing defines the axis of the screen, there may need to be transformations done to the angles given and the final drawing coordinates received.

## Files

Files for HUD math:

- matrix.cpp
  - Matrix functions written to save space on arduino by not using the svl library and objects, but structures and only necessary functions
  - Header: matrix.h
  - Structure file: matrix\_structures.h
  - Functions:

- MATRIX3 matMultiply(MATRIX3 a, MATRIX3 b)
  - CMATRIX matMultiply(MATRIX3 a, CMATRIX b)
  - CMATRIX matSubtract(CMATRIX a, CMATRIX b)
- matrix.h
  - Declares functions for matrix.cpp
  - Relies on:
    - matrix\_structures.h
- matrix\_structures.h
  - Structures:
    - MATRIX3
    - CMATRIX
- hud.cpp
  - Header: hud.h
  - Structure files: hud\_structures.h, matrix\_structures.h
  - Functions:
    - CMATRIX hudMakeCMatrix(GPSOBJ obj)
    - MATRIX3 hudMakeTransformX, hudMakeTransformY, hudMakeTransformZ(int angle)
    - DRAW\_COORDS hudFloatCoords(GPSOBJ device, GPSOBJ ally, COMPASS comp, SCREEN e)
    - bool hudIsInView(GPSOBJ device, GPSOBJ ally, COMPASS compass)
    - SCREEN hudScreenCalibrationX, hudScreenCalibrationY, hudScreenCalibrationZ(float depth, SCREEN s)
- hud.h
  - Declares functions for hud.cpp
  - Relies on:
    - hud\_structures.h
    - math.h
    - matrix.h
- hud\_structures.h
  - Defines structures used in calculations
  - Structures:
    - COMPASS
    - GPSOBJ
    - SCREEN
- math.h
  - Standard math library for C++

Files for simulator:

These are needed only for the simulator

- opengl\_structures.h
  - Defines structures used for rendering the simulator
  - Structures:
    - CAMERA
    - WINDOW
    - CHANGES
- project.cpp
  - Renders the simulation and calls the hudFloatCoords for testing math
  - Relies on:
    - hud.h
    - opengl\_structures.h
    - movement\_structures.h
    - hud\_structures.h

### Defined Structures

- GPSOBJ
  - file: hud\_structures.h
  - float x, float y, float z
  - Structure to contain a set of GPS coordinates
- COMPASS
  - file: hud\_structures.h
  - float heading, float pitch, float roll
  - Structure to contain readings from the compass
- SCREEN
  - hud\_structures.h
  - float depthx, float depthy, float depthz
  - Structure to contain screen calibration settings (distance from the user along the respective axis)
- DRAW\_COORDS
  - hud\_structures.h
  - float x, float y
  - Structure to contain coordinates on the screen
- MATRIX3
  - matrix\_structures.h
  - float index[3][3]
  - Contains a 3x3 matrix

- CMATRIX
  - matrix\_structures.h
  - float index[3]
  - Contains 3 numbers representing a column matrix
- CHANGES
  - opengl\_structures.h
  - float lx, ly, lz, deltaAngle, deltaMove
  - Used by the simulator to move the camera
- CAMERA
  - opengl\_structures.h
  - float angle, ratio
  - Used by the simulator to adjust the camera
- WINDOW
  - opengl\_structures.h
  - int height, width, border, main, hud, overhead
  - Used by the simulator to contain the values for the main window (height, width, border) and the window ids (main, hud, overhead)

### Functions

- MATRIX3 matMultiply(MATRIX3 a, MATRIX3 b)
  - matrix.cpp
  - Parameters: two 3x3 matrices
  - Returns: one 3x3 matrix which is the result of a\*b
- CMATRIX matMultiply(MATRIX3 a, CMATRIX b)
  - matrix.cpp
  - Parameters: one 3x3 matrix and one 3x1 matrix
  - Returns: 3x1 which is the result of a\*b
- CMATRIX matSubtract(CMATRIX a, CMATRIX b)
  - matrix.cpp
  - Parameters: two 3x1 matrices
  - Returns: 3x1 matrix which is the result of a-b
- CMATRIX hudMakeCMatrix(GPSOBJ obj)
  - hud.cpp
  - Parameters: GPS coordinates
  - Returns: 3x1 Column matrix of the gps coordinates
- MATRIX3 hudMakeTransformX, hudMakeTransformY, hudMakeTransformZ(int angle)
  - hud.cpp
  - Parameters: angle of the user's view about the respective axis

- Returns: 3x3 matrix for the transform equation for the respective angle
- DRAW\_COORDS hudFloatCoords(GPSOBJ device, GPSOBJ ally, COMPASS comp, SCREEN e)
  - hud.cpp
  - Parameters: gps coordinates of the user (device) and ally, the compass readings for the user, and the screen distance from the user
  - Returns: Set of x,y coordinates for drawing the ally marker on the screen using the equations defined in the Solution area.
- bool hudIsInView(GPSOBJ device, GPSOBJ ally, COMPASS compass)
  - hud.cpp
  - Parameters: gps coordinates of the user and ally, and compass readings for user
  - Returns: true if the ally is in view of the user, false if not
- SCREEN hudScreenCalibrationX, hudScreenCalibrationY, hudScreenCalibrationZ(float depth, SCREEN s)
  - hud.cpp
  - Parameters: new depth along the respective axis, SCREEN structure to update
  - Returns: updated SCREEN structure



# Appendix 5: Prototype Code

## Accelerometer Code

```
#include <Wire.h>

//accelerometer code-----

/*
   HMC6343 uses 3.3v VCC
   Hardcoded value for HMC6343Address assumes:
       SCL connected to Arduino analog in pin 5
       SDA connected to Arduino analog in pin 4
*/

int HMC6343Address = 0x32 >> 1;
float headerBuffer[4];
int c = 0; //Loop counter variable

struct
{
    float heading;
    float pitch;
    float roll;
} accelerometer;

void setupAccelerometer()
{
    //Initialize the Wire library
    Wire.begin();

    //Startup time of HMC6343 microcontroller is ~500ms
    delay(500);
}

void readAccelerometer(int c)
{
    int i, h;
    byte headingData[6];
    i = 0;

    //Transmit 'Get Data' request
    Wire.beginTransmission(HMC6343Address);
    Wire.send(0x50);
    Wire.endTransmission();

    //Delay for response processing - this could be as low as 2ms
    delay(10);

    //Request 6-byte 3-axis accelerometer data
```

```

Wire.requestFrom(HMC6343Address, 6);

//Flush data to buffer
while(Wire.available() && i < 6)
{
    headingData[i] = Wire.receive();
    i++;
}

//Place data into buffer and calculate mean
h = (float)(headingData[0] * 256 + headingData[1]) / 10;
if (headerBuffer[0] == h)
{
    headerBuffer = {h, h, h, h};
    accelerometer.heading = h;
}
else
{
    headerBuffer[3] = headerBuffer[2];
    headerBuffer[2] = headerBuffer[1];
    headerBuffer[1] = headerBuffer[0];
    headerBuffer[0] = h;
    if(c % 4 == 0)
        accelerometer.heading = (headerBuffer[0] + headerBuffer[1]
+ headerBuffer[2] + headerBuffer[3])/4;
}

//Place data into struct
accelerometer.pitch = (float)(headingData[2] * 256 +
headingData[3]) / 10;
accelerometer.roll = (float)(headingData[4] * 256 +
headingData[5]) / 10;
}

//end accelerometer code-----

void setup()
{
    //Initialize accelerometer
    setupAccelerometer();

    //Initialize debugging output
    Serial.begin(9600);
}

void loop()
{
    //Increment loop counter
    c++;

    //Read accelerometer data
    readAccelerometer(c);
}

```

```
    //Debugging output
    Serial.print("Current heading: ");
    Serial.print(accelerometer.heading);
    Serial.println(" degrees");

    Serial.print("Current pitch: ");
    Serial.print(accelerometer.pitch);
    Serial.println(" degrees");

    Serial.print("Current roll: ");
    Serial.print(accelerometer.roll);
    Serial.println(" degrees");

    delay(125);
}
```

CONFIDENTIAL

## GPU Code

```
#include "WProgram.h"

void gpuSetup(unsigned char tx)
{
    pinMode(tx, OUTPUT);
    digitalWrite(tx, HIGH);
}

unsigned short gpuColor(unsigned int red, unsigned int green, unsigned
int blue)
{
    return (unsigned short)((red & 15) | ((green & 15) << 4) | ((blue
& 15) << 8));
}

void gpuLine(unsigned char tx, unsigned int baud, unsigned short
color, unsigned short x, unsigned short y, unsigned short x2, unsigned
short y2)
{
    unsigned int transmitDelay = 1000000 / baud;
    unsigned int spacerDelay = transmitDelay * 9;

    unsigned int p1, p2;
    unsigned long p;

    //Calculate Parity bit 1 (even parity)
    //00 01 RRRR GGGG BBBB
    p = 0;
    p |= 1 << 3;
    p |= (color & 4095) << 4;

    while (p)
    {
        p1 += (p & 1);
        p >>= 1;
    }

    //Calculate Parity bit 2 (even parity)
    //00 01 RRRR GGGG BBBB 01 xx xxxx xxxx yyyy 10 yy yyyy XXXX
XXXX 11 XX YYYY YYYY YY P0

    p = 0;

    p |= (long)1 << 3;
    p |= (long)(color & 4095) << 5;

    p |= (long)1 << 17;
    p |= (long)(x & 1023) << 18;
    p |= (long)(y & 15) << 28;
```

```

p |= (long)1 << 32;
p |= (long)((y >> 4) & 63) << 34;
p |= (long)(x & 255) << 40;

p |= (long)3 << 48;
p |= (long)((x >> 8) & 3) << 50;
p |= (long)(y & 1023) << 52;
p |= (p1 & 1) << 62;

while (p)
{
    p2 += (p & 1);
    p >>= 1;
}

//F LL MM RRRR GGGG BBBB F SSSS SSSS
//0 00 01 XXXX XXXX XXXX 1 1111 1111

//F
digitalWrite(tx, LOW);
delayMicroseconds(transmitDelay);

//LL
digitalWrite(tx, LOW);
delayMicroseconds(transmitDelay);
digitalWrite(tx, LOW);
delayMicroseconds(transmitDelay);

//MM
digitalWrite(tx, LOW);
delayMicroseconds(transmitDelay);
digitalWrite(tx, HIGH);
delayMicroseconds(transmitDelay);

//RRRR
digitalWrite(tx, color & 8);
delayMicroseconds(transmitDelay);
digitalWrite(tx, color & 4);
delayMicroseconds(transmitDelay);
digitalWrite(tx, color & 2);
delayMicroseconds(transmitDelay);
digitalWrite(tx, color & 1);
delayMicroseconds(transmitDelay);

//GGGG
digitalWrite(tx, color & 128);
delayMicroseconds(transmitDelay);
digitalWrite(tx, color & 64);
delayMicroseconds(transmitDelay);
digitalWrite(tx, color & 32);

```

```

delayMicroseconds(transmitDelay);
digitalWrite(tx, color & 16);
delayMicroseconds(transmitDelay);

//BBBB
digitalWrite(tx, color & 2048);
delayMicroseconds(transmitDelay);
digitalWrite(tx, color & 1024);
delayMicroseconds(transmitDelay);
digitalWrite(tx, color & 512);
delayMicroseconds(transmitDelay);
digitalWrite(tx, color & 256);
delayMicroseconds(transmitDelay);

//F SSSS SSSS
digitalWrite(tx, HIGH);
delayMicroseconds(spacerDelay);

//F LL xx xxxx xxxx yyyy F SSSS SSSS
//0 01 XX XXXX XXXX XXXX 1 1111 1111

//F
digitalWrite(tx, LOW);
delayMicroseconds(transmitDelay);

//LL
digitalWrite(tx, LOW);
delayMicroseconds(transmitDelay);
digitalWrite(tx, HIGH);
delayMicroseconds(transmitDelay);

//xx
digitalWrite(tx, x & 512);
delayMicroseconds(transmitDelay);
digitalWrite(tx, x & 256);
delayMicroseconds(transmitDelay);

//xxxx
digitalWrite(tx, x & 128);
delayMicroseconds(transmitDelay);
digitalWrite(tx, x & 64);
delayMicroseconds(transmitDelay);
digitalWrite(tx, x & 32);
delayMicroseconds(transmitDelay);
digitalWrite(tx, x & 16);
delayMicroseconds(transmitDelay);

//xxxx
digitalWrite(tx, x & 8);
delayMicroseconds(transmitDelay);
digitalWrite(tx, x & 4);

```

```

delayMicroseconds(transmitDelay);
digitalWrite(tx, x & 2);
delayMicroseconds(transmitDelay);
digitalWrite(tx, x & 1);
delayMicroseconds(transmitDelay);

//yyyy
digitalWrite(tx, y & 512);
delayMicroseconds(transmitDelay);
digitalWrite(tx, y & 256);
delayMicroseconds(transmitDelay);
digitalWrite(tx, y & 128);
delayMicroseconds(transmitDelay);
digitalWrite(tx, y & 64);
delayMicroseconds(transmitDelay);

//F SSSS SSSS
digitalWrite(tx, HIGH);
delayMicroseconds(spacerDelay);

//F LL yy yyyy XXXX XXXX F SSSS SSSS
//0 10 XX XXXX XXXX XXXX 1 1111 1111

//F
digitalWrite(tx, LOW);
delayMicroseconds(transmitDelay);

//LL
digitalWrite(tx, HIGH);
delayMicroseconds(transmitDelay);
digitalWrite(tx, LOW);
delayMicroseconds(transmitDelay);

//yy
digitalWrite(tx, y & 32);
delayMicroseconds(transmitDelay);
digitalWrite(tx, y & 16);
delayMicroseconds(transmitDelay);

//yyyy
digitalWrite(tx, y & 8);
delayMicroseconds(transmitDelay);
digitalWrite(tx, y & 4);
delayMicroseconds(transmitDelay);
digitalWrite(tx, y & 2);
delayMicroseconds(transmitDelay);
digitalWrite(tx, y & 1);
delayMicroseconds(transmitDelay);

//XXXX
digitalWrite(tx, x2 & 512);

```

```

delayMicroseconds(transmitDelay);
digitalWrite(tx, x2 & 256);
delayMicroseconds(transmitDelay);
digitalWrite(tx, x2 & 128);
delayMicroseconds(transmitDelay);
digitalWrite(tx, x2 & 64);
delayMicroseconds(transmitDelay);

//XXXX
digitalWrite(tx, x2 & 32);
delayMicroseconds(transmitDelay);
digitalWrite(tx, x2 & 16);
delayMicroseconds(transmitDelay);
digitalWrite(tx, x2 & 8);
delayMicroseconds(transmitDelay);
digitalWrite(tx, x2 & 4);
delayMicroseconds(transmitDelay);

//F SSSS SSSS
digitalWrite(tx, HIGH);
delayMicroseconds(spacerDelay);

//F LL XX YYYY YYYY YPPP F SSSS SSSS
//0 11 XX XXXX XXXX XXXX 1 1111 1111

//F
digitalWrite(tx, LOW);
delayMicroseconds(transmitDelay);

//LL
digitalWrite(tx, HIGH);
delayMicroseconds(transmitDelay);
digitalWrite(tx, HIGH);
delayMicroseconds(transmitDelay);

//XX
digitalWrite(tx, x2 & 2);
delayMicroseconds(transmitDelay);
digitalWrite(tx, x2 & 1);
delayMicroseconds(transmitDelay);

//YYYY
digitalWrite(tx, y2 & 512);
delayMicroseconds(transmitDelay);
digitalWrite(tx, y2 & 256);
delayMicroseconds(transmitDelay);
digitalWrite(tx, y2 & 128);
delayMicroseconds(transmitDelay);
digitalWrite(tx, y2 & 64);

//YYYY

```



```

delayMicroseconds(transmitDelay);
digitalWrite(tx, y2 & 32);
delayMicroseconds(transmitDelay);
digitalWrite(tx, y2 & 16);
delayMicroseconds(transmitDelay);
digitalWrite(tx, y2 & 8);
delayMicroseconds(transmitDelay);
digitalWrite(tx, y2 & 4);
delayMicroseconds(transmitDelay);

//YY
digitalWrite(tx, y2 & 2);
delayMicroseconds(transmitDelay);
digitalWrite(tx, y2 & 1);
delayMicroseconds(transmitDelay);

//PP
digitalWrite(tx, p1 & 1);
delayMicroseconds(transmitDelay);
digitalWrite(tx, p2 & 1);
delayMicroseconds(transmitDelay);

//F SSSS SSSS
digitalWrite(tx, HIGH);
delayMicroseconds(spacerDelay);
}

void gpuFlip(unsigned char tx, unsigned int baud, bool screen)
{
    unsigned int transmitDelay = 1000000 / baud;
    unsigned int initDelay = transmitDelay * 3;
    unsigned int dontCareDelay = transmitDelay * 12;
    unsigned int spacerDelay = transmitDelay * 9;

    //F LL MM XXXX XXXX XXXX F SSSS SSSS
    //0 00 1? 0000 0000 0000 1 1111 1111

    //F LL
    digitalWrite(tx, LOW);
    delayMicroseconds(initDelay);

    //MM
    digitalWrite(tx, HIGH);
    delayMicroseconds(transmitDelay);
    digitalWrite(tx, screen);
    delayMicroseconds(transmitDelay);

    //XXXX XXXX XXXX
    digitalWrite(tx, LOW);
    delayMicroseconds(dontCareDelay);

    //F SSSS SSSS

```

```
digitalWrite(tx, HIGH);  
delayMicroseconds(spacerDelay);  
}
```

CONFIDENTIAL

## GPU Example – Fill Screen

```
#include <gpu.h>

/*
    Fills the screen with a shade of grey and performs a page flip
*/

#define GPU_PIN 4
#define GPU_BAUD 50000

int shade = 7;
bool screen = 0;

void setup()
{
    gpuSetup(GPU_PIN);

    Serial.begin(9600);

    delay(500);

    Serial.println("Starting GPU Test...");
}

void loop()
{
    int i = 0;
    unsigned short color = gpuColor(shade, shade, shade);

    for(i = 0; i < 480; i++)
    {
        gpuLine(GPU_PIN, GPU_BAUD, color, 0, i, 639, i);
    }

    screen = !screen;

    gpuFlip(GPU_PIN, GPU_BAUD, screen);

    Serial.println("Finished Rendering Frame");
}
```

## HUD

```
/*
 * HUD
 *
 * The hud code has to take the gps coordinates of a device and an
 ally and a
 * compass and find the projection of the device's screen of that
 ally.
 */

#ifndef HUD_H
#define HUD_H

#include "hud_structures.h"
#include "math.h"
#include "matrix.h"

#define DEG2RAD M_PI/180
#define RAD2DEG 180/M_PI
#define HUD_WIDTH 4.3
#define HUD_HEIGHT 2.5

// Functions to create the transform matrices
// based on the angle about the relevant axis
MATRIX3 hudMakeTransformX(int angle);
MATRIX3 hudMakeTransformY(int angle);
MATRIX3 hudMakeTransformZ(int angle);

// Returns a screen object when given a new depth for the screen
SCREEN hudScreenCalibrationX(float depth, SCREEN s);
SCREEN hudScreenCalibrationY(float depth, SCREEN s);
SCREEN hudScreenCalibrationZ(float depth, SCREEN s);

// Function to return the x,y coordinate to draw on the screen to mark
the ally
// based on the device, the compass readings for the device, and the
distance away
// from the screen the viewer is
DRAW_COORDS hudFloatCoords(GPSOBJ device, GPSOBJ ally, COMPASS comp,
SCREEN e) ;

// Creates a column matrix with the given gps coordinates
CMATRIX hudMakeCMatrix(GPSOBJ obj);

// Determines if the ally is in the field of view
bool hudIsInView(GPSOBJ device, GPSOBJ ally, COMPASS compass);

#endif
```

## HUD FUNCTIONS

```
/*
*****
* HUD functions
*****
*/

#include "hud.h"
#include <math.h>

CMATRIX hudMakeCMatrix(GPSOBJ obj)
{
    CMATRIX matrix; // Initialize to identity matrix

    matrix.index[0] = 0-obj.x;
    matrix.index[1] = obj.y;
    matrix.index[2] = obj.z;

    return matrix;
}

// Angles have to be passed as integer as there are issues
// if they are passed as floats
// They are converted once passed

MATRIX3 hudMakeTransformX(int angle)
{
    float ang = angle * DEG2RAD;
    MATRIX3 result;
    result.index[0][0] = 1;
    result.index[0][1] = result.index[0][2] = result.index[1][0] =
        result.index[2][0] = 0;
    result.index[1][1] = result.index[2][2] = cos(ang);
    result.index[2][1] = sin(ang);
    result.index[1][2] = 0 - result.index[2][1];

    return result;
}

MATRIX3 hudMakeTransformY(int angle)
{
    // If changing the heading produces opposite results, switch the
    sign of the angle
    float ang = 0-angle * DEG2RAD;
    MATRIX3 result;
    result.index[1][1] = 1;
    result.index[0][1] = result.index[1][0] = result.index[1][2]
        = result.index[2][1] = 0;
    result.index[0][0] = result.index[2][2] = cos(ang);
    result.index[0][2] = sin(ang);
    result.index[2][0] = 0 - result.index[0][2];
}
```

```

        return result;
    }

MATRIX3 hudMakeTransformZ(int angle)
{
    float ang = angle * DEG2RAD;
    MATRIX3 result;
    result.index[2][2] = 1;
    result.index[2][0] = result.index[2][1] = result.index[0][2]
        = result.index[1][2] = 0;
    result.index[0][0] = result.index[1][1] = cos(ang);
    result.index[1][0] = sin(ang);
    result.index[0][1] = 0 - result.index[1][0];

    return result;
}

/*
 *
 * floatCoords
 *
 * Uses the equation  $D = T_x * T_y * T_z * (A-C)$ 
 * Where D is a resultant matrix used in the next equation,
 *  $T_x$ ,  $T_y$ , and  $T_z$  are transformation matrices based on the angle of
the camera
 * around the respective axis
 * A is the point in 3D space
 * C is the camera
 *
 * We then take D and find our drawing points b using the equations:
 *  $b_x = (d_x - e_x)/(e_z/d_z)$ 
 *  $b_y = (d_y - e_y)/(e_z/d_z)$ 
 * Where the d's are taken from D (a 1x3 matrix), and the e's are the
viewer's
 * position relative to the display surface
 */
DRAW_COORDS hudFloatCoords(GPSOBJ device, GPSOBJ ally,
                            COMPASS comp, SCREEN e)
{
    CMATRIX c = hudMakeCMatrix(device);
    CMATRIX a = hudMakeCMatrix(ally);
    MATRIX3 tX = hudMakeTransformX(comp.roll);
    MATRIX3 tY = hudMakeTransformY(comp.heading);
    MATRIX3 tZ = hudMakeTransformZ(comp.pitch);

    MATRIX3 temp = matMultiply(tX,tY);
    temp = matMultiply(temp,tZ);

    CMATRIX d = matMultiply(tY, matSubtract(c,a));

    DRAW_COORDS b;

```

```

// Find if not in range of vision
if(hudIsInView(device, ally, comp))
{
    b.x = (d.index[0] - e.depthx) * (e.depthz/d.index[2]);
    b.y = (d.index[1] - e.depthy) * (e.depthz/d.index[2]);
}
else
{
    // Impossible number - size of screen + 1
    b.x = HUD_WIDTH + 1;
    b.y = HUD_HEIGHT + 1;
}

return b;
}

bool hudIsInView(GPSOBJ device, GPSOBJ ally, COMPASS compass)
{
    float distance = sqrt(pow(ally.x-device.x,2)+pow(ally.y-
device.z,2)
                    +pow(ally.z-device.z,2));
    float angle = (float)(acos((ally.z -
device.z)/distance))*RAD2DEG;
    if(sqrt(pow(angle - compass.heading,2)) >= 90)
    {
        return true;
    }
    return false;
}

/*
 * Calibration method to change screen's position relative to user
 */
SCREEN hudScreenCalibrationX(float depth, SCREEN s)
{
    s.depthx = depth;
    return s;
}

SCREEN hudScreenCalibrationY(float depth, SCREEN s)
{
    s.depthy = depth;
    return s;
}

SCREEN hudScreenCalibrationZ(float depth, SCREEN s)
{
    s.depthz = depth;
    return s;
}

```

## HUD STRUCTURES

```
#ifndef HUD_STRUCTURES_H
#define HUD_STRUCTURES_H

typedef struct {
    float x;
    float y;
    float z;
} GPSOBJ;

// Values obtained from compass
typedef struct {
    float heading;
    float pitch;
    float roll;
} COMPASS;

typedef struct
{
    float x;
    float y;
} DRAW_COORDS;

#endif
```



## INPUT

```
#include "input.h"
#include "hud.h"
#include "hud_structures.h"

GPSOBJ device;
COMPASS compass;
SCREEN screen = {0, 0, -0.1524};

//setter for the headset coordinates
//because I don't know where to get them from
void setHeadCoords(float x, float y, float z) {
    device.x = x;
    device.y = y;
    device.z = z;
}

//commenting this out because there are global variables somewhere
/* void setCompass(float head, float pitch, float roll) {
    compass.heading = head;
    compass.pitch = pitch;
    compass.roll = roll;
} */

//takes two x,y coordinate structs and draws a line
void lineDraw(DRAW_COORDS c1, DRAW_COORDS c2) {

}

void drawRect(DRAW_COORDS point, int size) {
    DRAW_COORDS one;
    one.x = point.x + size;
    one.y = point.y + size;
    DRAW_COORDS two;
    one.x = point.x + size;
    one.y = point.y - size;
    DRAW_COORDS three;
    one.x = point.x - size;
    one.y = point.y - size;
    DRAW_COORDS four;
    one.x = point.x - size;
    one.y = point.y + size;
    lineDraw(one, two);
    lineDraw(two, three);
    lineDraw(three, four);
    lineDraw(four, one);
}

void render(GPSOBJ *coords, int num) {
    int size;
```

```
float distance;
for(int i = 0; i < num; i++) {
    DRAW_COORDS b;
    distance = sqrt(pow(device.x-coords[i].x,2) + pow(device.y-
coords[i].y,2) + pow(device.z-coords[i].z,2));
    size = 100 - (int)(distance/5); //distance is divided by a scaling
factor
    if(size < 2) size = 2;

    //compass is global somewhere, but we still need to modify our
struct
    //these variable names will need to be changed to reflect the
actual variables
    //    compass.heading = compass_heading;
    //    compass.pitch = compass_pitch;
    //    compass.roll = compass_roll;

    b = hudFloatCoords(device, coords[i], compass, screen);
    drawRect(b, size);
}
}
```

## INPUT.H

```
#ifndef INPUT_H
#define INPUT_H

#include "hud_structures.h"

void setHeadCoords(float x, float y, float z);

void render(GPSOBJ *coords, int num);

void drawRect(DRAW_COORDS point, int size);

void lineDraw(DRAW_COORDS c1, DRAW_COORDS c2);

#endif
```

CONFIDENTIAL

## MATRIX

```
#include "matrix.h"

// A * B
MATRIX3 matMultiply(MATRIX3 a, MATRIX3 b)
{
    MATRIX3 result;

    int i, j;
    for (j = 0; j < 3; j++)
    {
        for(i = 0; i < 3; i++)
        {
            // Each result is the dot of row a[i] with column b[j]
            result.index[i][j] = a.index[i][0] * b.index[0][j]
                                + a.index[i][1] * b.index[1][j]
                                + a.index[i][2] * b.index[2][j];
        }
    }

    return result;
}

// A * B
CMATRIX matMultiply(MATRIX3 a, CMATRIX b)
{
    CMATRIX result;

    int i;
    for (i = 0; i < 3; i++)
    {
        // Result is the dot of row a[i] with column b[j=0]
        result.index[i] = a.index[i][0] * b.index[0]
                        + a.index[i][1] * b.index[1]
                        + a.index[i][2] * b.index[2];
    }

    return result;
}

// A - B
CMATRIX matSubtract(CMATRIX a, CMATRIX b)
{
    CMATRIX result;

    for(int i = 0; i < 3; i++)
    {
        result.index[i] = a.index[i] - b.index[i];
    }
}
```

```
}    return result;
```

CONFIDENTIAL

## MATRIX.H

```
#ifndef MATRIX_H
#define MATRIX_H

#include "matrix_structures.h"

// Overloaded function - takes two 3x3 matrices and multiplies them
// A * B
MATRIX3 matMultiply(MATRIX3 a, MATRIX3 b);

// Overloaded function - takes a 3x3 matrix and a 3x1 matrix and
multiplies them for
// a 3x1 matrix
// A * B
CMATRIX matMultiply(MATRIX3 b, CMATRIX a);

// Column matrices: A - B
CMATRIX matSubtract(CMATRIX a, CMATRIX b);

// Printing functions
void matPrintOut(MATRIX3);
void matPrintOut(CMATRIX);

#endif
```

## MATRIX STRUCTURES

```
typedef struct  
{  
    float index[3][3];  
} MATRIX3;
```

```
typedef struct  
{  
    float index[3];  
} CMATRIX;
```

CONFIDENTIAL

## DIRECTX EXAMPLE

```
//Undefining this will remove the debugging console
#define CONSOLE
#define WIDTH_PX 320
#define HEIGHT_PX 240
//should be floats?
#define WIDTH_M 0.04
#define HEIGHT_M 0.03
#define PI 3.141592653589

#include "dx.h"
#include <math.h>

#ifdef CONSOLE
    #include "stdio.h"
#endif

typedef struct {
    float x;
    float y;
    float z;
} gpsobj;

typedef struct {
    float heading;
    float pitch;
    float roll;
} compass;

//Globals-----
-----

TEXTURE bg;
gpsobj device;
gpsobj object;
compass accelerometer;
float distancev = 0; //vertical distance
float distanceh = 0; //horizontal distance
float theta1 = 0; //bearing from compass
float theta1p = 0; //pitch bearing from compass
float theta1l = 0; //stores bearing for approximations later
float theta1lp = 0; //stores bearing for approximations later
float theta2 = 0;
float theta2p = 0;
float depth = 0;
float disp = 0;
float dispp = 0;
float screen_depth = 0.01; //changed during calibration
int coords[2];
```



```

//Draw the scene-----
-----
//Note: The screen is cleared between EVERY call to render()

void render(void)
{
    /*int i;
    for (i = 100; i < 400; i++){
        DX_PSETbig(i, 100, 0xFF00FF00);
    }*/
    DX_PSETbig(coords[0], coords[1], 0xFF00FF00);
    //printf("%d\n", thetemp);
    //Note: setting keycode to null is necessary to prevent
duplicate reads
/* #ifdef CONSOLE
    if (DX_KEYBOARD.keycode && !DX_KEYBOARD.up)
    {
        printf("Key pressed: %u\n", DX_KEYBOARD.keycode);
        DX_KEYBOARD.keycode = NULL;
    }
#endif */
}

//
//      z  y
//      |  /
//  _____/____x
//      /  |
//      /  |

void floatCoords() {
//for use once in a while for corrections
    distanceh = sqrt(pow(object.x-device.x,2) + pow(object.y-
device.y,2));
    distancev = sqrt(pow(object.z-device.z,2) + pow(object.y-
device.y,2));
    theta2 = asin((object.y-device.y)/distanceh);
    theta2p = asin((object.y-device.y)/distancev);
    thetal1 = thetal;
    thetalp = thetalp;
    depth = cos(1.57079633-thetal-theta2)*distanceh;
    disp = sin(1.57079633-thetal-theta2)*distanceh;
    dispp = sin(1.57079633-thetalp-theta2p)*distancev;
    float screen_pcntw = (screen_depth * disp) / (WIDTH_M * depth);
// = size_m / ((screen_depth*disp)/depth)
    float screen_pcnth = (screen_depth * dispp) / (HEIGHT_M * depth);
// = size_m / ((screen_depth*disp)/depth)
    coords[0] = (WIDTH_PX/2) + (int)(screen_pcntw*WIDTH_PX); //turns
the percentage into actual pixels
    coords[1] = (HEIGHT_PX/2) - (int)(screen_pcnth*HEIGHT_PX);
//turns the percentage into actual pixels
    //coords[0] = (int)(screen_pcntw*WIDTH_PX); //turns the
percentage into actual pixels

```

```

        //coords[1] = (int)(screen_pcnth*HEIGHT_PX); //turns the
percentage into actual pixels
        printf("%d, %d", coords[0], coords[1]);
        //printf("%f, %f", disp, depth);
    }

void intCoords() {
    //check to see if it will fit in the draw region, may have to
fiddle with precision and casting
    //if it will not fit return a number which will otherwise never
be returned (will use this on pixel drawing one)
/*    if((depth/distancev) < 0.819152 || (depth/distanceh) < 0.9063077)
{ //if cos(theta) > cos(35)? (is the angle > 35 degrees?) //may have to
account for sign
        return WIDTH_PX+HEIGHT_PX;
    } */
    //replace with more accurate function later
    coords[0] = coords[0] + (int)((70.0/(thetal1-thetal))*WIDTH_PX);
    coords[1] = coords[1] + (int)((50.0/(thetal1p-
thetalp))*HEIGHT_PX);
}
//-----
-----

//Main-----
-----

//#ifndef CONSOLE
//    int WINAPI WinMain(HINSTANCE hinstance, HINSTANCE hprevinstance,
LPSTR lpcmdline, int ncmdshow)
//#else
    int main(char **argv, int argc) {
//#endif

        //Change: 640, 480 to change window size, clear color is ARGB
format
        DX_START("DirectX 9 Example", 0, 0, WIDTH_PX, HEIGHT_PX,
0xFF808080);
        device.x = 0;
        device.y = 0;
        device.z = 0;

        object.x = 10;
        object.y = 10.0; //depth
        object.z = 10.0;

        //set device.x, device.y, device.z from the gps
        //set object.x, object.y, object.z as the object (from somewhere)
        //grab thetal and thetalp from the compass
        floatCoords(); //sets up everything

        //printf("%f\n", coords[0]);

```

```
DX_MAIN(render);
#ifdef CONSOLE
    return DX_FINISH();
#endif
}
//-----
```

CONFIDENTIAL

## **SIMULATOR**

### **MAKE FILE**

```
CC = g++
CFLAGS = -Wall
PROG = project

SRCS = project.cpp matrix.cpp hud.cpp

ifeq ($(shell uname), Darwin)
    LIBS = -framework OpenGL -framework GLUT -lsvl
else
    LIBS = -lglut -lGLU -lsvl
endif

all: $(PROG)

$(PROG): $(SRCS)
    $(CC) $(CFLAGS) -o $(PROG) $(SRCS) $(LIBS)

clean:
    rm -f $(PROG)
```

## MOVEMENT STRUCTURES

```
typedef struct
{
    // Line of sight vectors
    float lx;
    float ly;
    float lz;

    // Changes in angle and position
    float deltaAngle;
    float deltaMove;
} CHANGES;

typedef struct
{
    int frame;
    int time;
    int timebase;
    char s[30];
} TIME;
```

## OPEN GL STRUCTURES

```
typedef struct
{
    float angle;
    float ratio;
} CAMERA;

typedef struct
{
    int height;
    int width;
    int border;
    int main;
    int hud;
    int overhead;
} WINDOW;

typedef struct
{
    int frame;
    int time;
    int timebase;
    char s[30];
} TIMING;
```

## PROJECT

```
/*
 * HUD Demo
 *
 * An opengl demo of the hud capabilities
 *
 */

#include <math.h>
#include <GL/glut.h>
#include <GL/gl.h>
#include <GL/glu.h>

#include "hud.h"
#include "matrix.h"

// Structures
#include "movement_structures.h"
#include "hud_structures.h"

// Screen size definitions
#define WIDTH_PX 320 // In Pixels
#define HEIGHT_PX 240
#define WIDTH_M 3.2 // In meters
#define HEIGHT_M 2.4

// Difference on Z axis between the gps location and the center of
screen
#define Z_CHANGE 0

// Radians and degrees
#define RAD2DEG 180/M_PI

/*
 * Global Variables
 */

// Camera
float angle=0.0, ratio, _angle = 30.0f;

// Windows
int border=6;
int h= HEIGHT_PX*2 + 2*border;
int w= WIDTH_PX + 2*border;
int mainWindow, hudWindow, overheadWindow;

// Rendering
static GLint person_display_list;

// Timing
```

```

TIME t;

// HUD
GPSOBJ device;
SCREEN screen;
COMPASS accelerometer;
GPSOBJ ally;
DRAW_COORDS coords;
CHANGES change;

/*****
 * HUD functions
 *****/

void drawSquare(DRAW_COORDS co)
{
    glMatrixMode(GL_MODELVIEW); //Switch to the drawing perspective
    glLoadIdentity(); //Reset the drawing perspective
    glTranslatef(0.0f, 0.0f, -5.0f); //Move forward 5 units
    //co = convert_coords(co);

    glPushMatrix(); //Save the transformations performed thus far
    glTranslatef(co.x, co.y, 0.0f);

    glBegin(GL_QUADS);
    glColor3f(0.0f, 1.0f, 1.0f);

    // Square
    glVertex3f(-0.2f, -0.2f, 0.0f);
    glVertex3f(0.2f, -0.2f, 0.0f);
    glVertex3f(0.2f, 0.2f, 0.0f);
    glVertex3f(-0.2f, 0.2f, 0.0f);

    glEnd();

    glPopMatrix(); //Undo the move to the center of the trapezoid
    glutSwapBuffers();
}

/*
 * Uses compass and gps information to place markers on ally targets
 * Use the width and height of the first window as basis for hud
display
 * Places circles on the screen using the coordinates determined
 */
void placeMarkers()
{
    // Do the calculations
    DRAW_COORDS coords = hudFloatCoords(device, ally, accelerometer,
screen);
    //intCoords();
}

```



```

        glColor3f(1, 0, 0);

        // Draw on screen using coordinates determined
        drawSquare(coords);
    }

    /*****
    * Window - specific functions
    *****/

void initWindow();

void changeSize2(int w1, int h1)
{
    // Prevent a divide by zero, when window is too short
    // (you cant make a window of zero width).
    ratio = 1.0f * w1 / h1;
    // Reset the coordinate system before modifying
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();

    // Set the viewport to be the entire window
    glViewport(0, 0, w1, h1);

    // Set the clipping volume
    gluPerspective(45,ratio,0.1,1000);
    glMatrixMode(GL_MODELVIEW);
}

void changeSize(int w1,int h1) {
    if(h1 == 0)
        h1 = 1;

    w = w1;
    h = h1;

    glutSetWindow(hudWindow);
    glutPositionWindow(border,border);
    glutReshapeWindow(WIDTH_PX, HEIGHT_PX);
    changeSize2(WIDTH_PX, HEIGHT_PX);

    glutSetWindow(overheadWindow);
    glutPositionWindow(border,(h+border)/2);
    glutReshapeWindow(w/2-border*3/2, h/2 - border*3/2);
    changeSize2(w/2-border*3/2,h/2 - border*3/2);
}

    /*****
    * Character-rending functions
    *****/

void drawPerson()

```

```

{
    glColor3f(0.0f, 0.5f, 0.5f);

    // Draw Body
    glTranslatef(0.0f ,0.75f, 0.0f);
    glutSolidSphere(0.75f,20,20);
}

GLuint createDL()
{
    GLuint personDL;

    // Create the id for the list
    personDL = glGenLists(2);

    glNewList(personDL+1, GL_COMPILE);
        drawPerson();
    glEndList();
    // start list
    glNewList(personDL, GL_COMPILE);

    // call the function that contains the rendering commands
    // ALLY
    glColor3f(0.0f, 0.0f, 1.0f);
    // Define GPS coordinates for made object
    ally.x = 0*10.0;
    ally.z = 0*10.0;
    ally.y = 1.75;

    glPushMatrix();
    glTranslatef(0,0,0);
    glCallList(personDL+1);
    glPopMatrix();

    // endList
    glEndList();

    return(personDL);
}

/*****
* Perspective - specific functions
*****/

void setOrthographicProjection() {
    // switch to projection mode
    glMatrixMode(GL_PROJECTION);
    // save previous matrix which contains the
    // settings for the perspective projection
    glPushMatrix();
    // reset matrix
    glLoadIdentity();

```

```

    // set a 2D orthographic projection
    gluOrtho2D(0, w, 0, h/2);
    // invert the y axis, down is positive
    glScalef(1, -1, 1);
    // mover the origin from the bottom left corner
    // to the upper left corner
    glTranslatef(0, -h/2, 0);
    glMatrixMode(GL_MODELVIEW);
}

void resetPerspectiveProjection() {
    // set the current matrix to GL_PROJECTION
    glMatrixMode(GL_PROJECTION);
    // restore previous settings
    glPopMatrix();
    // get back to GL_MODELVIEW matrix
    glMatrixMode(GL_MODELVIEW);
}

/*****
 * Scene - rendering functions
 *****/

void initScene() {
    glEnable(GL_DEPTH_TEST);
    glEnable(GL_CULL_FACE);

    person_display_list = createDL();
}

// Fixes orientation and sends data to compass
void orientMe(float ang) {
    change.lx = sin(ang);
    change.lz = -cos(ang);

    // Update Compass
    accelerometer.pitch = atan(change.ly/change.lz) * RAD2DEG;
    accelerometer.heading = atan((0-change.lx)/change.lz) * RAD2DEG;
    accelerometer.roll = atan(change.ly/change.lx) * RAD2DEG;

    // Adjusts to account for tangent
    if(accelerometer.heading < 0)
        accelerometer.heading = accelerometer.heading + 180;
    if(change.lx < 0)
        accelerometer.heading = 180+accelerometer.heading;
}

// Changes place and updates gps device
void moveMeFlat(int i) {
    device.x = device.x + i*(change.lx)*0.1;
    device.z = device.z + i*(change.lz)*0.1;
}

```

```

void renderScene2(int currentWindow) {
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    glPushMatrix();
        glColor3f(1.0,0.0,0.0);
        glTranslatef(device.x,device.y,device.z);
        glRotatef(180-angle*180.0/3.14,0.0,1.0,0.0);
        glutSolidCone(0.2,0.8f,4,4);
    glPopMatrix();

    // Draw ground
    glColor3f(0.0f, 0.8f, 0.1f);
    glBegin(GL_QUADS);
        glVertex3f(-100.0f, 0.0f, -100.0f);
        glVertex3f(-100.0f, 0.0f, 100.0f);
        glVertex3f( 100.0f, 0.0f, 100.0f);
        glVertex3f( 100.0f, 0.0f, -100.0f);
    glEnd();

    // Populate terrain if first subwindow
    glCallList(person_display_list);
    if (currentWindow == hudWindow)
    {
        /*
            frame++;
            time=glutGet(GLUT_ELAPSED_TIME);
            if (time - timebase > 1000) {
                timebase = time;
                frame = 0;
            }
        */

        glColor3f(0.0,1.0,1.0);
        setOrthographicProjection();
        glPushMatrix();
        glLoadIdentity();

        // Places Markers on friendlies
        placeMarkers();

        glPopMatrix();
        resetPerspectiveProjection();
    }

    glutSwapBuffers();
}

void renderScene() {
    glutSetWindow(mainWindow);
    glClear(GL_COLOR_BUFFER_BIT);
    glutSwapBuffers();
}

```

```

void renderHudScenesw() {
    glutSetWindow(hudWindow);
    glLoadIdentity();
    gluLookAt(device.x, device.y, device.z,
              device.x + change.lx, device.y + change.ly, device.z +
change.lz,
              0.0f,1.0f,0.0f);

    renderScene2(hudWindow);
    placeMarkers();
}

void renderOverheadScenesw() {
    glutSetWindow(overheadWindow);
    glLoadIdentity();
    gluLookAt(device.x, device.y+30, device.z,
              device.x, device.y - 1, device.z,
              change.lx, 0, change.lz);
    renderScene2(overheadWindow);
}

void renderSceneAll() {
    if (change.deltaMove)
        moveMeFlat(change.deltaMove);
    if (change.deltaAngle) {
        angle += change.deltaAngle;
        orientMe(angle);
    }
    renderHudScenesw();
    renderOverheadScenesw();

    _angle += 2.0f;
    if (_angle > 360) {
        _angle -= 360;
    }
}

/*****
 * Keyboard Handlers
 *****/

void processNormalKeys(unsigned char key, int x, int y) {
    if (key == 27)
        exit(0);
    switch(key)
    {
    case 'A': case 'a':
        device.x-=0.5;
        break;
    case 'D': case 'd':
        device.x+=0.5;

```

```

        break;
    }
}

void pressKey(int key, int x, int y) {
    switch (key) {
        case GLUT_KEY_LEFT :
            change.deltaAngle = -0.01f;
            break;
        case GLUT_KEY_RIGHT :
            change.deltaAngle = 0.01f;
            break;
        case GLUT_KEY_UP :
            change.deltaMove = 1;
            break;
        case GLUT_KEY_DOWN :
            change.deltaMove = -1;
            break;
    }
}

void releaseKey(int key, int x, int y) {
    switch (key) {
        case GLUT_KEY_LEFT :
            if (change.deltaAngle < 0.0f)
                change.deltaAngle = 0.0f;
            break;
        case GLUT_KEY_RIGHT :
            if (change.deltaAngle > 0.0f)
                change.deltaAngle = 0.0f;
            break;
        case GLUT_KEY_UP :
            if (change.deltaMove > 0)
                change.deltaMove = 0;
            break;
        case GLUT_KEY_DOWN :
            if (change.deltaMove < 0)
                change.deltaMove = 0;
            break;
    }
}

/*****
 * Main Function
 *****/

void setup()
{
    t.timebase = 0;
    // Initial GPS settings
    device.x=0.0;
    device.z=5.0;
}

```

```

device.y = 1.75; // y currently never changes

// Initial displacement variables
change.lx=0.0f;
change.ly=0.0f;
change.lz=-1.0f;
change.deltaMove = 0;
change.deltaAngle = 0.0;

// Initial Compass Settings
accelerometer.pitch = atan(change.ly/change.lz) * RAD2DEG;
accelerometer.heading = atan(change.lx/change.lz) * RAD2DEG;
accelerometer.roll = atan(change.ly/change.lx) * RAD2DEG;

// Call Initial Calibration
screen = hudScreenCalibrationX(1, screen);
screen = hudScreenCalibrationY(2, screen);
screen = hudScreenCalibrationZ(1, screen);
}

int main(int argc, char **argv)
{
    // Setup
    setup(); // sets up the values for rendering and calculations

    // Initialize GLUT
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGBA);

    // Initialize Main Window
    glutInitWindowPosition(100,100);
    glutInitWindowSize(w,h);
    mainWindow = glutCreateWindow("HUD demo");
    glutReshapeFunc(changeSize);
    orientMe(angle);
    moveMeFlat(0);

    // Define keyboard events
    glutIgnoreKeyRepeat(1);
    glutKeyboardFunc(processNormalKeys);
    glutSpecialFunc(pressKey);
    glutSpecialUpFunc(releaseKey);

    // Set render and idle functions
    glutDisplayFunc(renderScene);
    glutIdleFunc(renderSceneAll);

    // Create HUD window
    hudWindow = glutCreateSubWindow(mainWindow, border, border,
                                     w-2*border, h/2 -
border*3/2);
    glutDisplayFunc(renderScene);

```

```
    initScene();

    // Create Overhead-view window
    overheadWindow = glutCreateSubWindow(mainWindow,
                                         border,
                                         (h+border)/2,
                                         w/2-border*3/2,
                                         h/2 - border*3/2);
    glutDisplayFunc(renderOverheadScenesw);
    initScene();

    glutMainLoop();

    return(0);
}
```

CONFIDENTIAL



## GPS

```
TinyGPS gpsLib;
NewSoftSerial gpsSerial(3, 2);

struct
{
    float longitude;
    float latitude;
    float altitude;
    int valid;
    unsigned long timer;
} gps;

void setupGPS()
{
    gps.valid = 0;
    gps.timer = millis();
    gpsSerial.begin(4800);
}

int pollGPS()
{
    while (gpsSerial.available())
    {
        if (gpsLib.encode(gpsSerial.read()))
        {
            return true;
        }
    }
    return false;
}

void readGPS()
{
    unsigned long age;

    if (millis() - gps.timer >= 1000)
    {
        if (pollGPS())
        {
            if (!gps.valid)
            {
                gps.valid = 1;
            }

            gpsLib.f_get_position(&gps.latitude, &gps.longitude, &age);
            gps.altitude = gpsLib.f_altitude();
        }

        gps.timer = millis();
    }
}
```

}  
}

CONFIDENTIAL

## WIRELESS

```
#include "WProgram.h"

char wirelessBuffer[256];

void setupWireless()
{
    Serial.begin(9600);
}

void sendWireless(char *msg)
{
    Serial.print('$');

    Serial.print(msg);

    Serial.print('%');
}

char *recvWireless()
{
    int i = 0, enable;
    char c;

    enable = 0;

    if (!Serial.available())
    {
        return NULL;
    }

    while (Serial.available())
    {
        c = Serial.read();

        if (enable && c == '%')
        {
            wirelessBuffer[i] = '\0';

            return wirelessBuffer;
        }

        if (i < 254)
        {
            if (enable)
            {
                wirelessBuffer[i++] = c;
            }

            if (c == '$')

```

```
        {
            enable = 1;
        }
    }
else
{
    return NULL;
}
}
```

CONFIDENTIAL

## ACCELEROMETER

```
/*
   HMC6343 uses 3.3v VCC
   Hardcoded value for HMC6343Address assumes:
       SCL connected to Arduino analog in pin 5
       SDA connected to Arduino analog in pin 4
*/

#include "WProgram.h"

struct
{
    float heading;
    float pitch;
    float roll;
    int counter;
    float headingBuffer[4];
    int HMC6343Address;
} accelerometer;

void setupAccelerometer()
{
    //Analog i/o pin address
    accelerometer.HMC6343Address = 0x32 >> 1;

    //Initialize the Wire library
    Wire.begin();

    //Startup time of HMC6343 microcontroller is ~500ms
    delay(500);
}

void readAccelerometer()
{
    int i, h;
    char headingData[6];
    i = 0;

    //Transmit 'Get Data' request
    Wire.beginTransmission(accelerometer.HMC6343Address);
    Wire.send(0x50);
    Wire.endTransmission();

    //Delay for response processing - this could be as low as 2ms
    delay(10);

    //Retrieve 6-byte 3-axis accelerometer data
    Wire.requestFrom(accelerometer.HMC6343Address, 6);

    //Flush data to buffer
```

```

while (Wire.available() && i < 6)
{
    headingData[i] = Wire.receive();
    i++;
}

//Place data into buffer and calculate mean
h = (float)(headingData[0] * 256 + headingData[1]) / 10;

if (accelerometer.headingBuffer[0] == h)
{
    accelerometer.headingBuffer[0] =
accelerometer.headingBuffer[1] = accelerometer.headingBuffer[2] =
accelerometer.headingBuffer[3] = h;
    accelerometer.heading = h;
}
else
{
    accelerometer.headingBuffer[3] =
accelerometer.headingBuffer[2];
    accelerometer.headingBuffer[2] =
accelerometer.headingBuffer[1];
    accelerometer.headingBuffer[1] =
accelerometer.headingBuffer[0];
    accelerometer.headingBuffer[0] = h;

    if (accelerometer.counter++ % 4 == 0)
    {
        accelerometer.heading =
(accelerometer.headingBuffer[0] + accelerometer.headingBuffer[1] +
accelerometer.headingBuffer[2] + accelerometer.headingBuffer[3]) / 4;
    }
}

//Place data into struct
accelerometer.pitch = (float)(headingData[2] * 256 +
headingData[3]) / 10;
accelerometer.roll = (float)(headingData[4] * 256 +
headingData[5]) / 10;
}

```

## **Appendix 6: Projected Income Statements**

CONFIDENTIAL

## Smart Specs LLC

### Yearly Projected Cash Flow

	Running Totals	2/1/2012	3/1/2012	4/1/2012	5/1/2012	6/1/2012	7/1/2012	8/1/2012	9/1/2012	10/1/2012	11/1/2012	12/1/2012	1/1/2013	2/1/2013	13 Month Running Totals
<b>Beginning Cash Balance</b>		\$300,000.00	\$346,650.00	\$334,800.00	\$314,780.00	\$319,760.00	\$324,740.00	\$329,720.00	\$342,870.00	\$361,020.00	\$379,170.00	\$397,320.00	\$415,470.00	\$433,620.00	
<b>Cash Inflows</b>															
3F Invoice (50 Units)	\$ -	\$ -	\$ -	\$ 37,500.00	\$ 37,500.00	\$ 37,500.00	\$ 37,500.00	\$ 75,000.00	\$ 75,000.00	\$ 75,000.00	\$ 75,000.00	\$ 75,000.00	\$ 75,000.00	\$ 112,500.00	\$ 712,500.00
Lease Invoice	\$ -	\$ -	\$ -	\$ 10,000.00	\$ 10,000.00	\$ 10,000.00	\$ 10,000.00	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ 40,000.00
															<b>Total Invoiced</b>
															<b>\$ 752,500.00</b>
Cash Received 3F Product (50 Units)	\$ -	\$ -	\$ -	\$ 15,000.00	\$ 15,000.00	\$ 15,000.00	\$ 15,000.00	\$ 15,000.00	\$ 30,000.00	\$ 30,000.00	\$ 30,000.00	\$ 30,000.00	\$ 30,000.00	\$ 30,000.00	\$ 195,000.00
Cash Received Paintball Fields	\$ -	\$ -	\$ -	\$ 10,000.00	\$ 10,000.00	\$ 10,000.00	\$ 10,000.00	\$ 10,000.00	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ 40,000.00
Investment	\$ 300,000.00	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ 300,000.00
<b>Total Cash Inflows</b>	\$ -	\$ 300,000.00	\$ -	\$ 25,000.00	\$ 25,000.00	\$ 25,000.00	\$ 25,000.00	\$ 25,000.00	\$ 30,000.00	\$ 30,000.00	\$ 30,000.00	\$ 30,000.00	\$ 30,000.00	\$ 30,000.00	<b>\$ 580,000.00</b>
															<b>Total Received</b>
															<b>\$ 535,000.00</b>
<b>Cash Outflows</b>															
Corporate	\$ 35,150.00	\$ 9,850.00	\$ 9,850.00	\$ 9,850.00	\$ 9,850.00	\$ 9,850.00	\$ 9,850.00	\$ 9,850.00	\$ 9,850.00	\$ 9,850.00	\$ 9,850.00	\$ 9,850.00	\$ 9,850.00	\$ 9,850.00	\$ 163,200.00
Product Development	\$ -	\$ 72,500.00	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ 72,500.00
Research and Development	\$ 4,905.00	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ 4,905.00
Capital Purchases	\$ 35,750.00	\$ 2,000.00	\$ 2,000.00	\$ 2,000.00	\$ 2,000.00	\$ 2,000.00	\$ 2,000.00	\$ 2,000.00	\$ 2,000.00	\$ 2,000.00	\$ 2,000.00	\$ 2,000.00	\$ 2,000.00	\$ 2,000.00	\$ 61,750.00
Marketing Expenses	\$ -	\$ -	\$ -	\$ 8,170.00	\$ 8,170.00	\$ 8,170.00	\$ 8,170.00	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ 32,680.00
<b>Total Cash Outflows</b>	\$ 75,805.00	\$ 84,350.00	\$ 11,850.00	\$ 20,020.00	\$ 20,020.00	\$ 20,020.00	\$ 20,020.00	\$ 11,850.00	\$ 11,850.00	\$ 11,850.00	\$ 11,850.00	\$ 11,850.00	\$ 11,850.00	\$ 11,850.00	<b>\$ 335,035.00</b>
<b>Net Increase (Decrease) in Cash</b>		\$215,650.00	(\$11,850.00)	(\$20,020.00)	\$4,980.00	\$4,980.00	\$4,980.00	\$13,150.00	\$18,150.00	\$18,150.00	\$18,150.00	\$18,150.00	\$18,150.00	\$18,150.00	\$18,150.00
<b>Ending Cash Balance</b>	\$0.00	\$300,000.00	\$334,800.00	\$314,780.00	\$319,760.00	\$324,740.00	\$329,720.00	\$342,870.00	\$361,020.00	\$379,170.00	\$397,320.00	\$415,470.00	\$433,620.00	\$451,770.00	



## Smart Specs LLC

### Yearly Projected Cash Flow

	Running Totals	2/1/2013	3/1/2013	4/1/2013	5/1/2013	6/1/2013	7/1/2013	8/1/2013	9/1/2013	10/1/2013	11/1/2013	12/1/2013	1/1/2014	2/1/2014	13 Month Running Totals	
<b>Beginning Cash Balance</b>	\$545,780.00	\$500,780.00	\$520,930.00	\$556,080.00	\$606,230.00	\$669,380.00	\$732,530.00	\$795,680.00	\$858,830.00	\$921,980.00	\$970,130.00	\$1,003,280.00	\$1,036,430.00	\$1,069,580.00		
<b>Cash Inflows</b>																
3F Invoice		\$ 112,500.00	\$ 150,000.00	\$ 187,500.00	\$ 187,500.00	\$ 187,500.00	\$ 187,500.00	\$ 187,500.00	\$ 150,000.00	\$ 112,500.00	\$ 112,500.00	\$ 112,500.00	\$ 112,500.00	\$ 150,000.00	\$ 1,950,000.00	
Lease Invoice	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	
															<b>Total Invoiced</b>	
																<b>\$1,950,000.00</b>
Cash Received 3F Product		\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	
Cash Received Paintball Fields		\$ 30,000.00	\$ 45,000.00	\$ 60,000.00	\$ 75,000.00	\$ 75,000.00	\$ 75,000.00	\$ 75,000.00	\$ 75,000.00	\$ 60,000.00	\$ 45,000.00	\$ 45,000.00	\$ 45,000.00	\$ 45,000.00	\$ 750,000.00	
Investment		\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	
<b>Total Cash Inflows</b>	\$ -	\$ 30,000.00	\$ 45,000.00	\$ 60,000.00	\$ 75,000.00	\$ 75,000.00	\$ 75,000.00	\$ 75,000.00	\$ 75,000.00	\$ 60,000.00	\$ 45,000.00	\$ 45,000.00	\$ 45,000.00	\$ 45,000.00	\$ 750,000.00	
															<b>Total Received</b>	
																<b>\$ 750,000.00</b>
<b>Cash Outflows</b>																
Corporate	\$ -	\$ 9,850.00	\$ 9,850.00	\$ 9,850.00	\$ 9,850.00	\$ 9,850.00	\$ 9,850.00	\$ 9,850.00	\$ 9,850.00	\$ 9,850.00	\$ 9,850.00	\$ 9,850.00	\$ 9,850.00	\$ 9,850.00	\$ 128,050.00	
Product Development	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	
Research and Development	\$ 20,000.00	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ 20,000.00	
Capital Purchases	\$ -	\$ -	\$ 2,000.00	\$ 2,000.00	\$ 2,000.00	\$ 2,000.00	\$ 2,000.00	\$ 2,000.00	\$ 2,000.00	\$ 2,000.00	\$ 2,000.00	\$ 2,000.00	\$ 2,000.00	\$ 2,000.00	\$ 24,000.00	
Marketing Expenses	\$ 25,000.00	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ 25,000.00	
<b>Total Cash Outflows</b>	\$ 45,000.00	\$ 9,850.00	\$ 9,850.00	\$ 9,850.00	\$ 11,850.00	\$ 11,850.00	\$ 11,850.00	\$ 11,850.00	\$ 11,850.00	\$ 11,850.00	\$ 11,850.00	\$ 11,850.00	\$ 11,850.00	\$ 11,850.00	\$ 197,050.00	
<b>Net Increase (Decrease) in Cash</b>		\$ 20,150.00	\$ 35,150.00	\$ 50,150.00	\$ 63,150.00	\$ 63,150.00	\$ 63,150.00	\$ 63,150.00	\$ 63,150.00	\$ 48,150.00	\$ 33,150.00	\$ 33,150.00	\$ 33,150.00	\$ 33,150.00	\$ 33,150.00	
<b>Ending Cash Balance</b>	\$ 500,780.00	\$ 520,930.00	\$ 556,080.00	\$ 606,230.00	\$ 669,380.00	\$ 732,530.00	\$ 795,680.00	\$ 858,830.00	\$ 921,980.00	\$ 970,130.00	\$ 1,003,280.00	\$ 1,036,430.00	\$ 1,069,580.00	\$ 1,102,730.00		

## Smart Specs LLC

### Yearly Projected Cash Flow

	Running Totals	2/1/2014	3/1/2014	4/1/2014	5/1/2014	6/1/2014	7/1/2014	8/1/2014	9/1/2014	10/1/2014	11/1/2014	12/1/2014	1/1/2015	2/1/2015	13 Month Running Totals
<b>Beginning Cash Balance</b>	\$1,200,580.00	\$1,155,580.00	\$1,190,730.00	\$1,234,880.00	\$1,279,030.00	\$1,336,180.00	\$1,393,330.00	\$1,455,480.00	\$1,517,630.00	\$1,579,780.00	\$1,636,930.00	\$1,679,080.00	\$1,721,230.00	\$1,748,380.00	
<b>Cash Inflows</b>															
3F Invoice		\$ 150,000.00	\$ 150,000.00	\$ 187,500.00	\$ 187,500.00	\$ 200,000.00	\$ 200,000.00	\$ 200,000.00	\$ 187,500.00	\$ 150,000.00	\$ 150,000.00	\$ 112,500.00	\$ 112,500.00	\$ 150,000.00	\$ 2,137,500.00
Lease Invoice	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -
															<b>Total Invoiced</b>
															<b>\$2,137,500.00</b>
Cash Received 3F Product		\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -
Cash Received Paintball Fields		\$ 45,000.00	\$ 60,000.00	\$ 60,000.00	\$ 75,000.00	\$ 75,000.00	\$ 80,000.00	\$ 80,000.00	\$ 80,000.00	\$ 75,000.00	\$ 60,000.00	\$ 60,000.00	\$ 45,000.00	\$ 45,000.00	\$ 840,000.00
Investment		\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -
<b>Total Cash Inflows</b>	\$ -	\$ 45,000.00	\$ 60,000.00	\$ 60,000.00	\$ 75,000.00	\$ 75,000.00	\$ 80,000.00	\$ 80,000.00	\$ 80,000.00	\$ 75,000.00	\$ 60,000.00	\$ 60,000.00	\$ 45,000.00	\$ 45,000.00	<b>\$ 840,000.00</b>
															<b>Total Received</b>
															<b>\$ 840,000.00</b>
<b>Cash Outflows</b>															
Corporate	\$ -	\$ 9,850.00	\$ 15,850.00	\$ 15,850.00	\$ 15,850.00	\$ 15,850.00	\$ 15,850.00	\$ 15,850.00	\$ 15,850.00	\$ 15,850.00	\$ 15,850.00	\$ 15,850.00	\$ 15,850.00	\$ 15,850.00	\$ 200,050.00
Product Development	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -
Research and Development	\$ 20,000.00	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ 20,000.00
Capital Purchases	\$ -	\$ -	\$ 2,000.00	\$ 2,000.00	\$ 2,000.00	\$ 2,000.00	\$ 2,000.00	\$ 2,000.00	\$ 2,000.00	\$ 2,000.00	\$ 2,000.00	\$ 2,000.00	\$ 2,000.00	\$ 2,000.00	\$ 24,000.00
Marketing Expenses	\$ 25,000.00	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ 25,000.00
<b>Total Cash Outflows</b>	\$ 45,000.00	\$ 9,850.00	\$ 15,850.00	\$ 15,850.00	\$ 17,850.00	\$ 17,850.00	\$ 17,850.00	\$ 17,850.00	\$ 17,850.00	\$ 17,850.00	\$ 17,850.00	\$ 17,850.00	\$ 17,850.00	\$ 17,850.00	<b>\$ 269,050.00</b>
<b>Net Increase (Decrease) in Cash</b>		\$35,150.00	\$44,150.00	\$44,150.00	\$57,150.00	\$57,150.00	\$62,150.00	\$62,150.00	\$62,150.00	\$57,150.00	\$42,150.00	\$42,150.00	\$27,150.00	\$27,150.00	
<b>Ending Cash Balance</b>	\$1,155,580.00	\$1,190,730.00	\$1,234,880.00	\$1,279,030.00	\$1,336,180.00	\$1,393,330.00	\$1,455,480.00	\$1,517,630.00	\$1,579,780.00	\$1,636,930.00	\$1,679,080.00	\$1,721,230.00	\$1,748,380.00	\$1,775,530.00	

## Smart Specs LLC

### Yearly Projected Cash Flow

	Running Totals	2/1/2015	3/1/2015	4/1/2015	5/1/2015	6/1/2015	7/1/2015	8/1/2015	9/1/2015	10/1/2015	11/1/2015	12/1/2015	1/1/2016	2/1/2016	13 Month Running Totals
<b>Beginning Cash Balance</b>	\$1,829,380.00	\$1,784,380.00	\$1,813,530.00	\$1,851,680.00	\$1,889,830.00	\$1,940,980.00	\$2,007,130.00	\$2,073,280.00	\$2,154,430.00	\$2,235,580.00	\$2,301,730.00	\$2,352,880.00	\$2,389,030.00	\$2,425,180.00	
<b>Cash Inflows</b>															
3F Invoice		\$ 150,000.00	\$ 150,000.00	\$ 187,500.00	\$ 225,000.00	\$ 225,000.00	\$ 262,500.00	\$ 262,500.00	\$ 225,000.00	\$ 187,500.00	\$ 150,000.00	\$ 150,000.00	\$ 112,500.00	\$ 150,000.00	\$ 2,437,500.00
Lease Invoice	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -
															<b>Total Invoiced</b>
															<b>\$ 2,437,500.00</b>
Cash Received 3F Product		\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -
Cash Received Paintball Fields		\$ 45,000.00	\$ 60,000.00	\$ 60,000.00	\$ 75,000.00	\$ 90,000.00	\$ 90,000.00	\$ 105,000.00	\$ 105,000.00	\$ 90,000.00	\$ 75,000.00	\$ 60,000.00	\$ 60,000.00	\$ 45,000.00	\$ 960,000.00
Investment		\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -
<b>Total Cash Inflows</b>	\$ -	\$ 45,000.00	\$ 60,000.00	\$ 60,000.00	\$ 75,000.00	\$ 90,000.00	\$ 90,000.00	\$ 105,000.00	\$ 105,000.00	\$ 90,000.00	\$ 75,000.00	\$ 60,000.00	\$ 60,000.00	\$ 45,000.00	<b>\$ 960,000.00</b>
															<b>Total Received</b>
															<b>\$ 960,000.00</b>
<b>Cash Outflows</b>															
Corporate	\$ -	\$ 15,850.00	\$ 21,850.00	\$ 21,850.00	\$ 21,850.00	\$ 21,850.00	\$ 21,850.00	\$ 21,850.00	\$ 21,850.00	\$ 21,850.00	\$ 21,850.00	\$ 21,850.00	\$ 21,850.00	\$ 21,850.00	\$ 278,050.00
Product Development	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -
Research and Development	\$ 20,000.00	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ 20,000.00
Capital Purchases	\$ -	\$ -	\$ 2,000.00	\$ 2,000.00	\$ 2,000.00	\$ 2,000.00	\$ 2,000.00	\$ 2,000.00	\$ 2,000.00	\$ 2,000.00	\$ 2,000.00	\$ 2,000.00	\$ 2,000.00	\$ 2,000.00	\$ 24,000.00
Marketing Expenses	\$ 25,000.00	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ 25,000.00
<b>Total Cash Outflows</b>	\$ 45,000.00	\$ 15,850.00	\$ 21,850.00	\$ 21,850.00	\$ 23,850.00	\$ 23,850.00	\$ 23,850.00	\$ 23,850.00	\$ 23,850.00	\$ 23,850.00	\$ 23,850.00	\$ 23,850.00	\$ 23,850.00	\$ 23,850.00	<b>\$ 347,050.00</b>
<b>Net Increase (Decrease) in Cash</b>		\$ 29,150.00	\$ 38,150.00	\$ 38,150.00	\$ 51,150.00	\$ 66,150.00	\$ 66,150.00	\$ 81,150.00	\$ 81,150.00	\$ 66,150.00	\$ 51,150.00	\$ 36,150.00	\$ 36,150.00	\$ 21,150.00	
<b>Ending Cash Balance</b>	\$1,784,380.00	\$1,813,530.00	\$1,851,680.00	\$1,889,830.00	\$1,940,980.00	\$2,007,130.00	\$2,073,280.00	\$2,154,430.00	\$2,235,580.00	\$2,301,730.00	\$2,352,880.00	\$2,389,030.00	\$2,425,180.00	\$2,446,330.00	

## Smart Specs LLC

### Yearly Projected Cash Flow

	Running Totals	2/1/2016	3/1/2016	4/1/2016	5/1/2016	6/1/2016	7/1/2016	8/1/2016	9/1/2016	10/1/2016	11/1/2016	12/1/2016	1/1/2017	2/1/2017	13 Month Running Totals
<b>Beginning Cash Balance</b>	\$2,674,180.00	\$2,629,180.00	\$2,652,330.00	\$2,684,480.00	\$2,731,630.00	\$2,791,780.00	\$2,866,930.00	\$2,957,080.00	\$3,047,230.00	\$3,137,380.00	\$3,197,530.00	\$3,242,680.00	\$3,272,830.00	\$3,287,980.00	
<b>Cash Inflows</b>															
3F Invoice	\$ -	\$ 150,000.00	\$ 187,500.00	\$ 225,000.00	\$ 262,500.00	\$ 300,000.00	\$ 300,000.00	\$ 300,000.00	\$ 225,000.00	\$ 187,500.00	\$ 150,000.00	\$ 112,500.00	\$ 112,500.00	\$ -	\$ 2,512,500.00
Lease Invoice	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -
															<b>Total Invoiced</b>
															<b>\$ 2,512,500.00</b>
Cash Received 3F Product	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -
Cash Received Paintball Fields	\$ 45,000.00	\$ 60,000.00	\$ 75,000.00	\$ 90,000.00	\$ 105,000.00	\$ 120,000.00	\$ 120,000.00	\$ 120,000.00	\$ 120,000.00	\$ 90,000.00	\$ 75,000.00	\$ 60,000.00	\$ 45,000.00	\$ 45,000.00	\$ 1,050,000.00
Investment	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -
<b>Total Cash Inflows</b>	\$ -	\$ 45,000.00	\$ 60,000.00	\$ 75,000.00	\$ 90,000.00	\$ 105,000.00	\$ 120,000.00	\$ 120,000.00	\$ 120,000.00	\$ 90,000.00	\$ 75,000.00	\$ 60,000.00	\$ 45,000.00	\$ 45,000.00	<b>Total Received</b>
															<b>\$ 1,050,000.00</b>
<b>Cash Outflows</b>															
Corporate	\$ -	\$ 21,850.00	\$ 27,850.00	\$ 27,850.00	\$ 27,850.00	\$ 27,850.00	\$ 27,850.00	\$ 27,850.00	\$ 27,850.00	\$ 27,850.00	\$ 27,850.00	\$ 27,850.00	\$ 27,850.00	\$ 27,850.00	\$ 356,050.00
Product Development	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -
Research and Development	\$ 20,000.00	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ 10,000.00
Capitol Purchases	\$ -	\$ -	\$ 2,000.00	\$ 2,000.00	\$ 2,000.00	\$ 2,000.00	\$ 2,000.00	\$ 2,000.00	\$ 2,000.00	\$ 2,000.00	\$ 2,000.00	\$ 2,000.00	\$ 2,000.00	\$ 2,000.00	\$ 24,000.00
Marketing Expenses	\$ 25,000.00	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ -	\$ 25,000.00
<b>Total Cash Outflows</b>	\$ 45,000.00	\$ 21,850.00	\$ 27,850.00	\$ 27,850.00	\$ 29,850.00	\$ 29,850.00	\$ 29,850.00	\$ 29,850.00	\$ 29,850.00	\$ 29,850.00	\$ 29,850.00	\$ 29,850.00	\$ 29,850.00	\$ 29,850.00	\$ 415,050.00
<b>Net Increase (Decrease) in Cash</b>		\$ 23,150.00	\$ 32,150.00	\$ 47,150.00	\$ 60,150.00	\$ 75,150.00	\$ 90,150.00	\$ 90,150.00	\$ 90,150.00	\$ 60,150.00	\$ 45,150.00	\$ 30,150.00	\$ 15,150.00	\$ 15,150.00	
<b>Ending Cash Balance</b>	\$2,629,180.00	\$2,652,330.00	\$2,684,480.00	\$2,731,630.00	\$2,791,780.00	\$2,866,930.00	\$2,957,080.00	\$3,047,230.00	\$3,137,380.00	\$3,197,530.00	\$3,242,680.00	\$3,272,830.00	\$3,287,980.00	\$3,303,130.00	

**SMART SPECS LLC**

**EXPENSES**

**Corporate**

Salaries	\$6,000.00	monthly
GA	\$1,700.00	monthly
Utilities	\$150.00	monthly
Insurance	\$1,500.00	monthly
Misc	\$500.00	monthly
	<u>\$9,850.00</u>	

3 employees @ \$24,000 annually (years 1-2)  
 3 employees @ \$48,000 annually (year 3)  
 3 employees @ \$72,000 annually (year 4)  
 3 employees @ \$96,000 annually (year 5)

**Contribution Margin**

Sales per 50 units	\$37,500
50 unit cost	(\$22,500)
CM	\$5,150

**Fleet Salary & Expenses (A.K.A. Marketing)**

Fuel	\$2,000.00	monthly
Maintenance	\$250.00	monthly
Lodging	\$150.00	trip
Per Deim	\$140.00	trip
	<u>\$2,540.00</u>	

Mktg \$25,000 yearly (years 2-5)

**R&D**

\$4,905 for year 1  
 \$20,000 yearly ( years 2-5)

**Capital Purchases**

Van Purchase	\$20,000.00	1 van
Trailer	\$5,000.00	1 trailer
	<u>\$25,000.00</u>	

**After 6 months**

Rent \$2,000 monthly

**Initial Product Development Cost**

MFG Consulting	\$5,000.00	total
3F Product	\$22,500	total
	<u>\$45,000</u>	total
100 units		
	<u>\$72,500.00</u>	

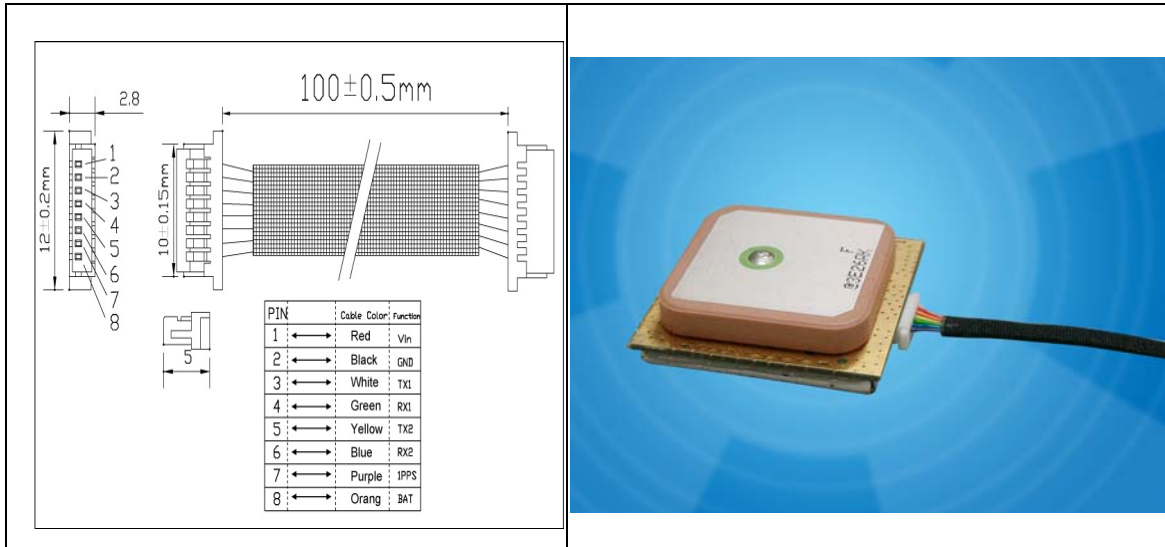
## Appendix 7: Hardware Data Sheets

CONFIDENTIAL



# GPS Engine Board

## Model: FV-M8



©2008 San Jose Technology, Inc. All specifications subject to change without notice.

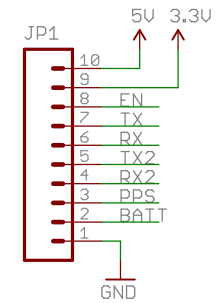
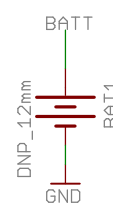
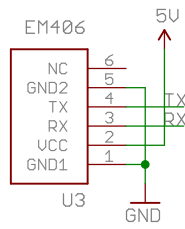
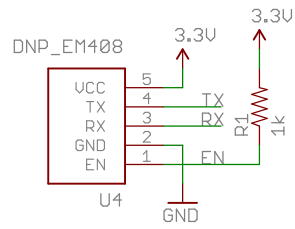
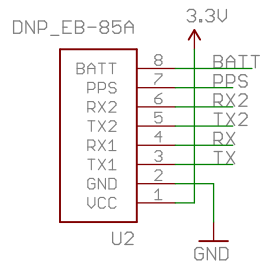
### Specifications:

PHYSICAL CONSTRUCTION		PERFORMANCE		
Dimension	L30mm*W30mm*H8.6mm	Built-in Antenna	Highly-reliable ceramic patch	
Weight	15 grams	Sensitivity	-158dbm	
		SBAS	1 channel (Support WAAS, EGNOS, MSAS)	
		DGPS	RTCM Protocol	
Receiving frequency	1575.42MHZ; C/A code	Receiver architecture	32 parallel channels	
Connector	8pin connector with 1.0mm pitch	Start-up time	Hot start	1 sec. typical
			Warm start	35 sec. typical
			Cold start	41sec. typical
Mounting	Soldering	Position accuracy	Without aid	3.3 m CEP
			DGPS (RTCM)	2.6 m

Construction	Full EMI Shielding		Velocity accuracy	0.1 Knot RMS steady state	
<b>ENVIRONMENTAL CONDITIONS</b>			Update Rate	1 ~ 5Hz	
Temperature	Operating: -30 ~ +80 °C		Power Supply	3.3~5V +- 5%	
	Storage: -40 ~ +85 °C			Current Consumption	Acquisition
<b>COMMUNICATION</b>			Tracking		59mA (first 5 minutes)
					42mA (after 5 minutes)
Protocol	NMEA V3.01		33mA (after 20minutes)		
Signal level	UART @ 2.8V * 2				
<b>INTERFACE CAPABILITY</b>			Baud Rate	4800 bps (default) & 4800/9600/38400/57600/11520 0 bps are adjustable	
Standard Output Sentences	Default	RMC, GGA, GSV*5, VTG, GSA*5			
	Optional	GLL, ZDA			

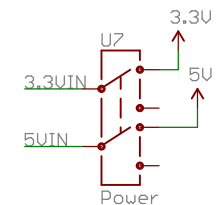
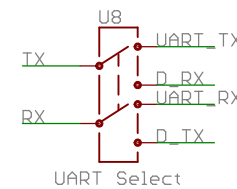
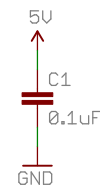
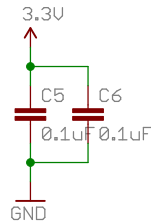
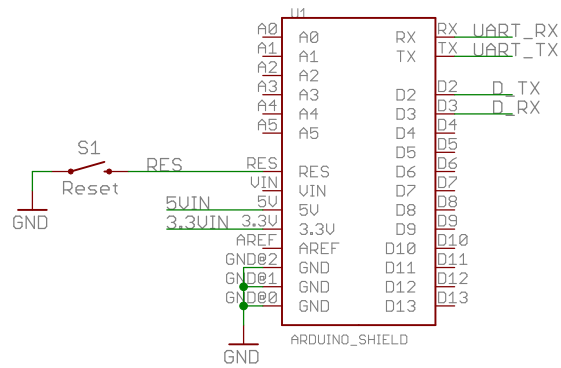
©2008 San Jose Technology, Inc.  
All specifications subject to change without notice.





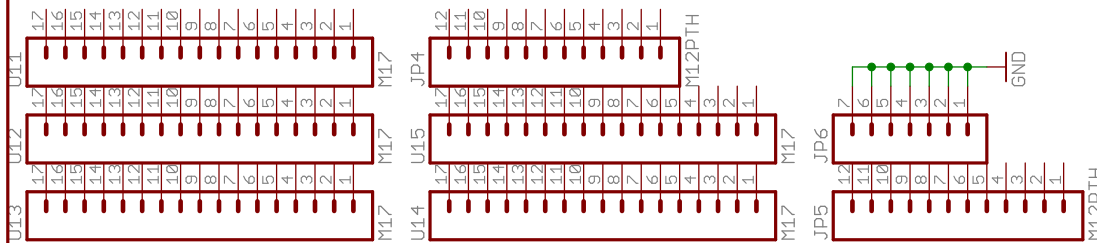
GPS Connectors

Unused GPS Pins



Arduino

Switches



EM406

EM408 GPS

TITLE: GPSShield-v11

SFE

Document Number:

REV:

Date: 10/13/2009 3:00:01 PM

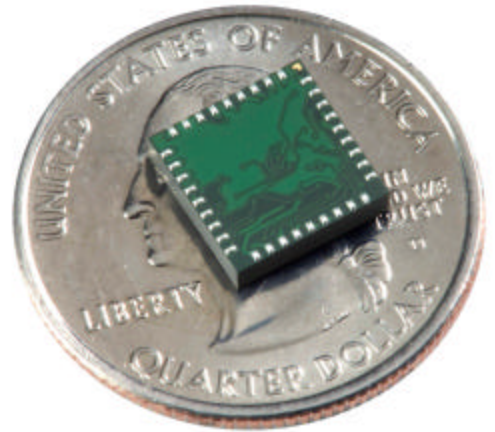
Sheet: 1/1

Prototyping

# 3-Axis Compass with Algorithms HMC6343

---

The Honeywell HMC6343 is a fully integrated compass module that includes firmware for heading computation and calibration for magnetic distortions. The module combines 3-axis magneto-resistive sensors and 3-axis MEMS accelerometers, analog and digital support circuits, microprocessor and algorithms required for heading computation. By combining the sensor elements, processing electronics, and firmware into a 9.0mm by 9.0mm by 1.9mm LCC package, Honeywell offers a complete, ready to use tilt-compensated electronic compass. This provides design engineers with the simplest solution to integrate high volume, cost effective compasses into binoculars, cameras, night vision optics, laser ranger finders, antenna positioning, and other industrial compassing applications.



The HMC6343 utilizes Honeywell's Anisotropic Magneto-resistive (AMR) technology that provides advantages over other magnetic sensor technologies. The sensors feature precision sensitivity and linearity, solid-state construction with very low cross-axis sensitivity designed to measure both direction and magnitude of Earth's magnetic fields. Honeywell's Magnetic Sensors are among the most sensitive and reliable low-field sensors in the industry.

Honeywell continues to maintain product excellence and performance by introducing innovative solid-state magnetic sensor solutions. Honeywell's magnetic sensor solutions provide real solutions you can count on.

## FEATURES

## BENEFITS

- 
- |  |   |
|--|---|
| ▶ Compass with Heading/Tilt Outputs  | ▶ A complete compass solution including compass firmware  |
| ▶ 3-axis MR Sensors, Accelerometers and a Microprocessor in a Single Package | ▶ A digital compass solution with heading and tilt angle outputs in a chip-scale package        |
| ▶ Compass Algorithms   | ▶ For computation of heading, and magnetic calibration for hard-iron                            |
| ▶ 9 x 9 x 1.9mm LCC Surface Mount Package                                    | ▶ Small size, easy to assemble and compatible with high speed surface mount technology assembly |
| ▶ Low Voltage Operations   | ▶ Compatible with battery powered applications  |
| ▶ EEPROM Memory  | ▶ To store compass data for processor routines  |
| ▶ Digital Serial Data Interface  | ▶ I <sup>2</sup> C Interface, easy to use 2-wire communication for heading output               |
| ▶ Moderate Precision Outputs   | ▶ Typical 2° Heading Accuracy with 1° Pitch and Roll Accuracy                                   |
| ▶ Lead Free Package Construction   | ▶ Complies with RoHS environmental standards  |
| ▶ Flexible Mounting  | ▶ Can be mounted on horizontal or vertical circuit boards                                       |

# HMC6343

## SPECIFICATIONS

Characteristics	Conditions*	Min	Typ	Max	Units
-----------------	-------------	-----	-----	-----	-------

### Power Supply

Supply Voltage	VDD Referenced to GND	2.7	3.3	3.6	Volts	
Current	All VDD pins connected together	3.5	4.5	5.5	mA	
	Run Mode (10Hz Output)					
	Standby Mode					1.0
	Sleep mode					10
	Power-up peak (VDD = 3.3V)		8		mA	
Power-on Rate	Minimum rise time for POR	0.05	-	-	V/msec	

### Compass Function

Field Range	total applied magnetic field (de-gauss if exposed to >5gauss)		±1	±2	gauss
Heading Accuracy	At Level, +3.3V	1.0	2.0	3.0	deg RMS
	±15° tilt				
	±60° tilt				
Heading Resolution	Output Data		0.1		degrees
Heading Repeatability	Output Data (1σ)		±0.3		degrees
Heading Hysteresis	Output Data (1σ)		±0.3		degrees
Update Rate	Run Mode (1, 5, 10Hz)	1	5	10	Hz
Tilt Range	From Horizontal		±80		degrees
Tilt Accuracy	0° to ±15°, +3.3V		±1		degrees
	±15° to ±60°		±2		
Tilt Resolution	Output Data		0.1		degrees
Tilt Repeatability	Output Data (1σ)		±0.2		degrees

### Offset Straps

Resistance	Measured from OFF+ to OFF-	5	8	11	ohms
Offset Constant	DC Current Field applied in sensitive direction		10		mA/gauss
Resistance Tempco	T <sub>A</sub> =-40 to 125°C	1800	2700	4500	ppm/°C

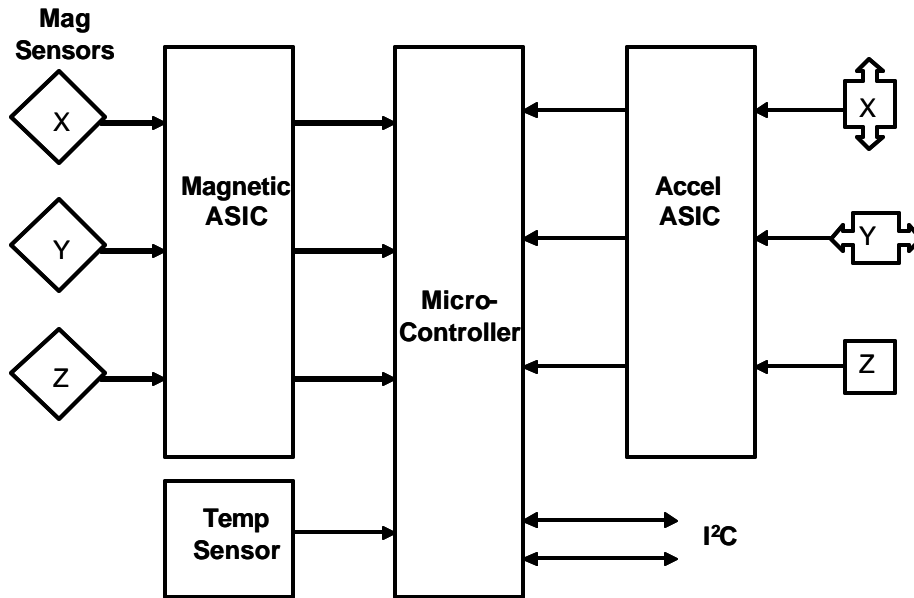
### General

Operating Temperature	Ambient	-40		80	°C
Storage Temperature	Ambient, unbiased	-55		125	°C
Weight			0.32		grams
ESD Voltage				400	V
MSL	Moisture Sensitivity Level	3			-
Solder Temp	Peak Reflow Temp (< 30 seconds)			250	°C

\* Tested at 25°C and 3.3V except stated otherwise.

# HMC6343

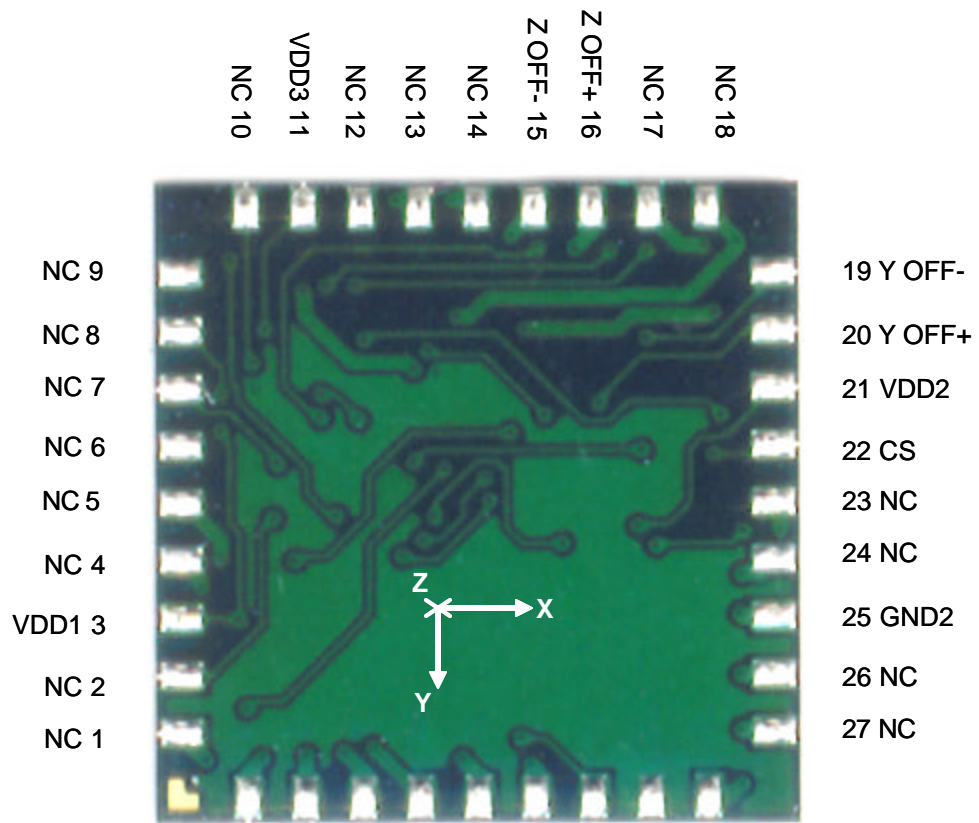
## FUNCTIONAL DIAGRAM



## PIN CONFIGURATIONS

Pin Number	Description	Pin Number	Description
1	NC	19	Y OFF-
2	NC	20	Y OFF+
3	VDD1	21	VDD2
4	NC	22	CS
5	NC	23	X OFF-
6	NC	24	X OFF+
7	NC	25	GND2
8	NC	26	NC
9	NC	27	NC
10	NC	28	NC
11	VDD3	29	GND1
12	NC	30	NC
13	NC	31	NC
14	NC	32	SCK/SCL
15	Z OFF-	33	NC
16	Z OFF+	34	NC
17	NC	35	CS_CTRL
18	NC	36	SDA

# HMC6343



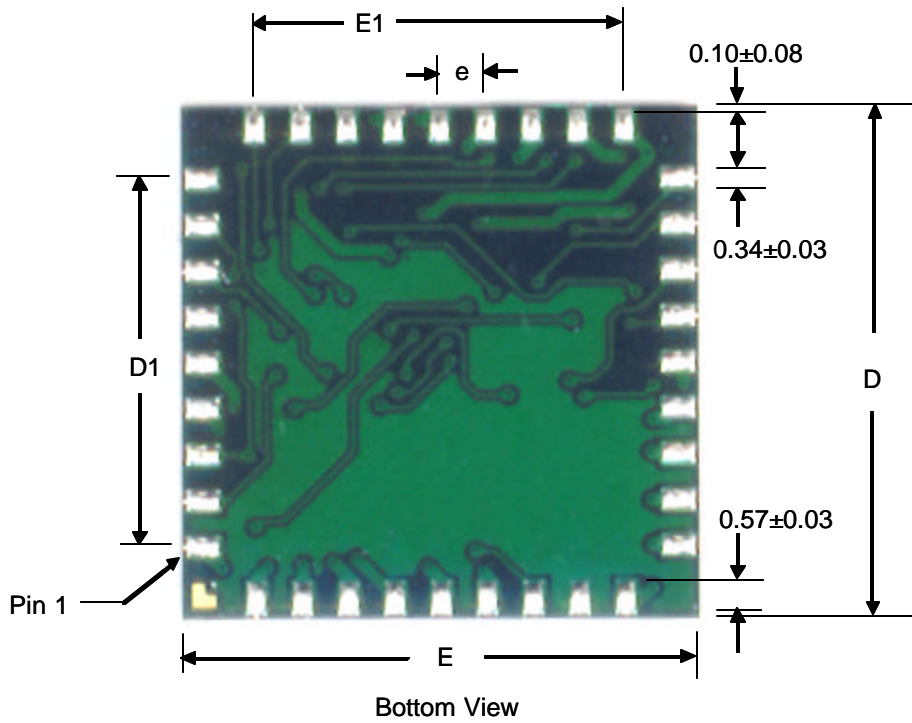
**BOTTOM VIEW**

- 28 NC
- 29 GND1
- 30 NC
- 31 NC
- 32 SCK/SCL
- 33 NC
- 34 NC
- 35 CS\_CTRL
- 36 SDA

# HMC6343

## PACKAGE OUTLINES

PACKAGE DRAWING HMC6343 (32-PIN LPCC, dimensions in millimeters)



Dimensions (mm)	Minimum	Nominal	Maximum
A (height)	1.73	1.87	2.02
D	-	9.00 BSC	-
D1	-	6.40 BSC	-
E	-	9.00 BSC	-
E1	-	6.40 BSC	-
e	-	0.8 Basic	-

## MOUNTING CONSIDERATIONS

The following is the recommend printed circuit board (PCB) footprint for the HMC6343. All dimensions are nominal and in millimeters.

### Stencil Design and Solder Paste

A 4-6 mil stencil and 100% paste coverage is recommended for the electrical contact pads. The HMC6343 has been assembled successfully with no-clean solder paste.

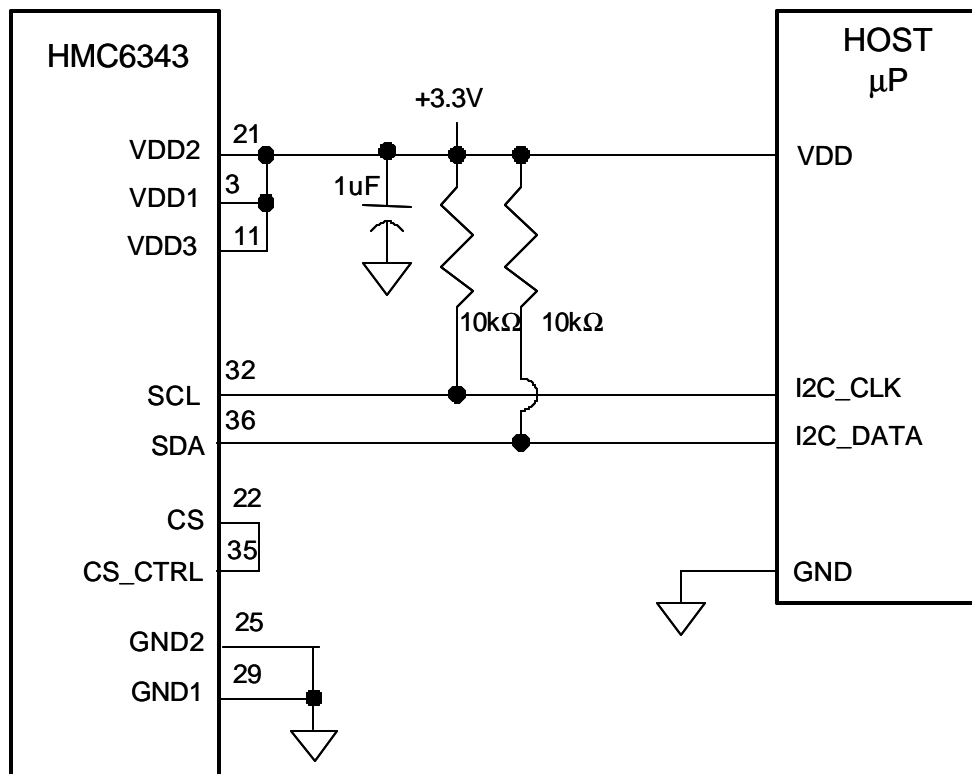
# HMC6343

## BASIC DEVICE OPERATION

The Honeywell HMC6343 magnetoresistive sensor circuit is a trio of magnetic sensors, accelerometers, and analog support circuits to measure magnetic fields. Additionally a microcontroller is integrated for computation of direction and calibration. With power supply applied, the sensor converts any incident magnetic field in the sensitive axis direction to a differential voltage output. In addition to the bridge circuit, the sensors have on-chip magnetically coupled offset straps for incident field adjustment.

The circuit is sensitive to power supply noise, and adding a 1.0 microfarad ceramic capacitor is recommended on the positive supply to help reduce noise. Also careful layout practices should be enforced to keep high current traces (>10mA) a few millimeters away from the sensors. Also, since the sensors are typically sensing the earth's magnetic field direction, avoid employing RF/EMI shields using ferrous metals or coatings.

## BASIC SCHEMATIC INTERFACE



### Offset Straps

The three offset straps have a spiral of metallization that couples in the sensor element's sensitive axis. The straps will handle currents to buck or boost fields through the  $\pm 4$  gauss linear measurement range, but designers should note the thermal heating on the die when doing so.

With most applications, the offset strap is not utilized and can be ignored. Designers can leave one or both strap connections (Off- and Off+) open circuited, or ground one connection node.

### Operational Modes

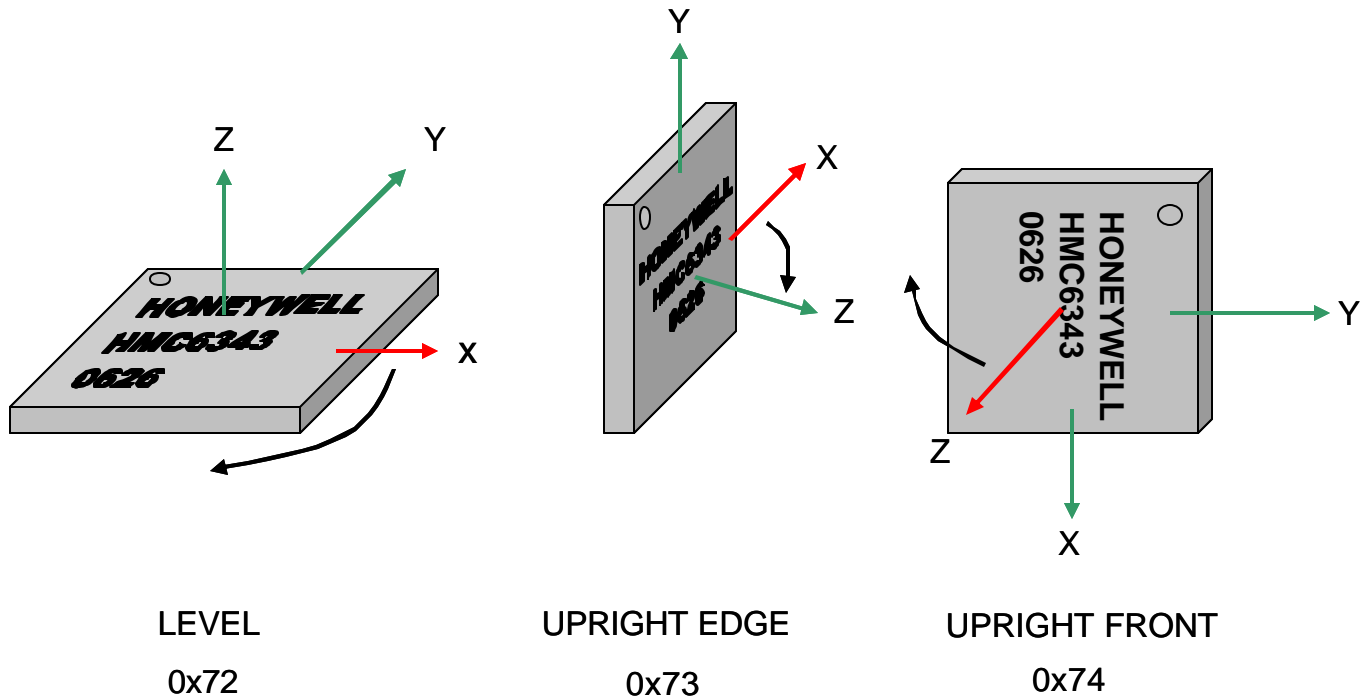
The HMC6343 has three operational modes; Sleep, Standby, and Run. Sleep mode is defined as having the analog circuitry powered off, and has the lowest power consumption while power is applied to the VDD pins. Standby mode has the HMC6343 fully powered, but with no measurements performed and the processor is waiting for commands to perform. Run mode is fully engaged in continuous measurements at the set rate, and ready to receive further commands. The operational mode settings are stored in EEPROM register 0x04, and shown further the HMC6343 protocol definition.

# HMC6343

## Mounting Orientations

The HMC6343 provides for three standard mounting orientations, with a flat horizontal orientation (Level) as the factory default. For vertical mounting, there are two upright orientations with either the X-axis or the Z-axis designated as the forward reference directions. To change the forward reference direction temporarily, send the appropriate command byte (0x72, 0x73, or 0x74) for level or upright orientations. For other orientations, you can add or subtract 90 degree increments of deviation angle as required from the three choices. The figure below shows pictorially the orientations.

To permanently change orientation, poke EEPROM Operational Mode Register 1 (0x04) with the appropriate binary bits set for Level, Upright Edge (UE), or Upright Front (UF). The HMC6343 will operate in the selected orientation after a power-up or reset command. More on the EEPROM registers in the following sections.



## HMC6343 ORIENTATIONS

Red Arrow is the Forward Direction

## I<sup>2</sup>C COMMUNICATION PROTOCOL

The HMC6343 communicates via a two-wire I<sup>2</sup>C bus system as a slave device. The HMC6343 uses a layered protocol with the interface protocol defined by the I<sup>2</sup>C bus specification, and the lower command protocol defined by Honeywell. The data rate is the standard-mode 100kbps rate as defined in the I<sup>2</sup>C Bus Specification 2.1. The bus bit format is an 8-bit Data/Address send and a 1-bit acknowledge bit. The format of the data bytes (payload) shall be case sensitive ASCII characters or binary data to the HMC6343 slave, and binary data returned. Negative binary values will be in two's complement form. The default (factory) HMC6343 7-bit slave address is 0x32 for write operations, or 0x33 for read operations.

The HMC6343 Serial Clock (SCL) and Serial Data (SDA) lines do not have internal pull-up resistors, and require resistive pull-ups (R<sub>p</sub>) between the master device (usually a host microprocessor) and the HMC6343. Pull-up resistance values of about 10k ohms are recommended with a nominal 3.3-volt supply voltage. Other values may be used as defined in the I<sup>2</sup>C Bus Specification 2.1.

The SCL and SDA lines in this bus specification can be connected to a host of devices. The bus can be a single master to multiple slaves, or it can be a multiple master configuration. All data transfers are initiated by the master device which is responsible for generating the clock signal, and the data transfers are 8 bit long. All devices are addressed by I<sup>2</sup>C's unique 7 bit address. After each 8-bit transfer, the master device generates a 9<sup>th</sup> clock pulse, and releases the SDA line.



## HMC6343

The receiving device (addressed slave) will pull the SDA line low to acknowledge (ACK) the successful transfer or leave the SDA high to negative acknowledge (NACK).

Per the I<sup>2</sup>C spec, all transitions in the SDA line must occur when SCL is low. This requirement leads to two unique conditions on the bus associated with the SDA transitions when SCL is high. Master device pulling the SDA line low while the SCL line is high indicates the Start (S) condition, and the Stop (P) condition is when the SDA line is pulled high while the SCL line is high. The I<sup>2</sup>C protocol also allows for the Restart condition in which the master device issues a second start condition without issuing a stop.

All bus transactions begin with the master device issuing the start sequence followed by the slave address byte. The address byte contains the slave address; the upper 7 bits (bits7-1), and the Least Significant bit (LSb). The LSb of the address byte designates if the operation is a read (LSb=1) or a write (LSb=0). At the 9<sup>th</sup> clock pulse, the receiving slave device will issue the ACK (or NACK). Following these bus events, the master will send data bytes for a write operation, or the slave will clock out data with a read operation. All bus transactions are terminated with the master issuing a stop sequence.

I<sup>2</sup>C bus control can be implemented with either hardware logic or in software. Typical hardware designs will release the SDA and SCL lines as appropriate to allow the slave device to manipulate these lines. In a software implementation, care must be taken to perform these tasks in code.

### I<sup>2</sup>C Slave Address

The I<sup>2</sup>C slave address byte consists of the 7 most significant bits with the least significant bit zero filled. As described earlier, the default (factory) value is 0x32 and the legal I<sup>2</sup>C bounded values are between 0x10 and 0xF6. This slave address is in EEPROM address 0x00. Users can change the slave address by writing to this location. Any address updates will become effective after the next power up or after a reset command.

### Software Version

This EEPROM software version number byte contains the binary value of the programmed software. Values of 0x05 and beyond are considered production software.

### Deviation Angle Correction

Typically the HMC6343 X-axis (or Z-axis) is designated the forward direction of the compass, and is placed mechanically towards the forward direction of the end user product. The deviation angle is used to correct for mechanical angle errors in package orientation by adding the deviation angle to the internal compass heading before the result is placed as the computed heading. Two EEPROM Bytes are used to store the deviation angle, and the binary value is in tenths of a degree and in two's complement form for a  $\pm 1800$  representation. The deviation angle MSB is located in EEPROM register 0x0B and the LSB in 0x0A.

### Variation Angle Correction

The variation angle or declination angle of the HMC6343 is the number of degree that must be added to the internal compass heading to convert the magnetic north reference direction to the geographic (true) north reference direction. This angle information is provided to the HMC6343 from external latitude and longitude data processed through a World Magnetic Model equation to compute variation angle, or by lookup table. Two EEPROM Bytes are used to store the variation angle, and the binary value is in tenths of a degree and in two's complement form for a  $\pm 1800$  representation. The deviation angle MSB is located in EEPROM register 0x0D and the LSB in 0x0C.

### Magnetometer Offsets

The Magnetometer Offset bytes are the values stored after the completion of the last factory or user hard-iron calibration routine. Additional value changes are possible, but will be overwritten when the next calibration routine is completed. Note that these offset values are added to the sensor offset values computed by the set/reset routine to convert the raw magnetometer data to the compensated magnetometer data. These values are written into EEPROM addresses 0x0E to 0x13 and loaded to RAM on the power up.

# HMC6343

## Heading Filter

This allows for an Infinite Impulse Response (IIR) filter to be employed on current and previous heading data outputs. Typical values are 0 to 15 with a factory default of zero. The filter is only applied in run mode where a continuous stream of data is present. At the 5Hz default update rate, a filter value of 4 would weight the latest heading with the previous four headings of regressive weightings for a second's worth of filtering.

## EEPROM Registers

The HMC6343 contains EEPROM non-volatile memory locations (registers) to store useful compass data for processor routines. The following Table shows the register locations, content, description, and factory shipped defaults.

**Table 1 – EEPROM Registers**

EEPROM Location	Content	Description	Factory Default
0x00	Slave Address	I2C Slave Address	0x32
0x01	Reserved		
0x02	S/W_Version	Software Version Number	
0x03	Reserved		
0x04	OP_Mode1	Operational Mode Register 1	0x11
0x05	OP_Mode2	Operational Mode Register 2	0x01
0x06	S/N LSB	Device Serial Number	
0x07	S/N MSB	Device Serial Number	
0x08	Date Code: YY	Package Date Code: Last Two Digits of the Year	Year
0x09	Date Code: WW	Package Date Code: Fiscal Week	Week
0x0A	Deviation LSB	Deviation Angle ( $\pm 1800$ ) in tenths of a degree	0x00
0x0B	Deviation MSB	Deviation Angle ( $\pm 1800$ ) in tenths of a degree	0x00
0x0C	Variation LSB	Variation Angle ( $\pm 1800$ ) in tenths of a degree	0x00
0x0D	Variation MSB	Variation Angle ( $\pm 1800$ ) in tenths of a degree	0x00
0x0E	X_Offset LSB	Hard-Iron Calibration Offset for the X-axis	0x00
0x0F	X_Offset MSB	Hard-Iron Calibration Offset for the X-axis	0x00
0x10	Y_Offset LSB	Hard-Iron Calibration Offset for the Y-axis	0x00
0x11	Y_Offset MSB	Hard-Iron Calibration Offset for the Y-axis	0x00
0x12	Z_Offset LSB	Hard-Iron Calibration Offset for the Z-axis	0x00
0x13	Z_Offset MSB	Hard-Iron Calibration Offset for the Z-axis	0x00
0x14	Filter LSB	Heading IIR Filter (0x00 to 0x0F typical)	0x00
0x15	Filter MSB	Heading IIR Filter (set at zero)	0x00

# HMC6343

## Command Protocol

The command protocol defines the content of the data (payload) bytes of I<sup>2</sup>C protocol sent by the master, and the slave device (HMC6343). Note that angular outputs are in tenths of a degree (0-3600 heading, ±0-900 tilt).

After the master device sends the 7-bit slave address, the 1-bit Read/Write, and gets the 1-bit slave device acknowledge bit returned; the next one to three sent data bytes are defined as the input command and argument bytes. To conserve data traffic, all response data (Reads) will be context sensitive to the last command (Write) sent. All write commands shall have the address byte least significant bit cleared (factory default 0x32). These commands then follow with the command byte and command specific binary formatted argument bytes in the general form of:

(Command Byte) (Argument Binary MS Byte) (Argument Binary LS Byte)

The slave (HMC6343) shall provide the acknowledge bits between each data byte per the I<sup>2</sup>C protocol. Response byte reads are done by sending the address byte (factory default 0x33) with the least significant bit set, and then clocking back response bytes, last command dependant. Table 2 shows the HMC6343 command and response data flow.

**Table 2 – HMC6343 Interface Commands/Responses**

Command Byte (hex)	Argument 1 Byte (Binary)	Argument 2 Byte (Binary)	Response Bytes (Binary)	Command Description
(0x40)			MSB/LSB Data (6 Bytes)	Post Accel Data. AxMSB, AxLSB, AyMSB, AyLSB, AzMSB, AzLSB
(0x45)			MSB/LSB Data (6 Bytes)	Post Mag Data. MxMSB, MxLSB, MyMSB, MyLSB, MzMSB, MzLSB
(0x50)			MSB/LSB Data (6 Bytes)	Post Heading Data. HeadMSB, HeadLSB, PitchMSB, PitchLSB, RollMSB, RollLSB
(0x55)			MSB/LSB Data (6 Bytes)	Post Tilt Data. PitchMSB, PitchLSB, RollMSB, RollLSB, TempMSB, TempLSB
(0x65)			Post OP Mode 1	Read the current value of OP Mode 1
(0x71)				Enter User Calibration Mode
(0x72)				Level Orientation (X=forward, +Z=up) (default)
(0x73)				Upright Sideways Orientation (X=forward, Y=up)
(0x74)				Upright Flat Front Orientation (Z=forward, -X=up)
(0x75)				Enter Run Mode (from Standby Mode)
(0x76)				Enter Standby Mode (from Run Mode)
(0x7E)				Exit User Calibration Mode
(0x82)				Reset the Processor
(0x83)				Enter Sleep Mode (from Run Mode)
(0x84)				Exit Sleep Mode (to Standby Mode)
(0xE1)	EEPROM Address		Data (1 Byte)	Read from EEPROM
(0xF1)	EEPROM Address	Data		Write to EEPROM

# HMC6343

## Timing

Upon power application to the HMC6343, wait nominally 500 milli-seconds before sending the first I2C command (typically a 0x32 byte followed by a 0x50 byte for the usual heading/pitch/roll). Depending on the command sent, a delay time should be inserted before clocking out the response bytes (send 0x33, clock back response bytes). The following table indicates the response delay times for various commands.

**Table 3 – HMC6343 Command to Response Delay Times**

Prior Command (hex)	Commanded Action	Response Bytes & Description	Response/Delay Time (milli-seconds)
Power Applied	VDD1-3 low to high	No Response Data	500 nominally
0x40	Post Accel Data.	6 binary data Bytes. AxMSB, AxLSB, AyMSB, AyLSB, AzMSB, AzLSB	1
0x45	Post Mag Data.	6 binary data Bytes. MxMSB, MxLSB, MyMSB, MyLSB, MzMSB, MzLSB	1
0x50	Post Heading Data.	6 binary data Bytes. HeadMSB, HeadLSB, PitchMSB, PitchLSB, RollMSB, RollLSB	1
0x55	Post Tilt Data.	6 binary data Bytes. PitchMSB, PitchLSB, RollMSB, RollLSB, TempMSB, TempLSB	1
0x65	Post OP Mode 1	OP Mode 1	1
0x71	Enter User Calibration Mode	No Response Data	0.3
0x72	Level Orientation	(X=forward, +Z=up) (default) No Response Data	0.3
0x73	Upright Sideways Orientation	(X=forward, Y=up) No Response Data	0.3
0x74	Upright Flat Front Orientation	(Z=forward, -X=up) No Response Data	0.3
0x75	Enter Run Mode	No Response Data	0.3
0x76	Enter Standby Mode	No Response Data	0.3
0x7E	Exit User Calibration Mode	No Response Data	50
0x82	Reset the Processor	No Response Data	500
0x83	Enter Sleep Mode	No Response Data	1
0x84	Exit Sleep Mode	No Response Data	20
0xE1	Read from EEPROM, RAM	1 binary data Byte	10
0xF1	Write to EEPROM, RAM	No Response Data. Data Settling Time	10

# HMC6343

## Operational Mode Registers

EEPROM registers 0x04 and 0x05 contain bits that are read for operational mode status and for setting the Run Mode measurement rate. The tables below describe the register contents and interpretation. It is recommended that Operational Mode Register 1 and 2 written only to change default orientation and update measurement rate.

**Table 4 – Operational Mode Register 1 (EEPROM 0x04)**

OM1_7	OM1_6	OM1_5	OM1_4	OM1_3	OM1_2	OM1_1	OM1_0
Comp(0)	Cal(0)	Filter(0)	Run(1)	Stdby(0)	UF(0)	UE(0)	Level(1)

**Table 5 – Operational Mode Register 1 Bit Designations**

Location	Name	Description
OM1_7	Comp	Calculating compass data if set. (read only)
OM1_6	Cal	Calculating calibration offsets if set. (read only)
OM1_5	Filter	IIR Heading Filter used if set.
OM1_4	Run	Run Mode if set.
OM1_3	Stdby	Standby Mode if set.
OM1_2	UF	Upright Front Orientation if set.
OM1_1	UE	Upright Edge Orientation if set.
OM1_0	Level	Level Orientation if set

**Table 6 – Operational Mode Register 2 (EEPROM 0x05)**

OM2_7	OM2_6	OM2_5	OM2_4	OM2_3	OM2_2	OM2_1	OM2_0
(0)	(0)	(0)	(0)	(0)	(0)	MR1(0)	MR0(1)

**Table 7 – Operational Mode Register 2 Bit Designations**

Location	Name	Description
OM2_7 to OM2_2	0	These bits must be cleared for correct operation.
OM2_1 to OM2_0	MR1, MR0	Measurement Rate 0,0 = 1Hz 0,1 = 5Hz (default) 1,0 = 10Hz 1,1 = Not Assigned

## User Hard-Iron Calibration

The HMC6343 provides a user calibration routine with the 0x71 command permitting entry into the calibration mode and the 0x7E command to exit the calibration mode.

After entering the calibration mode, rotate the device reasonably steady for 360 degrees about the Y (Left - Right) axis and then 360 degrees about Z (Up - Down) axis. During the first rotation, maintain the Y axis at Level as much as possible. Maintain the Z axis upright as much as possible during the second rotation and until the exit calibration

# HMC6343

command is issued. The first rotation can also be done by rotating 360 degrees about X (Fore -Aft) axis. Then exit calibration.

The calibration routine collects these readings to correct for hard-iron distortions of the magnetic field. These hard-iron effects are due to magnetized materials nearby the HMC6343 part that in a fixed position with respect to the end user platform. An example would be the magnetized chassis or engine block of a vehicle in which the compass is mounted onto. Upon exiting the calibration mode, the resulting magnetometer offsets are updated.

## Example Communication

For basic power up and compassing using the defaults, the following order of operations is recommended:

1. Apply power to the VDD pins (nominally +3.3 volts)
2. Wait at least 500 milli-seconds for device initialization. The HMC6343 is in the default Run Mode.
3. Send 0x32 and 0x50 to command the Heading and Tilt Data to be clocked out next.
4. Wait at least 1 milli-second to allow the HMC6343 to process the command.
5. Send 0x33 and clock back six more response Bytes from the HMC6343. These will be the Heading, Pitch and Roll Byte pairs; binary format in tenths of a degree with 2's compliment on pitch and roll angles. (0 to 3600 heading,  $\pm 900$  pitch, and  $\pm 900$  roll)
6. Repeat steps 3 - 5 every 200 milli-seconds or longer to get fresh data from the default 5Hz update rate.

## ORDERING INFORMATION

Ordering Number	Product	Packaging
HMC6343	3 axis Compass with Algorithms	Tubes
HMC6343-demo	Development Kit	Demo Board, USB Cable and Demo Software
HMC6343-eval	Evaluation Board	Board



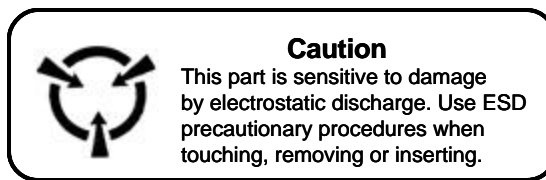
## FIND OUT MORE

For more information on Honeywell's Magnetic Sensors visit us online at [www.magneticsensors.com](http://www.magneticsensors.com) or contact us at 800-323-8295 (763-954-2474 internationally).

The application circuits herein constitute typical usage and interface of Honeywell product. Honeywell does not warranty or assume liability of customer-designed circuits derived from this description or depiction.

Honeywell reserves the right to make changes to improve reliability, function or design. Honeywell does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights nor the rights of others.

U.S. Patents 4,441,072, 4,533,872, 4,569,742, 4,681,812, 4,847,584 and 6,529,114 apply to the technology described



**CAUTION: ESDS CAT. 1A**

Honeywell  
12001 Highway 55  
Plymouth, MN 55441  
Tel: 800-323-8295  
[www.honeywell.com](http://www.honeywell.com)

Form #900357  
October 2008  
©2008 Honeywell International Inc.

**Honeywell**

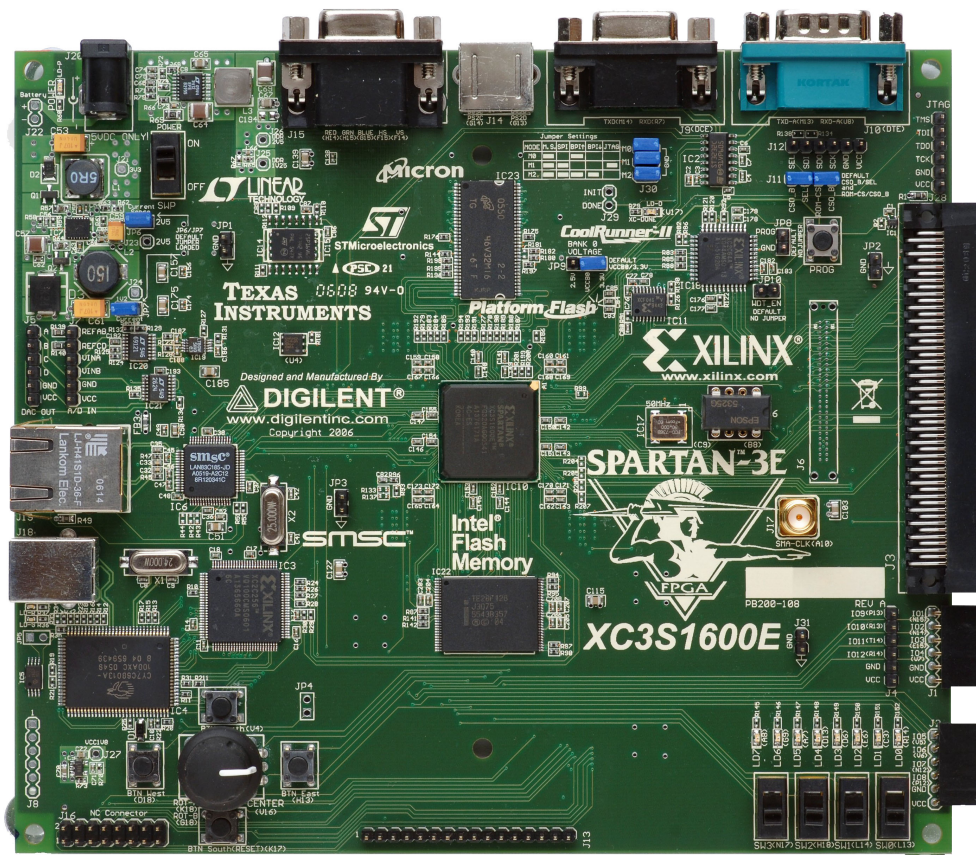
1	!OE	41	21	!ICE	6
B4	~	~	A13	~	~
2	D10	31	22	A09	22
A4	D10	31	B13	A09	22
3	D09	30	23	A08	21
D5	D09	30	A14	A08	21
4	D08	29	24	A7	20
C5	D08	29	B14	A7	20
5	A18	28	25	A06	19
A6	A18	28	C14	A06	19
6	A14	27	26	D03	10
B6	A14	27	D14	D03	10
7	A10	23	27	D02	9
E7	A10	23	A16	D02	9
8	A11	24	28	D01	8
F7	A11	24	B16	D01	8
9	A12	25	29	D00	7
D7	A12	25	E13	D00	7
10	A13	26	30	A04	5
C7	A13	26	C4	A04	5
11	D11	32	31	A03	4
F8	D11	32	B11	A03	4
12	D12	35	32	A02	3
E8	D12	35	A11	A02	3
13	D13	36	33	A01	2
F9	D13	36	A8	A01	2
14	D14	37	34	A00	1
E9	D14	37	G9	A00	1
15	D15	38	35	D04	13
D11	D15	38	A7	D04	13
16	A15	42	36	D05	14
C11	A15	42	D13	D05	14
17	A16	43	37	D06	15
F11	A16	43	E6	D06	15
18	A17	44	38	D07	16
E11	A17	44	D6	D07	16
19	~	~	39	!WE	17
E12	!OE	41	C3	!WE	17
20	~	~	40	A05	18
F12	!ICE	6	C15	A05	18

23	A10	A9	22
24	A11	A8	21
25	A12	A7	20
26	A13	A6	19
27	A14	A5	18
28	A18	!WE	17
29	D8	D7	16
30	D9	D6	15
31	D10	D5	14
32	D11	D4	13
33	VDD	GND	12
34	GND	VDD	11
35	D12	D3	10
36	D13	D2	9
37	D14	D1	8
38	D15	D0	7
39	!LB	!ICE	6
40	!UB	A4	5
41	!OE	A3	4
42	A15	A2	3
43	A16	A1	2
44	A17	A0	1

23	A10	A9	22
24	A11	A8	21
25	A12	A7	20
26	A13	A6	19
27	A14	A5	18
28	A18	!WE	17
29	D8	D7	16
30	D9	D6	15
31	D10	D5	14
32	D11	D4	13
33	VDD	GND	12
34	GND	VDD	11
35	D12	D3	10
36	D13	D2	9
37	D14	D1	8
38	D15	D0	7
39	!LB	!ICE	6
40	!UB	A4	5
41	!OE	A3	4
42	A15	A2	3
43	A16	A1	2
44	A17	A0	1

# MicroBlaze Development Kit Spartan-3E 1600E Edition User Guide

UG257 (v1.1) December 5, 2007







Xilinx is disclosing this Document and Intellectual Property (hereinafter “the Design”) to you for use in the development of designs to operate on, or interface with Xilinx FPGAs. Except as stated herein, none of the Design may be copied, reproduced, distributed, republished, downloaded, displayed, posted, or transmitted in any form or by any means including, but not limited to, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of Xilinx. Any unauthorized use of the Design may violate copyright laws, trademark laws, the laws of privacy and publicity, and communications regulations and statutes.

Xilinx does not assume any liability arising out of the application or use of the Design; nor does Xilinx convey any license under its patents, copyrights, or any rights of others. You are responsible for obtaining any rights you may require for your use or implementation of the Design. Xilinx reserves the right to make changes, at any time, to the Design as deemed desirable in the sole discretion of Xilinx. Xilinx assumes no obligation to correct any errors contained herein or to advise you of any correction if such be made. Xilinx will not assume any liability for the accuracy or correctness of any engineering or technical support or assistance provided to you in connection with the Design.

THE DESIGN IS PROVIDED “AS IS” WITH ALL FAULTS, AND THE ENTIRE RISK AS TO ITS FUNCTION AND IMPLEMENTATION IS WITH YOU. YOU ACKNOWLEDGE AND AGREE THAT YOU HAVE NOT RELIED ON ANY ORAL OR WRITTEN INFORMATION OR ADVICE, WHETHER GIVEN BY XILINX, OR ITS AGENTS OR EMPLOYEES. XILINX MAKES NO OTHER WARRANTIES, WHETHER EXPRESS, IMPLIED, OR STATUTORY, REGARDING THE DESIGN, INCLUDING ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE, AND NONINFRINGEMENT OF THIRD-PARTY RIGHTS.

IN NO EVENT WILL XILINX BE LIABLE FOR ANY CONSEQUENTIAL, INDIRECT, EXEMPLARY, SPECIAL, OR INCIDENTAL DAMAGES, INCLUDING ANY LOST DATA AND LOST PROFITS, ARISING FROM OR RELATING TO YOUR USE OF THE DESIGN, EVEN IF YOU HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. THE TOTAL CUMULATIVE LIABILITY OF XILINX IN CONNECTION WITH YOUR USE OF THE DESIGN, WHETHER IN CONTRACT OR TORT OR OTHERWISE, WILL IN NO EVENT EXCEED THE AMOUNT OF FEES PAID BY YOU TO XILINX HEREUNDER FOR USE OF THE DESIGN. YOU ACKNOWLEDGE THAT THE FEES, IF ANY, REFLECT THE ALLOCATION OF RISK SET FORTH IN THIS AGREEMENT AND THAT XILINX WOULD NOT MAKE AVAILABLE THE DESIGN TO YOU WITHOUT THESE LIMITATIONS OF LIABILITY.

The Design is not designed or intended for use in the development of on-line control equipment in hazardous environments requiring fail-safe controls, such as in the operation of nuclear facilities, aircraft navigation or communications systems, air traffic control, life support, or weapons systems (“High-Risk Applications”). Xilinx specifically disclaims any express or implied warranties of fitness for such High-Risk Applications. You represent that use of the Design in such High-Risk Applications is fully at your risk.

© 2002-2006 Xilinx, Inc. All rights reserved. XILINX, the Xilinx logo, and other designated brands included herein are trademarks of Xilinx, Inc. All other trademarks are the property of their respective owners.

---

---

## Revision History

The following table shows the revision history for this document.

Date	Version	Revision
6/23/06	1.0	Initial release.
12/5/07	1.1	Updated Figures 15-8, 15-9, and 15-10 to comply with UCF I/O location constraints.

# Table of Contents

---

## Preface: About This Guide

Acknowledgements .....	7
Guide Contents .....	7
Additional Resources .....	8

## Chapter 1: Introduction and Overview

<b>Choose the Starter Kit Board for Your Needs</b> .....	9
Spartan-3E FPGA Features and Embedded Processing Functions .....	9
Learning Xilinx FPGA, CPLD, and ISE Development Software Basics .....	9
Advanced Spartan-3 Generation Development Boards .....	9
<b>Key Components and Features</b> .....	10
<b>Design Trade-Offs</b> .....	11
Configuration Methods Galore! .....	11
Voltages for all Applications .....	11
<b>Related Resources</b> .....	11

## Chapter 2: Switches, Buttons, and Knob

<b>Slide Switches</b> .....	13
Locations and Labels .....	13
Operation .....	13
UCF Location Constraints .....	13
<b>Push-Button Switches</b> .....	14
Locations and Labels .....	14
Operation .....	14
UCF Location Constraints .....	15
<b>Rotary Push-Button Switch</b> .....	15
Locations and Labels .....	15
Operation .....	15
UCF Location Constraints .....	17
<b>Discrete LEDs</b> .....	17
Locations and Labels .....	17
Operation .....	18
UCF Location Constraints .....	18
<b>Related Resources</b> .....	18

## Chapter 3: Clock Sources

<b>Overview</b> .....	19
<b>Clock Connections</b> .....	20
<b>Voltage Control</b> .....	20
<b>50 MHz On-Board Oscillator</b> .....	20
<b>Auxiliary Clock Oscillator Socket</b> .....	20
<b>SMA Clock Input or Output Connector</b> .....	20

<b>UCF Constraints</b> .....	20
Location .....	21
Clock Period Constraints .....	21
<b>Related Resources</b> .....	21

## **Chapter 4: FPGA Configuration Options**

<b>Configuration Mode Jumpers</b> .....	25
<b>PROG Push Button</b> .....	26
<b>DONE Pin LED</b> .....	26
<b>Programming the FPGA, CPLD, or Platform Flash PROM via USB</b> .....	27
Connecting the USB Cable .....	27
Programming via iMPACT .....	28
Programming Platform Flash PROM via USB .....	30

## **Chapter 5: Character LCD Screen**

<b>Overview</b> .....	41
<b>Character LCD Interface Signals</b> .....	42
<b>Voltage Compatibility</b> .....	42
<b>Interaction with Intel StrataFlash</b> .....	43
<b>UCF Location Constraints</b> .....	43
<b>LCD Controller</b> .....	44
Memory Map .....	44
Command Set .....	46
<b>Operation</b> .....	50
Four-Bit Data Interface .....	50
Transferring 8-Bit Data over the 4-Bit Interface .....	51
Initializing the Display .....	51
Writing Data to the Display .....	52
Disabling the Unused LCD .....	52
<b>Related Resources</b> .....	52

## **Chapter 6: VGA Display Port**

<b>Signal Timing for a 60 Hz, 640x480 VGA Display</b> .....	54
<b>VGA Signal Timing</b> .....	56
<b>UCF Location Constraints</b> .....	57
<b>Related Resources</b> .....	57

## **Chapter 7: RS-232 Serial Ports**

<b>Overview</b> .....	59
<b>UCF Location Constraints</b> .....	60

## **Chapter 8: PS/2 Mouse/Keyboard Port**

<b>Keyboard</b> .....	64
<b>Mouse</b> .....	66
<b>Voltage Supply</b> .....	67

UCF Location Constraints .....	67
Related Resources .....	67

## Chapter 9: Digital to Analog Converter (DAC)

<b>SPI Communication</b> .....	69
Interface Signals .....	70
Disable Other Devices on the SPI Bus to Avoid Contention .....	70
SPI Communication Details .....	71
Communication Protocol .....	71
<b>Specifying the DAC Output Voltage</b> .....	72
DAC Outputs A and B .....	72
DAC Outputs C and D .....	73
UCF Location Constraints .....	73
Related Resources .....	73

## Chapter 10: Analog Capture Circuit

Digital Outputs from Analog Inputs .....	76
<b>Programmable Pre-Amplifier</b> .....	77
Interface .....	77
Programmable Gain .....	77
SPI Control Interface .....	78
UCF Location Constraints .....	79
<b>Analog to Digital Converter (ADC)</b> .....	79
Interface .....	79
SPI Control Interface .....	79
UCF Location Constraints .....	80
Disable Other Devices on the SPI Bus to Avoid Contention .....	81
Connecting Analog Inputs .....	81
Related Resources .....	81

## Chapter 11: Intel StrataFlash Parallel NOR Flash PROM

StrataFlash Connections .....	84
<b>Shared Connections</b> .....	87
Character LCD .....	87
Xilinx XC2C64A CPLD .....	87
SPI Data Line .....	87
<b>UCF Location Constraints</b> .....	88
Address .....	88
Data .....	88
Control .....	89
Setting the FPGA Mode Select Pins .....	89
Related Resources .....	89

## Chapter 12: SPI Serial Flash

UCF Location Constraints .....	92
<b>Configuring from SPI Flash</b> .....	92
Setting the FPGA Mode Select Pins .....	92

Creating an SPI Serial Flash PROM File .....	93
Downloading the Design to SPI Flash .....	98
Downloading the SPI Flash using XSPI .....	98
<b>Additional Design Details</b> .....	101
Shared SPI Bus with Peripherals .....	101
Other SPI Flash Control Signals .....	102
Variant Select Pins, VS[2:0] .....	102
Jumper Block J11 .....	102
Programming Header J12 .....	102
Multi-Package Layout .....	102
<b>Related Resources</b> .....	104

## Chapter 13: DDR SDRAM

<b>DDR SDRAM Connections</b> .....	106
<b>UCF Location Constraints</b> .....	108
Address .....	108
Data .....	108
Control .....	109
Reserve FPGA VREF Pins .....	109
<b>Related Resources</b> .....	109

## Chapter 14: 10/100 Ethernet Physical Layer Interface

<b>Ethernet PHY Connections</b> .....	112
<b>MicroBlaze Ethernet IP Cores</b> .....	113
<b>UCF Location Constraints</b> .....	114
<b>Related Resources</b> .....	114

## Chapter 15: Expansion Connectors

<b>Hirose 100-pin FX2 Edge Connector (J3)</b> .....	115
Voltage Supplies to the Connector .....	116
Connector Pinout and FPGA Connections .....	116
Compatible Board .....	118
Mating Receptacle Connectors .....	118
Differential I/O .....	118
UCF Location Constraints .....	122
<b>Six-Pin Accessory Headers</b> .....	123
Header J1 .....	123
Header J2 .....	123
Header J4 .....	124
<b>UCF Location Constraints</b> .....	124
<b>Connectorless Debugging Port Landing Pads (J6)</b> .....	125
<b>Related Resources</b> .....	126

## Chapter 16: XC2C64A CoolRunner-II CPLD

<b>UCF Location Constraints</b> .....	128
FPGA Connections to CPLD .....	129
CPLD .....	129

Related Resources .....	130
-------------------------	-----

## **Chapter 17: DS2432 1-Wire SHA-1 EEPROM**

UCF Location Constraints .....	131
Related Resources .....	131

## **Appendix A: Schematics**

FX2 Expansion Header, 6-pin Headers, and Connectorless Probe Header . . . .	134
RS-232 Ports, VGA Port, and PS/2 Port .....	136
Ethernet PHY, Magnetics, and RJ-11 Connector .....	138
Voltage Regulators .....	140
FPGA Configurations Settings, Platform Flash PROM, SPI Serial Flash, JTAG Connections .....	142
FPGA I/O Banks 0 and 1, Oscillators .....	144
FPGA I/O Banks 2 and 3 .....	146
Power Supply Decoupling .....	148
XC2C64A CoolRunner-II CPLD .....	150
Linear Technology ADC and DAC .....	152
Intel StrataFlash Parallel NOR Flash Memory and Micron DDR SDRAM . . .	154
Buttons, Switches, Rotary Encoder, and Character LCD .....	156
DDR SDRAM Series Termination and FX2 Connector Differential Termination	158

## **Appendix B: Example User Constraints File (UCF)**



## About This Guide

---

This user guide provides basic information on the MicroBlaze Development Kit board capabilities, functions, and design. It includes general information on how to use the various peripheral functions included on the board. For detailed reference designs, including VHDL or Verilog source code, please visit the following web link.

- Spartan™-3E Starter Kit Board Reference Page  
<http://www.xilinx.com/sp3e1600e>

## Acknowledgements

Xilinx wishes to thank the following companies for their support of the MicroBlaze Development Kit board:

- Intel Corporation for the 128 Mbit StrataFlash memory
- Linear Technology for the SPI-compatible A/D and D/A converters, the programmable pre-amplifier, and the power regulators for the non-FPGA components
- Micron Technology, Inc. for the 32M x 16 DDR SDRAM
- SMSC for the 10/100 Ethernet PHY
- STMicroelectronics for the 16M x 1 SPI serial Flash PROM
- Texas Instruments Incorporated for the three-rail TPS75003 regulator supplying most of the FPGA supply voltages
- Xilinx, Inc. Configuration Solutions Division for the XCF04S Platform Flash PROM and their support for the embedded USB programmer
- Xilinx, Inc. CPLD Division for the XC2C64A CoolRunner™-II CPLD

## Guide Contents

This manual contains the following chapters:

- [Chapter 1, "Introduction and Overview,"](#) provides an overview of the key features of the MicroBlaze Development Kit board.
- [Chapter 2, "Switches, Buttons, and Knob,"](#) defines the switches, buttons, and knobs present on the MicroBlaze Development Kit board.
- [Chapter 3, "Clock Sources,"](#) describes the various clock sources available on the MicroBlaze Development Kit board.
- [Chapter 4, "FPGA Configuration Options,"](#) describes the configuration options for the FPGA on the MicroBlaze Development Kit board.



- [Chapter 5, “Character LCD Screen,”](#) describes the functionality of the character LCD screen.
- [Chapter 6, “VGA Display Port,”](#) describes the functionality of the VGA port.
- [Chapter 7, “RS-232 Serial Ports,”](#) describes the functionality of the RS-232 serial ports.
- [Chapter 8, “PS/2 Mouse/Keyboard Port,”](#) describes the functionality of the PS/2 mouse and keyboard port.
- [Chapter 9, “Digital to Analog Converter \(DAC\),”](#) describes the functionality of the DAC.
- [Chapter 10, “Analog Capture Circuit,”](#) describes the functionality of the A/D converter with a programmable gain pre-amplifier.
- [Chapter 11, “Intel StrataFlash Parallel NOR Flash PROM,”](#) describes the functionality of the StrataFlash PROM.
- [Chapter 12, “SPI Serial Flash,”](#) describes the functionality of the SPI Serial Flash memory.
- [Chapter 13, “DDR SDRAM,”](#) describes the functionality of the DDR SDRAM.
- [Chapter 14, “10/100 Ethernet Physical Layer Interface,”](#) describes the functionality of the 10/100Base-T Ethernet physical layer interface.
- [Chapter 15, “Expansion Connectors,”](#) describes the various connectors available on the MicroBlaze Development Kit board.
- [Chapter 16, “XC2C64A CoolRunner-II CPLD”](#) describes how the CPLD is involved in FPGA configuration when using Master Serial and BPI mode.
- [Chapter 17, “DS2432 1-Wire SHA-1 EEPROM”](#) provides a brief introduction to the SHA-1 secure EEPROM for authenticating or copy-protecting FPGA configuration bitstreams.
- [Appendix A, “Schematics,”](#) lists the schematics for the MicroBlaze Development Kit board.
- [Appendix B, “Example User Constraints File \(UCF\),”](#) provides example code from a UCF.

## Additional Resources

To find additional resources for the MicroBlaze Processor or the Xilinx Embedded development tools, see the Xilinx website at:

<http://www.Xilinx.com/Microblaze> or <http://www.Xilinx.com/EDK>

To find additional documentation, see the Xilinx website at:

<http://www.xilinx.com/literature>.

To search the Answer Database of silicon, software, and IP questions and answers, or to create a technical support WebCase, see the Xilinx website at:

<http://www.xilinx.com/support>.

## Introduction and Overview

---

Thank you for purchasing the Xilinx MicroBlaze™ Development Kit Spartan™-3E 1600E Edition. You will find it useful in developing your Spartan-3E FPGA application.

### Choose the Starter Kit Board for Your Needs

Depending on specific requirements, choose the Xilinx development board that best suits your needs.

### Spartan-3E FPGA Features and Embedded Processing Functions

The MicroBlaze Development Kit board highlights the unique features of the Spartan-3E FPGA family and provides a convenient development board for embedded processing applications. The board highlights these features:

- Spartan-3E specific features
  - ◆ Parallel NOR Flash configuration
  - ◆ MultiBoot FPGA configuration from Parallel NOR Flash PROM
  - ◆ SPI serial Flash configuration
- Embedded development
  - ◆ MicroBlaze 32-bit embedded RISC processor
  - ◆ PicoBlaze™ 8-bit embedded controller
  - ◆ DDR memory interfaces
  - ◆ 10-100 Ethernet
  - ◆ UART

### Learning Xilinx FPGA, CPLD, and ISE Development Software Basics

The MicroBlaze Development Kit board is more advanced and complex compared to other Spartan development boards.

### Advanced Spartan-3 Generation Development Boards

The MicroBlaze Development Kit board demonstrates the basic capabilities of the MicroBlaze embedded processor and the Xilinx Embedded Development Kit (EDK). For more advanced development on a board with additional peripherals and FPGA logic, consider the V4 FX12 Board:

- [http://www.xilinx.com/xlnx/xebiz/designResources/ip\\_product\\_details.jsp?key=DO-ML403-EDK-ISE](http://www.xilinx.com/xlnx/xebiz/designResources/ip_product_details.jsp?key=DO-ML403-EDK-ISE)

Also consider the capable boards offered by Xilinx partners:

- <http://www.xilinx.com/products/devboards/index.htm>

## Key Components and Features

The key features of the MicroBlaze Development Kit board are:

- Xilinx XC3S1600E Spartan-3E FPGA
  - ◆ Up to 250
  - ◆ user-I/O pins
  - ◆ 320-pin FBGA package
  - ◆ Over 33,000 logic cells
- Two Xilinx 4 Mbit Platform Flash configuration PROM
- Xilinx 64-macrocell XC2C64A CoolRunner CPLD
- 64 MByte (512 Mbit) of DDR SDRAM, x16 data interface, 100+ MHz
- 16 MByte (128 Mbit) of parallel NOR Flash (Intel StrataFlash)
  - ◆ FPGA configuration storage
  - ◆ MicroBlaze code storage/shadowing
- 16 Mbits of SPI serial Flash (STMicro)
  - ◆ FPGA configuration storage
  - ◆ MicroBlaze code shadowing
- 2 x 16 LCD display screen
- PS/2 mouse or keyboard port
- VGA display port
- 10/100 Ethernet PHY (requires Ethernet MAC in FPGA)
- Two 9-pin RS-232 ports (DTE- and DCE-style)
- On-board USB-based FPGA/CPLD download/debug interface
- 50 MHz and 66 MHz clock oscillators
- SHA-1 1-wire serial EEPROM for bitstream copy protection
- Hirose FX2 expansion connector with 40-user I/O
- Three Digilent 6-pin expansion connectors
- Four-output, SPI-based Digital-to-Analog Converter (DAC)
- Two-input, SPI-based Analog-to-Digital Converter (ADC) with programmable-gain pre-amplifier
- ChipScope™ SoftTouch debugging port
- Rotary-encoder with push-button shaft
- Eight discrete LEDs
- Four slide switches
- Four push-button switches
- SMA clock input

- 8-pin DIP socket for auxiliary clock oscillator

## Design Trade-Offs

A few system-level design trade-offs were required in order to provide the MicroBlaze Development Kit board with the most functionality.

### Configuration Methods Galore!

A typical FPGA application uses a single non-volatile memory to store configuration images. To demonstrate new Spartan-3E capabilities, the MicroBlaze Development Kit board has three different configuration memory sources that all need to function well together. The extra configuration functions make the starter kit board more complex than typical Spartan-3E applications.

The starter kit board also includes an on-board USB-based JTAG programming interface. The on-chip circuitry simplifies the device programming experience. In typical applications, the JTAG programming hardware resides off-board or in a separate programming module, such as the Xilinx Platform USB cable. This USB port is for programming only and can not be used as an independent USB interface.

### Voltages for all Applications

The MicroBlaze Development Kit board showcases a triple-output regulator developed by Texas Instruments, the [TPS75003](#) specifically to power Spartan-3 and Spartan-3E FPGAs. This regulator is sufficient for most stand-alone FPGA applications. However, the starter kit board includes DDR SDRAM, which requires its own high-current supply. Similarly, the USB-based JTAG download solution requires a separate 1.8V supply.

## Related Resources

- Xilinx MicroBlaze Soft Processor  
<http://www.xilinx.com/microblaze>
- Xilinx PicoBlaze Soft Processor  
<http://www.xilinx.com/picoblaze>
- Xilinx Embedded Development Kit  
[http://www.xilinx.com/ise/embedded\\_design\\_prod/platform\\_studio.htm](http://www.xilinx.com/ise/embedded_design_prod/platform_studio.htm)
- Xilinx software tutorials  
<http://www.xilinx.com/support/techsup/tutorials/>
- Texas Instruments TPS75003  
<http://focus.ti.com/docs/prod/folders/print/tps75003.html>



## Switches, Buttons, and Knob

### Slide Switches

#### Locations and Labels

The MicroBlaze Development Kit board has four slide switches, as shown in [Figure 2-1](#). The slide switches are located in the lower right corner of the board and are labeled SW3 through SW0. Switch SW3 is the left-most switch, and SW0 is the right-most switch.

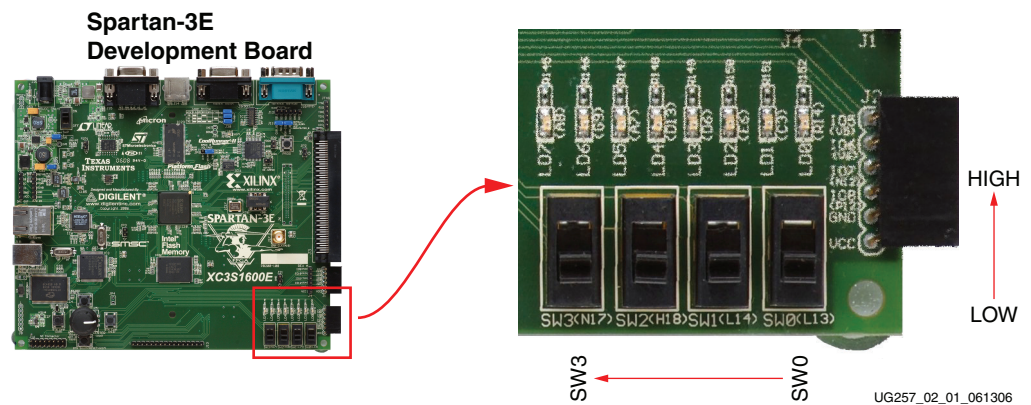


Figure 2-1: Four Slide Switches

#### Operation

When in the UP or ON position, a switch connects the FPGA pin to 3.3V, a logic High. When DOWN or in the OFF position, the switch connects the FPGA pin to ground, a logic Low. The switches typically exhibit about 2 ms of mechanical bounce and there is no active debouncing circuitry, although such circuitry could easily be added to the FPGA design programmed on the board.

#### UCF Location Constraints

[Figure 2-2](#) provides the UCF constraints for the four slide switches, including the I/O pin assignment and the I/O standard used. The PULLUP resistor is not required, but it defines the input value when the switch is in the middle of a transition.

```

NET "SW<0>" LOC = "L13" | IOSTANDARD = LVTTTL | PULLUP ;
NET "SW<1>" LOC = "L14" | IOSTANDARD = LVTTTL | PULLUP ;
NET "SW<2>" LOC = "H18" | IOSTANDARD = LVTTTL | PULLUP ;
NET "SW<3>" LOC = "N17" | IOSTANDARD = LVTTTL | PULLUP ;

```

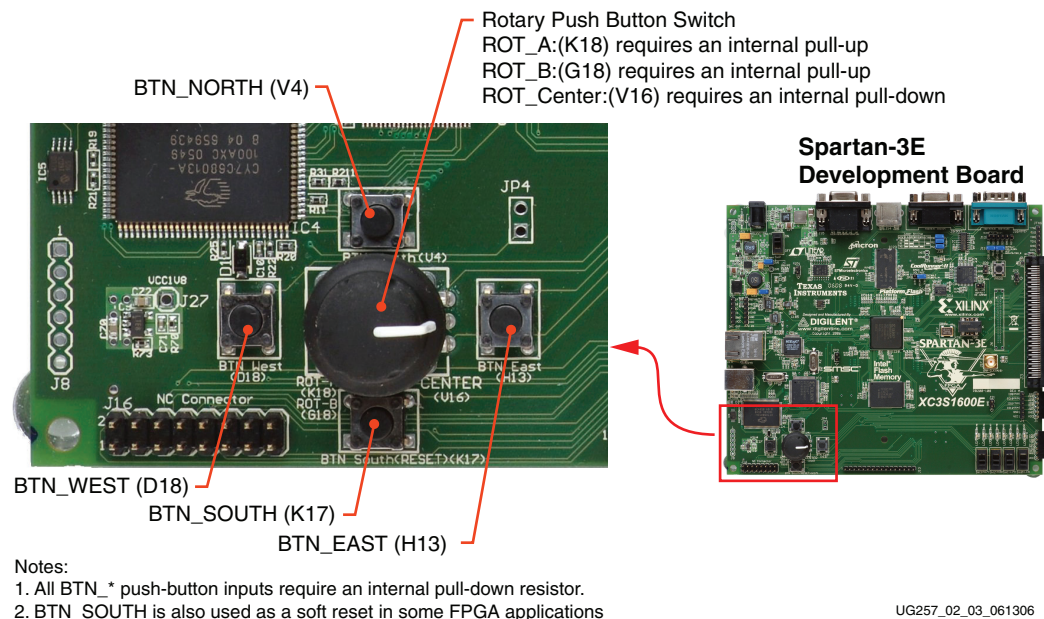
UG257\_02\_060206

Figure 2-2: UCF Constraints for Slide Switches

## Push-Button Switches

### Locations and Labels

The MicroBlaze Development Kit board has four momentary-contact push-button switches, shown in Figure 2-3. The push buttons are located in the lower left corner of the board and are labeled BTN\_NORTH, BTN\_EAST, BTN\_SOUTH, and BTN\_WEST. The FPGA pins that connect to the push buttons appear in parentheses in Figure 2-3 and the associated UCF appears in Figure 2-5.

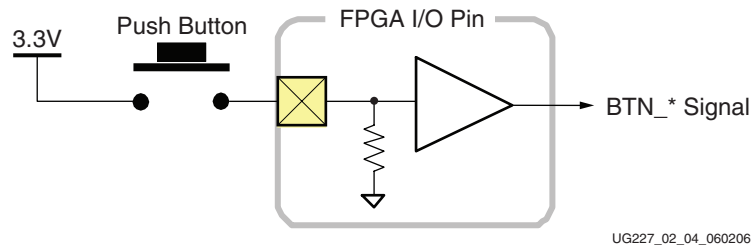


UG257\_02\_03\_061306

Figure 2-3: Four Push-Button Switches Surround Rotary Push-Button Switch

### Operation

Pressing a push button connects the associated FPGA pin to 3.3V, as shown in Figure 2-4. Use an internal pull-down resistor within the FPGA pin to generate a logic Low when the button is not pressed. Figure 2-5 shows how to specify a pull-down resistor within the UCF. There is no active debouncing circuitry on the push button.



**Figure 2-4: Push-Button Switches Require an Internal Pull-Down Resistor in FPGA Input Pin**

In some applications, the BTN\_SOUTH push-button switch is also a soft reset that selectively resets functions within the FPGA.

## UCF Location Constraints

Figure 2-5 provides the UCF constraints for the four push-button switches, including the I/O pin assignment and the I/O standard used, and defines a pull-down resistor on each input.

NET "BTN_EAST"	LOC = "H13"	IOSTANDARD = LVTTTL	PULLDOWN ;
NET "BTN_NORTH"	LOC = "V4"	IOSTANDARD = LVTTTL	PULLDOWN ;
NET "BTN_SOUTH"	LOC = "K17"	IOSTANDARD = LVTTTL	PULLDOWN ;
NET "BTN_WEST"	LOC = "D18"	IOSTANDARD = LVTTTL	PULLDOWN ;

UG257\_02\_05\_060206

**Figure 2-5: UCF Constraints for Push-Button Switches**

## Rotary Push-Button Switch

### Locations and Labels

The rotary push-button switch is located in the center of the four individual push-button switches, as shown in Figure 2-3. The switch produces three outputs. The two shaft encoder outputs are ROT\_A and ROT\_B. The center push-button switch is ROT\_CENTER.

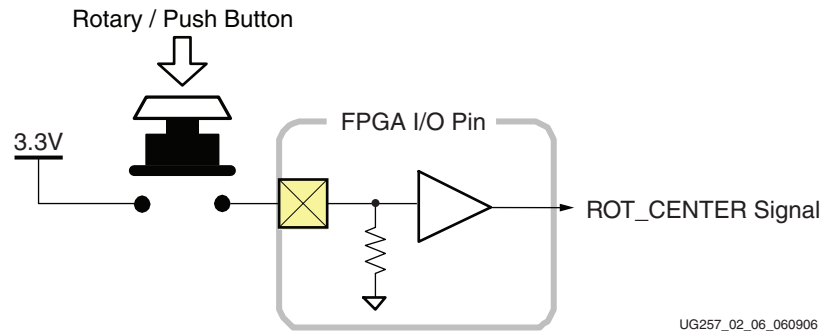
### Operation

The rotary push-button switch integrates two different functions. The switch shaft rotates and outputs values whenever the shaft turns. The shaft can also be pressed, acting as a push-button switch.

### Push-Button Switch

Pressing the knob on the rotary/push-button switch connects the associated FPGA pin to 3.3V, as shown in Figure 2-6. Use an internal pull-down resistor within the FPGA pin to generate a logic Low. Figure 2-9 shows how to specify a pull-down resistor within the UCF. There is no active debouncing circuitry on the push button.



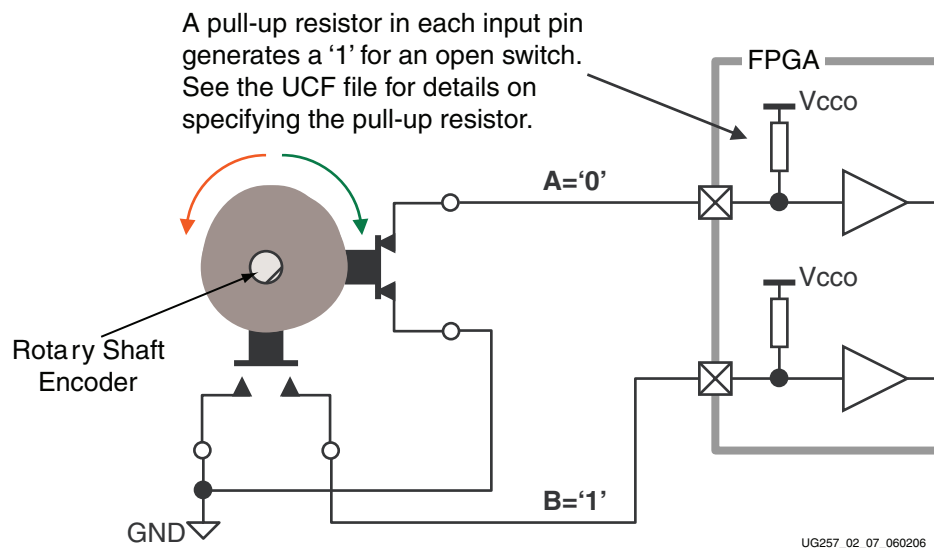


UG257\_02\_06\_060906

Figure 2-6: Push-Button Switches Require Internal Pull-up Resistor in FPGA Input Pin

## Rotary Shaft Encoder

In principal, the rotary shaft encoder behaves much like a cam, connected to central shaft. Rotating the shaft then operates two push-button switches, as shown in [Figure 2-7](#). Depending on which way the shaft is rotated, one of the switches opens before the other. Likewise, as the rotation continues, one switch closes before the other. However, when the shaft is stationary, also called the *detent* position, both switches are closed.



UG257\_02\_07\_060206

Figure 2-7: Basic example of rotary shaft encoder circuitry

Closing a switch connects it to ground, generating a logic Low. When the switch is open, a pull-up resistor within the FPGA pin pulls the signal to a logic High. The UCF constraints in [Figure 2-9](#) describe how to define the pull-up resistor.

The FPGA circuitry to decode the 'A' and 'B' inputs is simple, but must consider the mechanical switching noise on the inputs, also called chatter. As shown in [Figure 2-8](#), the chatter can falsely indicate extra rotation events or even indicate rotations in the opposite direction! See the Rotary Encoder Interface reference design in "[Related Resources](#)" for an example.

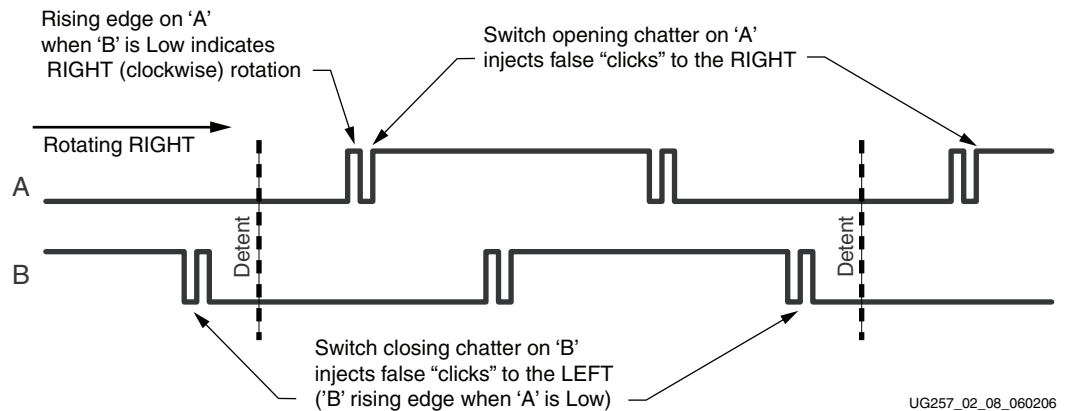


Figure 2-8: Outputs from Rotary Shaft Encoder May Include Mechanical Chatter

## UCF Location Constraints

Figure 2-9 provides the UCF constraints for the four push-button switches, including the I/O pin assignment and the I/O standard used, and defines a pull-down resistor on each input.

```

NET "ROT_A"      LOC = "K18" | IOSTANDARD = LVTTTL | PULLUP ;
NET "ROT_B"      LOC = "G18" | IOSTANDARD = LVTTTL | PULLUP ;
NET "ROT_CENTER" LOC = "V16" | IOSTANDARD = LVTTTL | PULLDOWN ;
    
```

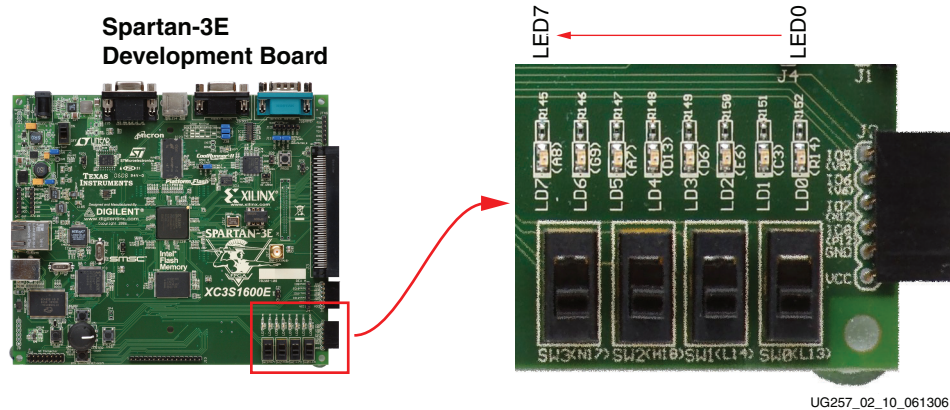
UG257\_03\_060206

Figure 2-9: UCF Constraints for Rotary Push-Button Switch

## Discrete LEDs

### Locations and Labels

The MicroBlaze Development Kit board has eight individual surface-mount LEDs located above the slide switches as shown in Figure 2-10. The LEDs are labeled LED7 through LED0. LED7 is the left-most LED, LED0 the right-most LED.



UG257\_02\_10\_061306

Figure 2-10: Eight Discrete LEDs

## Operation

Each LED has one side connected to ground and the other side connected to a pin on the Spartan-3E device via a 390Ω current limiting resistor. To light an individual LED, drive the associated FPGA control signal High.

## UCF Location Constraints

Figure 2-11 provides the UCF constraints for the four push-button switches, including the I/O pin assignment, the I/O standard used, the output slew rate, and the output drive current.

```

NET "LED<7>" LOC = "A8" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LED<6>" LOC = "G9" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LED<5>" LOC = "A7" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LED<4>" LOC = "D13" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LED<3>" LOC = "E6" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LED<2>" LOC = "D6" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LED<1>" LOC = "C3" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LED<0>" LOC = "D4" | IOSTANDARD = SSTL2_I ;

```

UG257\_02\_11\_062106

Figure 2-11: UCF Constraints for Eight Discrete LEDs

## Related Resources

- Rotary Encoder Interface for Spartan-3E Starter Kit (Reference Design)  
<http://www.xilinx.com/s3estarter>  
<http://www.xilinx.com/sp3e1600E>

## Clock Sources

### Overview

As shown in [Figure 3-1](#), the MicroBlaze Development Kit board supports three primary clock input sources, all of which are located below the Xilinx logo, near the Spartan-3E logo.

- The board includes an on-board 50 MHz clock oscillator.
- The user clock socket is populated with a 66 MHz oscillator
- Clocks can be supplied off-board via an SMA-style connector. Alternatively, the FPGA can generate clock signals or other high-speed signals on the SMA-style connector.
- Optionally install a separate 8-pin DIP-style clock oscillator in the supplied socket

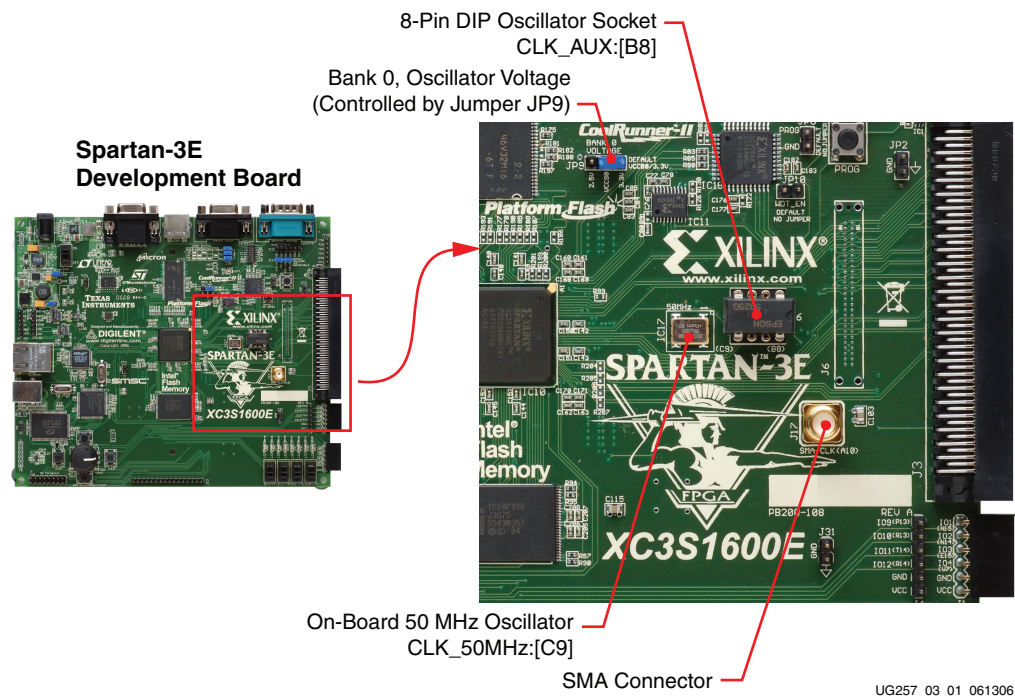


Figure 3-1: Available Clock Inputs

UG257\_03\_01\_061306

## Clock Connections

Each of the clock inputs connect directly to a global buffer input in I/O Bank 0, along the top of the FPGA. As shown in [Table 3-1](#), each of the clock inputs also optimally connects to an associated DCM.

*Table 3-1: Clock Inputs and Associated Global Buffers and DCMs*

Clock Input	FPGA Pin	Global Buffer	Associated DCM
CLK_50MHZ	C9	GCLK10	DCM_X0Y1
CLK_AUX	B8	GCLK8	DCM_X0Y1
CLK_SMA	A10	GCLK7	DCM_X1Y1

## Voltage Control

The voltage for all I/O pins in FPGA I/O Bank 0 is controlled by jumper JP9. Consequently, these clock resources are also controlled by jumper JP9. By default, JP9 is set for 3.3V. The on-board oscillator is a 3.3V device and might not perform as expected when jumper JP9 is set for 2.5V.

## 50 MHz On-Board Oscillator

The board includes a 50 MHz oscillator with a 40% to 60% output duty cycle. The oscillator is accurate to  $\pm 2500$  Hz or  $\pm 50$  ppm.

## Auxiliary Clock Oscillator Socket

The provided 8-pin socket accepts clock oscillators that fit the 8-pin DIP footprint. Use this socket if the FPGA application requires a frequency other than 50 MHz. This socket is populated with a 66 MHz oscillator. This clock input is used for some of the reference designs provided with the board. Alternatively, use the FPGA's Digital Clock Manager (DCM) to generate or synthesize other frequencies from the on-board 50 MHz oscillator.

## SMA Clock Input or Output Connector

To provide a clock from an external source, connect the input clock signal to the SMA connector. The FPGA can also generate a single-ended clock output or other high-speed signal on the SMA clock connector for an external device.

## UCF Constraints

The clock input sources require two different types of constraints. The location constraints define the I/O pin assignments and I/O standards. The period constraints define the clock period—and consequently the clock frequency—and the duty cycle of the incoming clock signal.

## Location

Figure 3-2 provides the UCF constraints for the three clock input sources, including the I/O pin assignment and the I/O standard used. The settings assume that jumper JP9 is set for 3.3V. If JP9 is set for 2.5V, adjust the IOSTANDARD settings accordingly.

```
NET "CLK_50MHZ" LOC = "C9" | IOSTANDARD = LVCMOS33 ;  
NET "CLK_SMA" LOC = "A10" | IOSTANDARD = LVCMOS33 ;  
NET "CLK_AUX" LOC = "B8" | IOSTANDARD = LVCMOS33 ;
```

UG257\_03\_02\_061306

Figure 3-2: UCF Location Constraints for Clock Sources

## Clock Period Constraints

The Xilinx ISE development software uses timing-driven logic placement and routing. Set the clock PERIOD constraint as appropriate. An example constraint appears in Figure 3-3 for the on-board 50 MHz clock oscillator. The CLK\_50MHZ frequency is 50 MHz, which equates to a 20 ns period. The output duty cycle from the oscillator ranges between 40% to 60%.

```
# Define clock period for 50 MHz oscillator  
NET "CLK_50MHZ" PERIOD = 20.0ns HIGH 40%;
```

UG257\_03\_03\_060206

Figure 3-3: UCF Clock PERIOD Constraint

## Related Resources

- Epson SG-8002JF Series Oscillator Data Sheet (50 MHz Oscillator)  
[http://www.eea.epson.com/go/Prod\\_Admin/Categories/EEA/QD/Crystal\\_Oscillators/prog\\_oscillators/go/Resources/TestC2/SG8002JF](http://www.eea.epson.com/go/Prod_Admin/Categories/EEA/QD/Crystal_Oscillators/prog_oscillators/go/Resources/TestC2/SG8002JF)



# FPGA Configuration Options

---

The MicroBlaze Development Kit board supports a variety of FPGA configuration options:

- Download FPGA designs directly to the Spartan-3E FPGA via JTAG, using the on-board USB interface. The on-board USB-JTAG logic also provides in-system programming for the on-board Platform Flash PROM and the Xilinx XC2C64A CPLD. SPI serial Flash and StrataFlash programming are performed separately.
- Program the on-board 4 Mbit Xilinx XCF04S serial Platform Flash PROM, then configure the FPGA from the image stored in the Platform Flash PROM using Master Serial mode.
- Program the on-board 16 Mbit ST Microelectronics SPI serial Flash PROM, then configure the FPGA from the image stored in the SPI serial Flash PROM using SPI mode.
- Program the on-board 128 Mbit Intel StrataFlash parallel NOR Flash PROM, then configure the FPGA from the image stored in the Flash PROM using BPI Up or BPI Down configuration modes. Further, an FPGA application can dynamically load two different FPGA configurations using the Spartan-3E FPGA's MultiBoot mode. See the Spartan-3E data sheet ([DS312](#)) for additional details on the MultiBoot feature.

[Figure 4-1](#) indicates the position of the USB download/programming interface and the on-board non-volatile memories that potentially store FPGA configuration images. [Figure 4-2](#) provides additional details on configuration options.



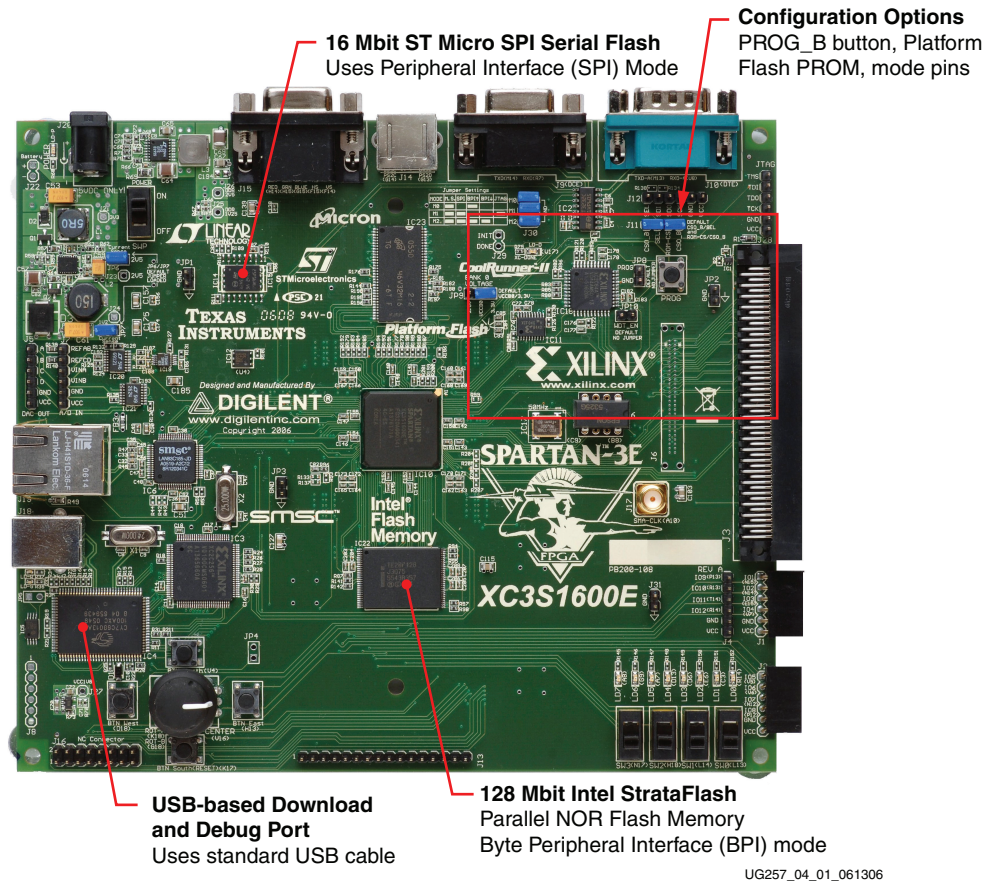


Figure 4-1: MicroBlaze Development Kit Board FPGA Configuration Options

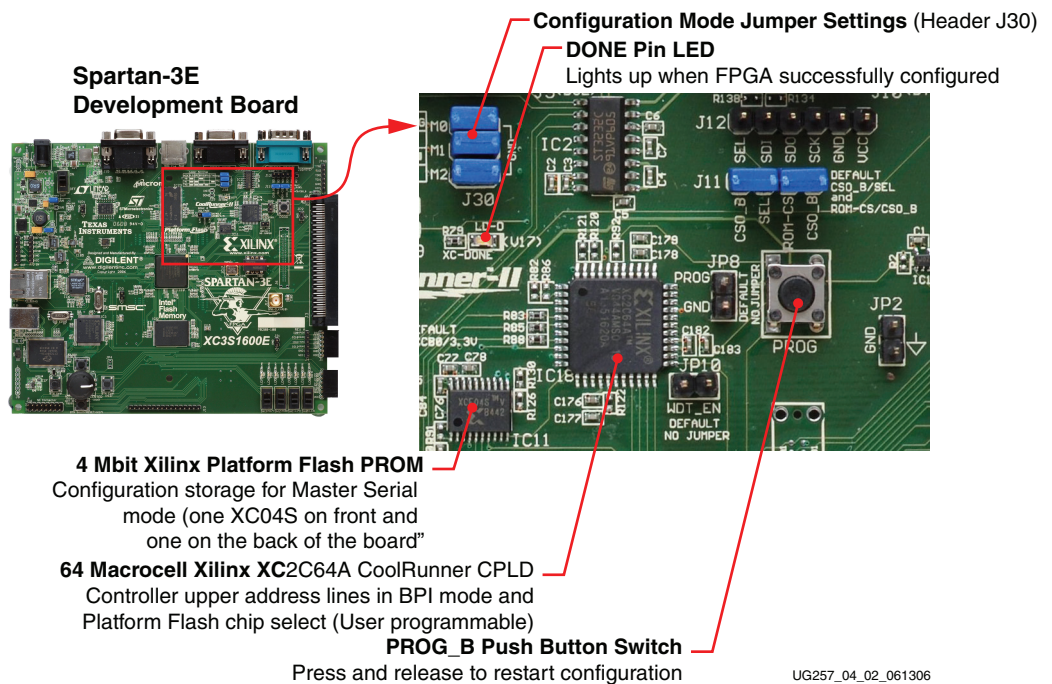


Figure 4-2: Detailed Configuration Options

The configuration mode jumpers determine which configuration mode the FPGA uses when power is first applied, or whenever the PROG button is pressed.

The DONE pin LED lights when the FPGA successfully finishes configuration.

Pressing the PROG button forces the FPGA to restart its configuration process.

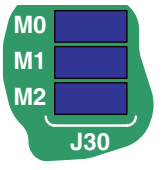
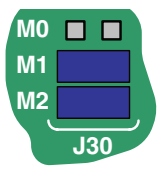
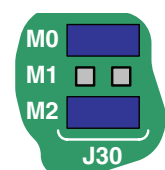
The 4 Mbit Xilinx Platform Flash PROM provides easy, JTAG-programmable configuration storage for the FPGA. The FPGA configures from the Platform Flash using Master Serial mode.

The 64-macrocell XC2C64A CoolRunner II CPLD provides additional programming capabilities and flexibility when using the BPI Up, BPI Down, or MultiBoot configuration modes and loading the FPGA from the StrataFlash parallel Flash PROM. The CPLD is user-programmable.

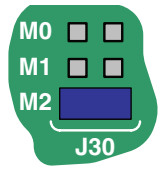
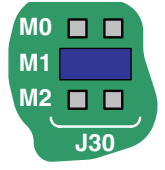
## Configuration Mode Jumpers

As shown in [Table 4-1](#), the J30 jumper block settings control the FPGA’s configuration mode. Inserting a jumper grounds the associated mode pin. Insert or remove individual jumpers to select the FPGA’s configuration mode and associated configuration memory source.

**Table 4-1: MicroBlaze Development Kit Board Configuration Mode Jumper Settings (Header J30 in [Figure 4-2](#))**

Configuration Mode	Mode Pins M2:M1:M0	FPGA Configuration Image Source	Jumper Settings
Master Serial	000	Platform Flash PROM	
SPI (see <a href="#">Chapter 12, “SPI Serial Flash”</a> )	001	SPI Serial Flash PROM starting at address 0	
BPI Up (see <a href="#">Chapter 11, “Intel StrataFlash Parallel NOR Flash PROM”</a> )	010	StrataFlash parallel Flash PROM, starting at address 0 and incrementing through address space. The CPLD controls address lines A[24:20] during BPI configuration.	

**Table 4-1: MicroBlaze Development Kit Board Configuration Mode Jumper Settings (Header J30 in Figure 4-2)**

Configuration Mode	Mode Pins M2:M1:M0	FPGA Configuration Image Source	Jumper Settings
BPI Down (see <a href="#">Chapter 11</a> , “Intel StrataFlash Parallel NOR Flash PROM”)	011	StrataFlash parallel Flash PROM, starting at address 0x1FF_FFFF and decrementing through address space. The CPLD controls address lines A[24:20] during BPI configuration.	
JTAG	101	Downloaded from host via USB-JTAG port	

## PROG Push Button

The PROG push button, shown in [Figure 4-2, page 24](#), forces the FPGA to reconfigure from the selected configuration memory source. Press and release this button to restart the FPGA configuration process at any time.

## DONE Pin LED

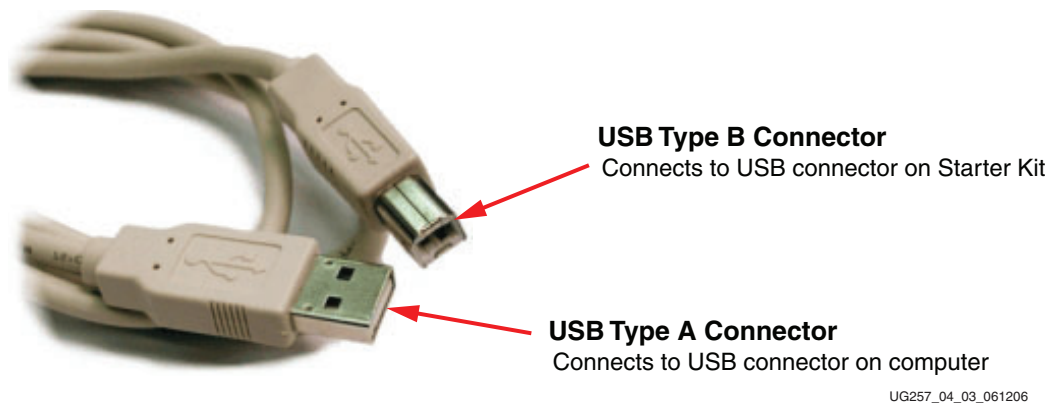
The DONE pin LED, shown in [Figure 4-2, page 24](#), lights whenever the FPGA is successfully configured. If this LED is not lit, then the FPGA is not configured.

## Programming the FPGA, CPLD, or Platform Flash PROM via USB

As shown in [Figure 4-1, page 24](#), the MicroBlaze Development Kit board includes embedded USB-based programming logic and an USB endpoint with a Type B connector. Via a USB cable connection with the host PC, the iMPACT programming software directly programs the FPGA, the Platform Flash PROM, or the on-board CPLD. Direct programming of the parallel or serial Flash PROMs is not presently supported.

### Connecting the USB Cable

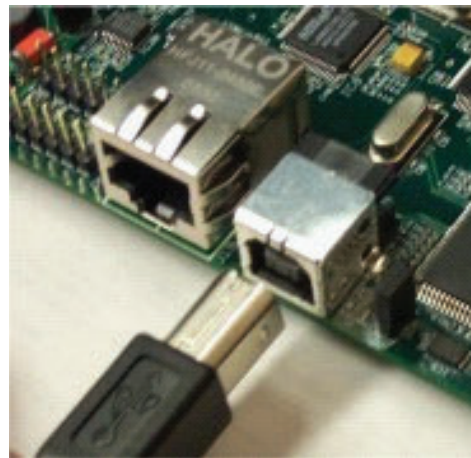
The kit includes a standard USB Type A/Type B cable, similar to the one shown in [Figure 4-3](#). The actual cable color might vary from the picture.



*Figure 4-3: Standard USB Type A/Type B Cable*

The wider and narrower Type A connector fits the USB connector at the back of the computer.

After installing the Xilinx software, connect the square Type B connector to the MicroBlaze Development Kit board, as shown in [Figure 4-4](#). The USB connector is on the left side of the board, immediately next to the Ethernet connector. When the board is powered on, the Windows operating system should recognize and install the associated driver software.



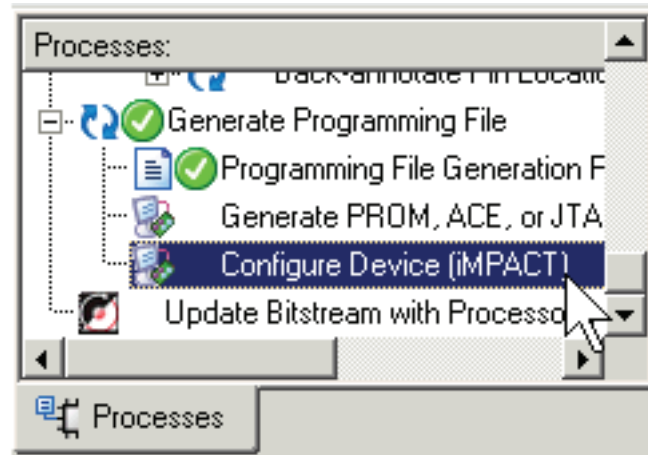
*Figure 4-4: Connect the USB Type B Connector to the MicroBlaze Development Kit Board Connector*

When the USB cable driver is successfully installed and the board is correctly connected to the PC, a green LED lights up, indicating a good connection.

## Programming via iMPACT

After successfully compiling an FPGA design using the Xilinx development software, the design can be downloaded using the iMPACT programming software and the USB cable.

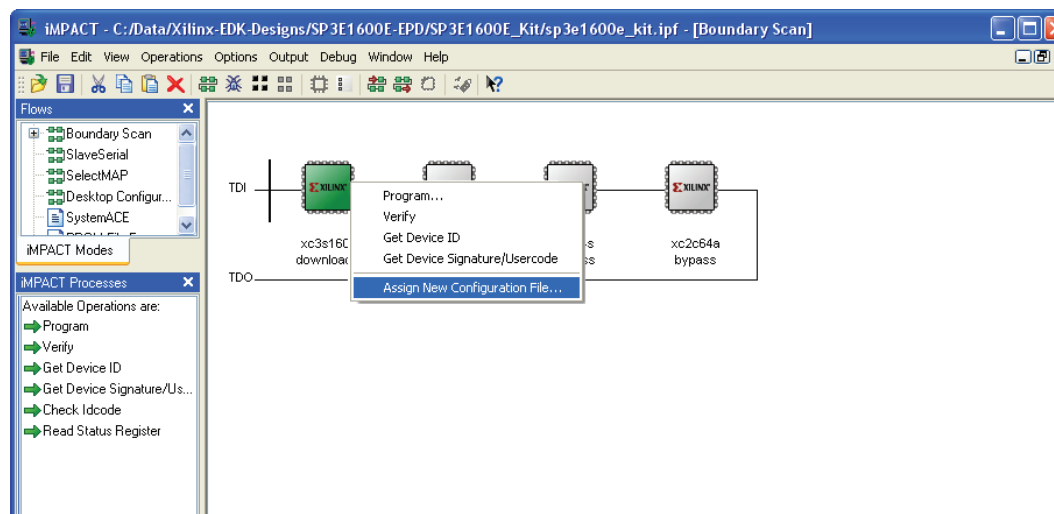
To begin programming, connect the USB cable to the starter kit board and apply power to the board. Then, double-click **Configure Device (iMPACT)** from within Project Navigator, as shown in Figure 4-5.



UG257\_04\_05\_061206

Figure 4-5: Double-Click to Invoke iMPACT

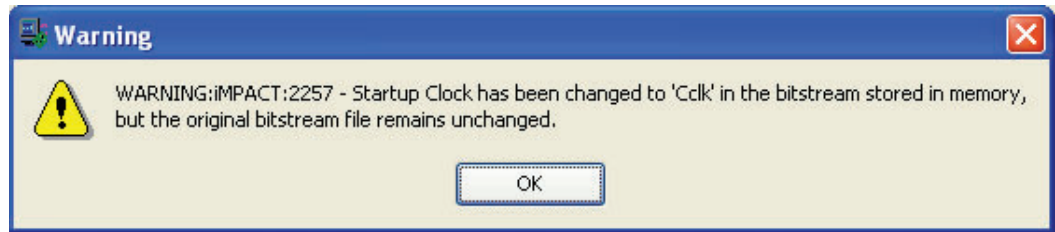
If the board is connected properly, the iMPACT programming software automatically recognizes the three devices in the JTAG programming file, as shown in Figure 4-6. If not already prompted, click the first device in the chain, the Spartan-3E FPGA, to highlight it. Right-click the FPGA and select **Assign New Configuration File**. Select the desired FPGA configuration file and click **OK**.



UG257\_04\_06\_06121

Figure 4-6: Right-Click to Assign a Configuration File to the Spartan-3E FPGA

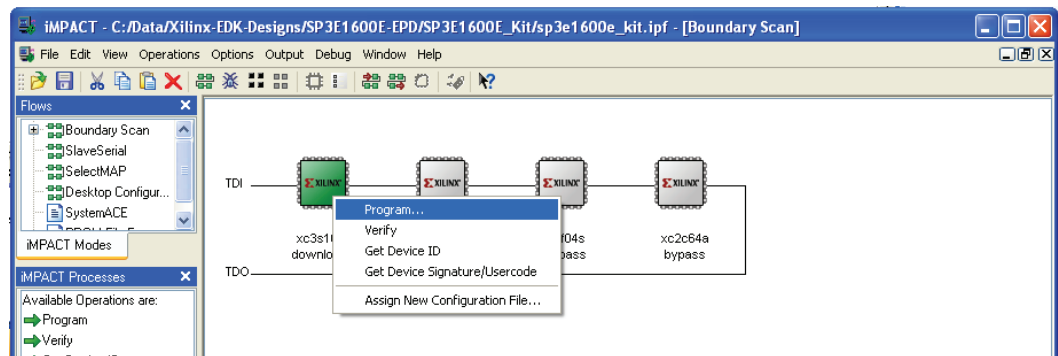
If the original FPGA configuration file used the default StartUp clock source, CCLK, iMPACT issues the warning message shown in Figure 4-7. This message can be safely ignored. When downloading via JTAG, the iMPACT software must change the StartUP clock source to use the TCK JTAG clock source.



UG257\_04-07\_06906

Figure 4-7: iMPACT Issues a Warning if the StartUp Clock Was Not CCLK

To start programming the FPGA, right-click the FPGA and select **Program**. The iMPACT software reports status during programming process. Direct programming to the FPGA takes a few seconds to less than a minute, depending on the speed of the PC’s USB port and the iMPACT settings.



UG257\_04\_08\_061206

Figure 4-8: Right-Click to Program the Spartan-3E FPGA

When the FPGA successfully programs, the iMPACT software indicates success, as shown in Figure 4-9. The FPGA application is now executing on the board and the DONE pin LED (see Figure 4-2) lights up.

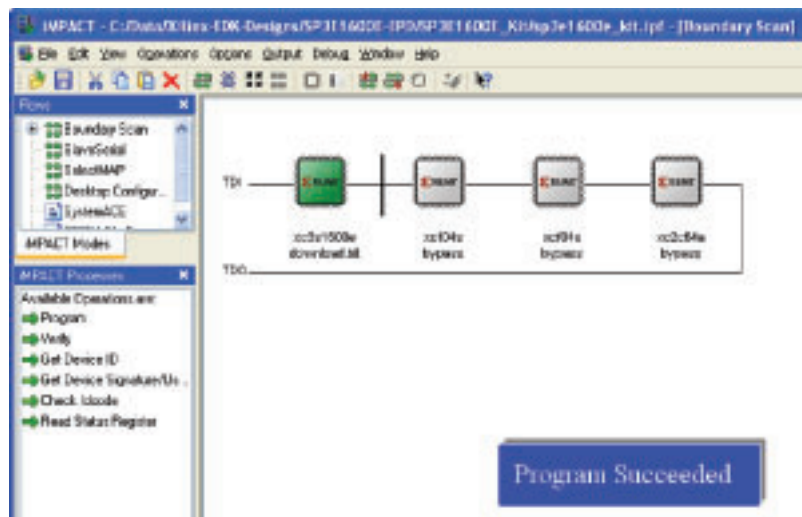


Figure 4-9: **IMPACT Programming Succeeded, the FPGA's DONE Pin is High**

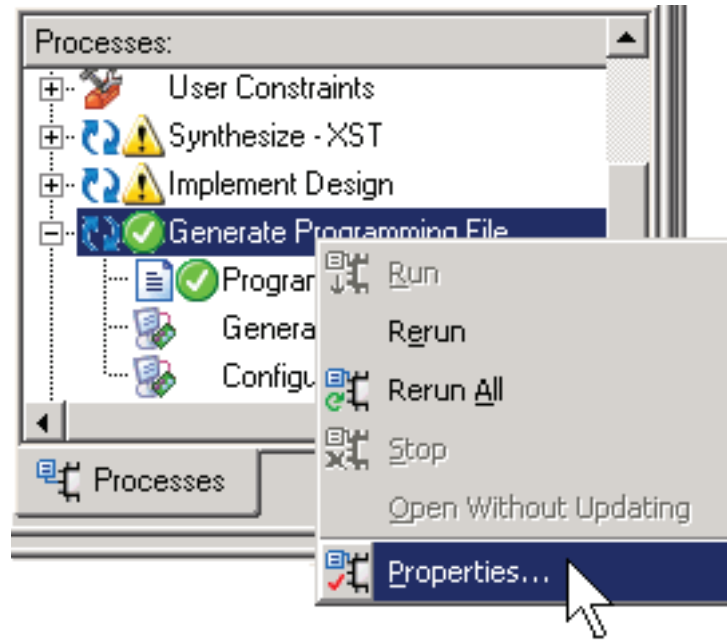
## Programming Platform Flash PROM via USB

The on-board USB-JTAG circuitry also programs the two Xilinx XCF04S serial Platform Flash PROM. The steps provided in this section describe how to set up the PROM file and how to download it to the board to ultimately program the FPGA.

### Generating the FPGA Configuration Bitstream File

Before generating the PROM file, create the FPGA bitstream file. The FPGA provides an output clock, CCLK, when loading itself from an external PROM. The FPGA's internal CCLK oscillator always starts at its slowest setting, approximately 1.5 MHz. Most external PROMs support a higher frequency. Increase the CCLK frequency as appropriate to reduce the FPGA's configuration time. The Xilinx XCF04S Platform Flash supports a 25 MHz CCLK frequency.

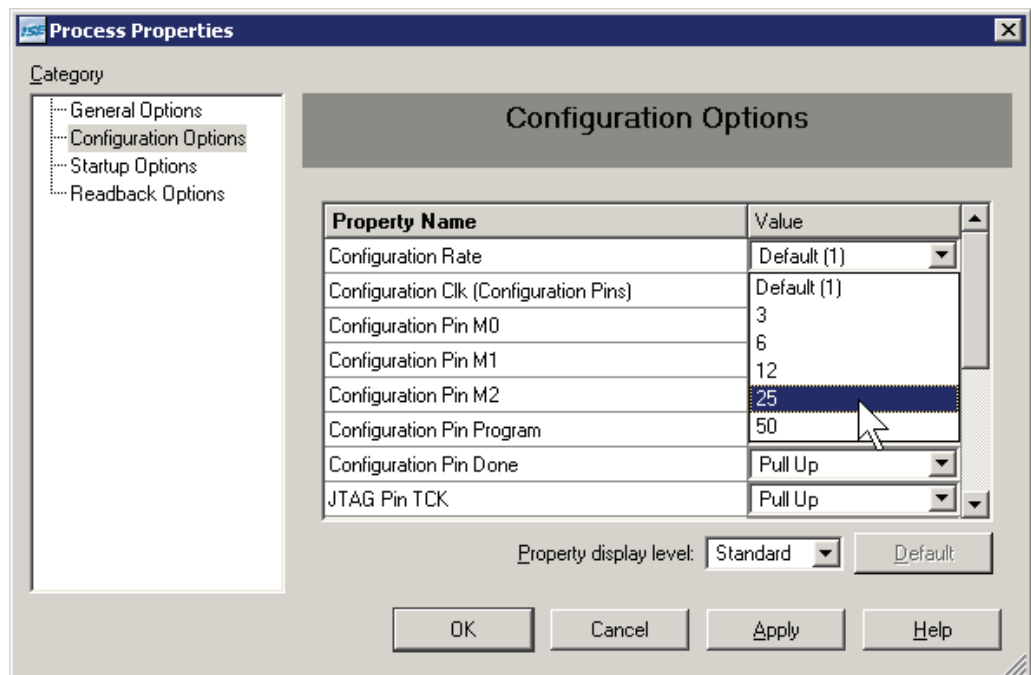
Right-click **Generator Programming File** in the Processes pane, as shown in [Figure 4-10](#). Left-click **Properties**.



UG257\_04\_10\_061206

Figure 4-10: Set Properties for Bitstream Generator

Click **Configuration Options** as shown in Figure 4-11. Using the **Configuration Rate** drop list, choose **25** to increase the internal CCLK oscillator to approximately 25 MHz, the fastest frequency when using an XCF04S Platform Flash PROM. Click **OK** when finished.

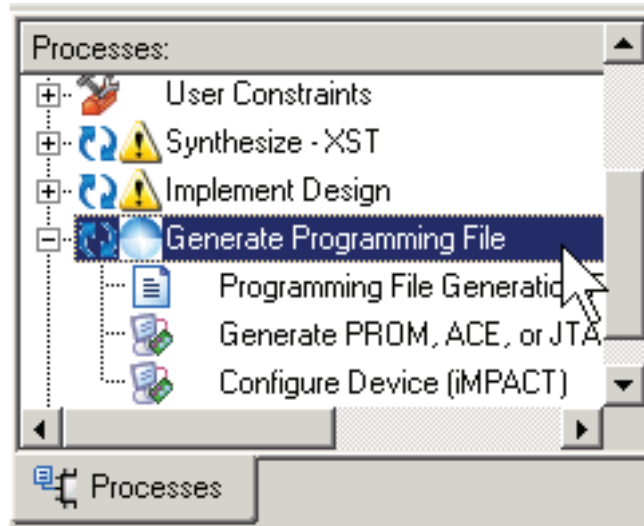


UG257\_04\_11\_061206

Figure 4-11: Set CCLK Configuration Rate under Configuration Options



To regenerate the programming file, double-click **Generate Programming File**, as shown in [Figure 4-12](#).

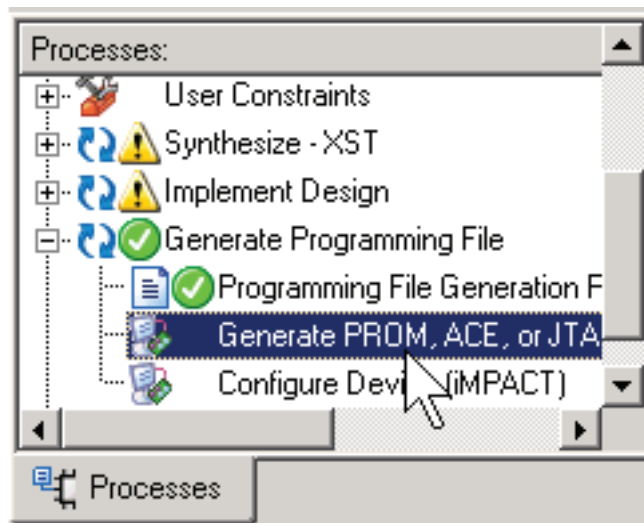


UG257\_04\_12\_022706

*Figure 4-12: Double-Click Generate Programming File*

### Generating the PROM File

After generating the program file, double-click **Generate PROM, ACE, or JTAG File** to launch the iMPACT software, as shown in [Figure 4-13](#).



UG257\_04\_13\_061206

*Figure 4-13: Double-Click Generate PROM, ACE, or JTAG File*

After iMPACT starts, double-click **PROM File Formatter**, as shown in [Figure 4-14](#).

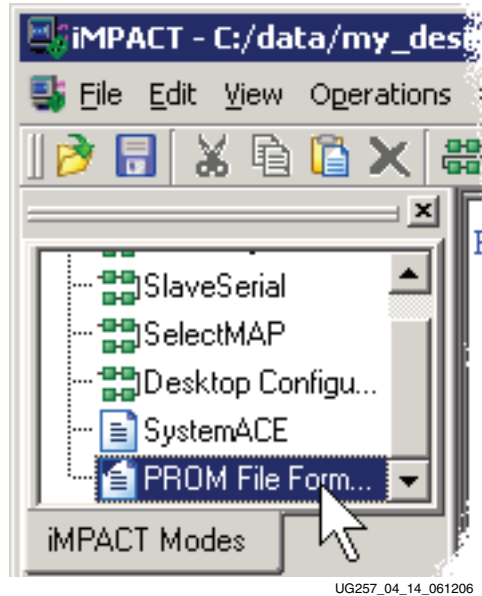


Figure 4-14: Double-Click PROM File Formatter

Choose **Xilinx PROM** as the target PROM type, as shown in Figure 4-15. Select from any of the PROM File Formats; the Intel Hex format (**MCS**) is popular. Enter the **Location** of the directory and the **PROM File Name**. Click **Next >** when finished.

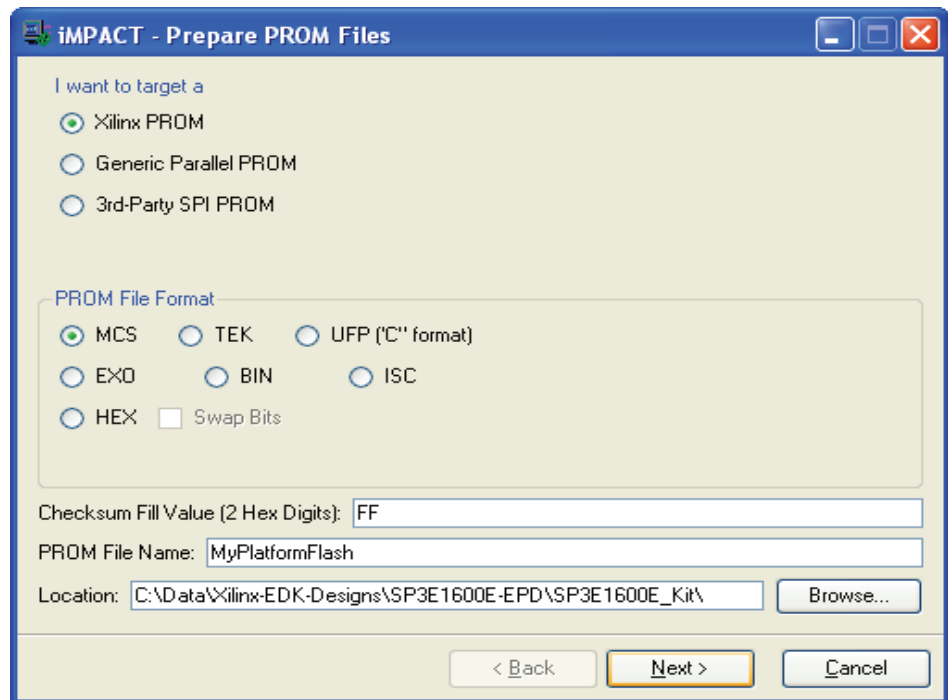
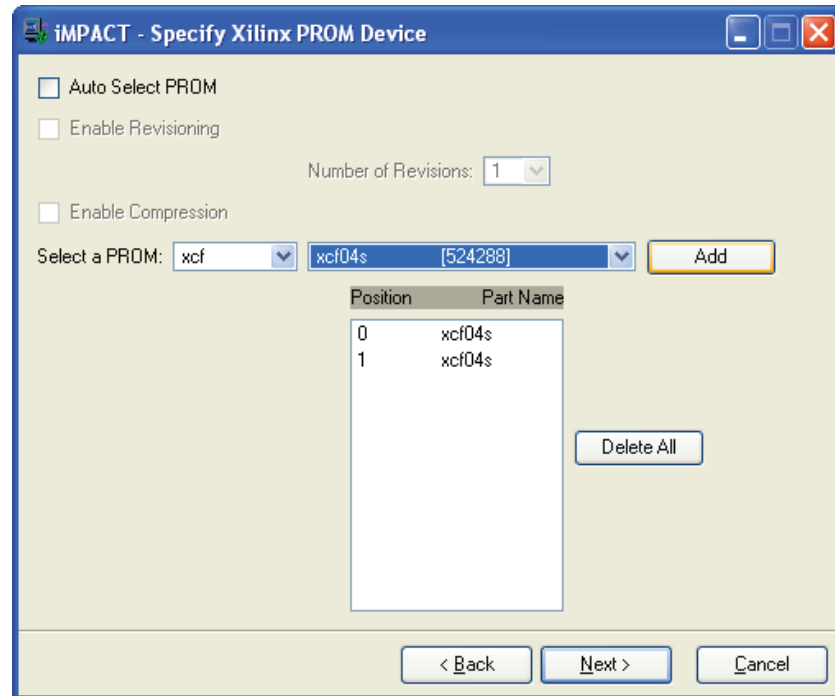


Figure 4-15: Choose the PROM Target Type, the, Data Format, and File Location

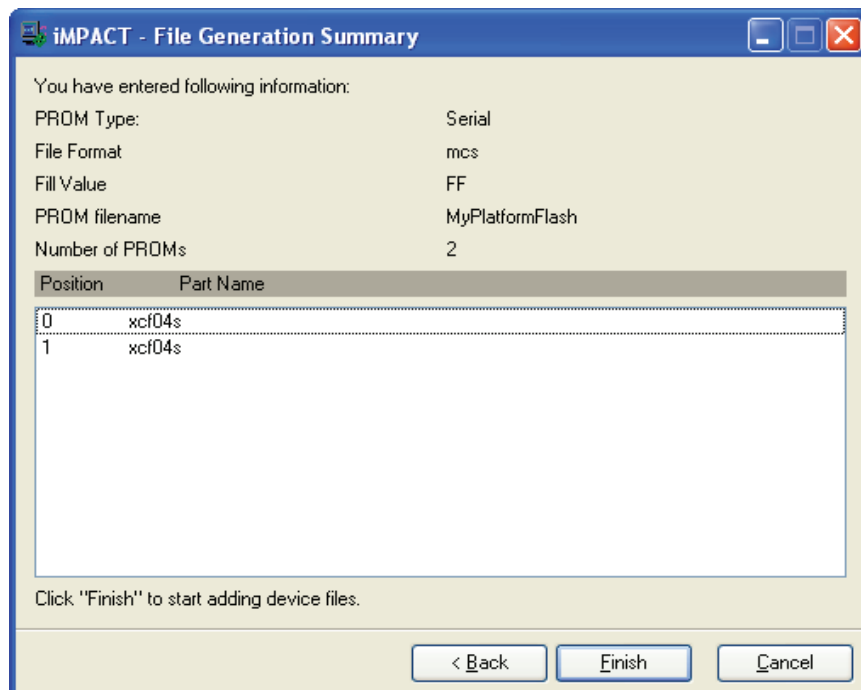
The Spartan-3E Starter Kit board has an XCF04S Platform Flash PROM. Select **xcf04s** from the drop list, as shown in [Figure 4-16](#). Click **Add**, then click **Next** >.



UG257\_4-16\_061206

*Figure 4-16: Choose the XCF04S Platform Flash PROM*

The PROM Formatter then echoes the settings, as shown in [Figure 4-17](#). Click **Finish**.



UG257\_4-17\_061206

*Figure 4-17: Click Finish after Entering PROM Formatter Settings*

The PROM Formatter then prompts for the name(s) of the FPGA configuration bitstream file. As shown in [Figure 4-18](#), click **OK** to start selecting files. Select an FPGA bitstream file (\*.bit). Choose **No** after selecting the last FPGA file. Finally, click **OK** to continue.



UG257\_4-18\_060906

*Figure 4-18: Enter FPGA Configuration Bitstream File(s)*

When PROM formatting is complete, the iMPACT software presents the present settings by showing the PROM, the select FPGA bitstream(s), and the amount of PROM space consumed by the bitstream. [Figure 4-19](#) shows an example for a single XC3S500E FPGA bitstream stored in an XCF04S Platform Flash PROM.

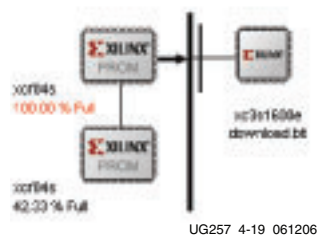


Figure 4-19: PROM Formatting Completed

To generate the actual PROM file, click **Operations** → **Generate File** as shown in Figure 4-20.

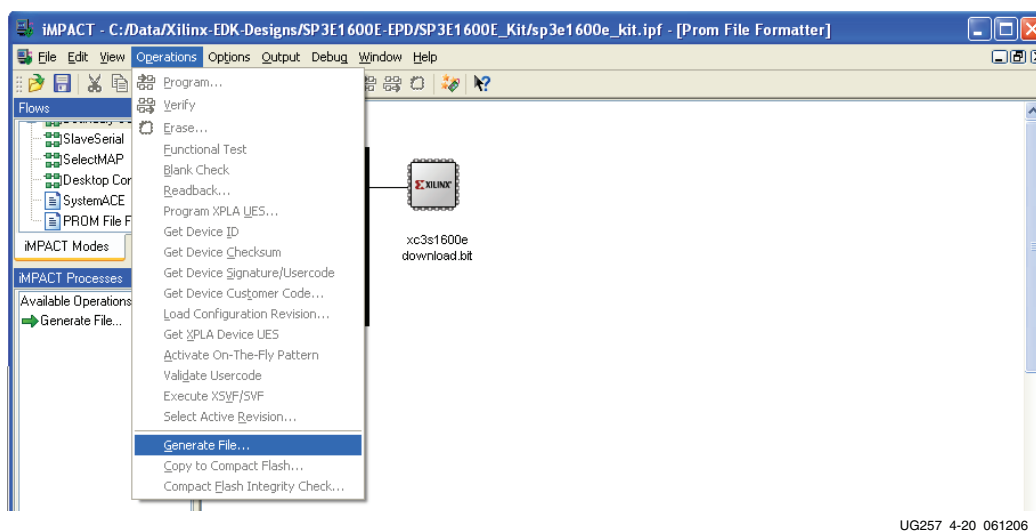


Figure 4-20: Click **Operations** → **Generate File** to Create the Formatted PROM File

The iMPACT software indicates that the PROM file was successfully created, as shown in Figure 4-21.

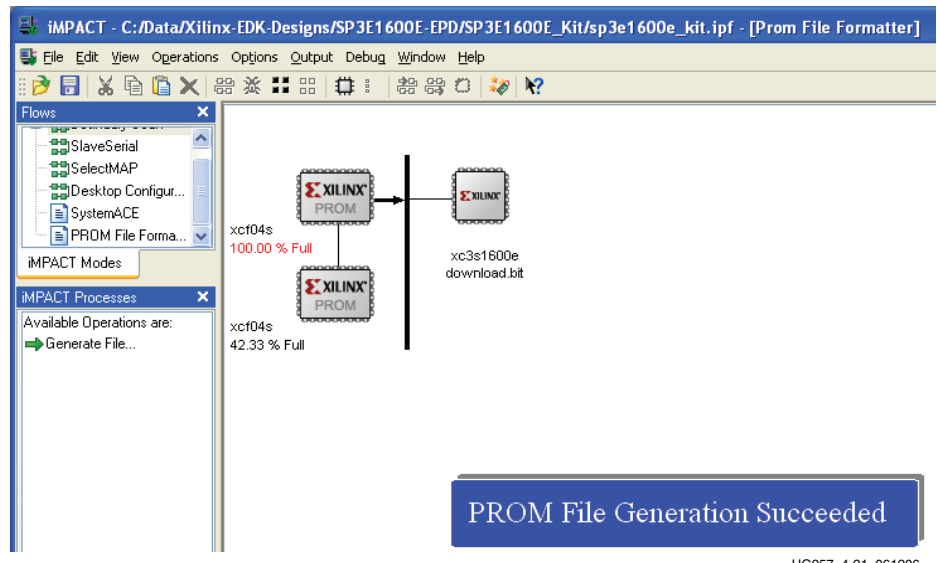


Figure 4-21: PROM File Formatter Succeeded

### Programming the Platform Flash PROM

To program the formatted PROM file into the Platform Flash PROM via the on-board USB-JTAG circuitry, follow the steps outlined in this subsection.

Place the iMPACT software in the JTAG Boundary Scan mode, either by choosing **Boundary Scan** in the iMPACT Modes pane, as shown in Figure 4-22, or by clicking on the **Boundary Scan** tab.

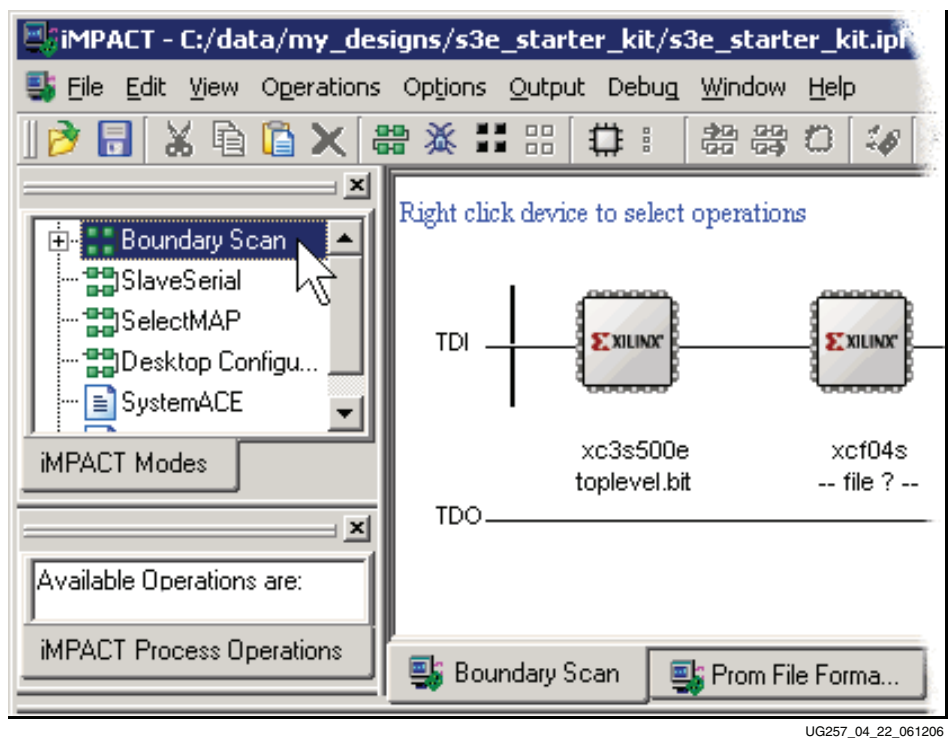
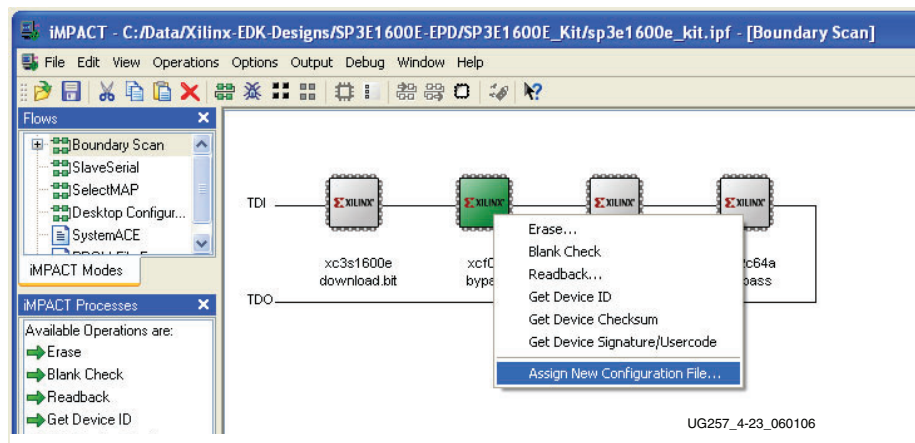


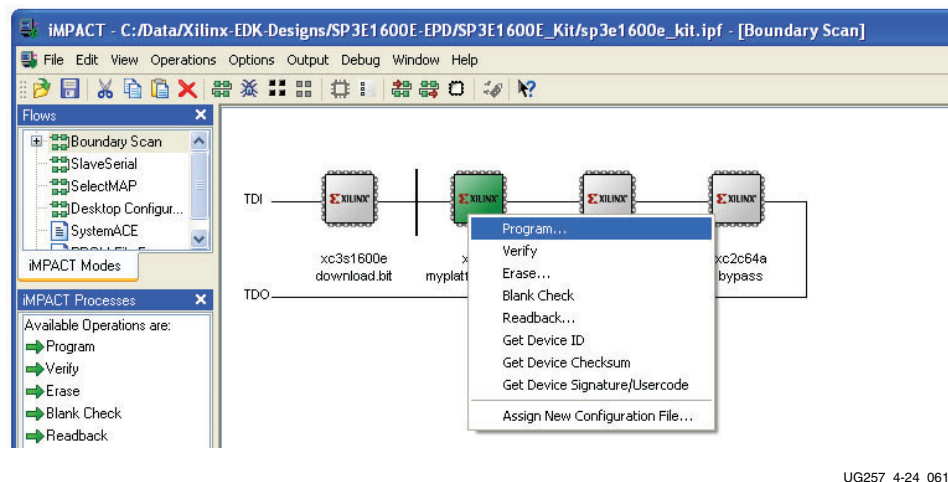
Figure 4-22: Switch to Boundary Scan Mode

Assign the PROM file to the XCF04S Platform Flash PROM on the JTAG chain, as shown in [Figure 4-23](#). Right-click the PROM icon, then click **Assign New Configuration File**. Select a previously generated PROM format file and click **OK**.



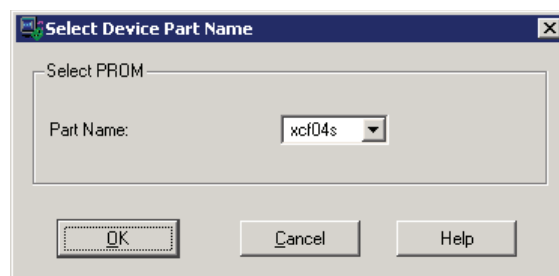
**Figure 4-23: Assign the PROM File to the XCF04S Platform Flash PROM**

To start programming the PROM, right-click the PROM icon and then click **Program**.



**Figure 4-24: Program the XCF04S Platform Flash PROM**

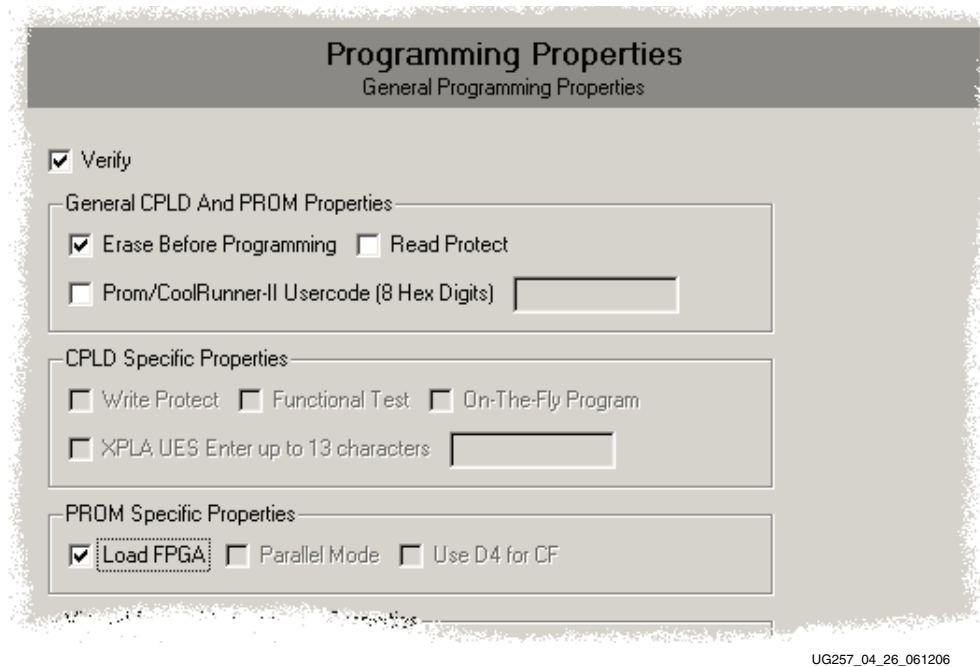
The programming software again prompts for the PROM type to be programmed. Select **xcfc04s** and click **OK**, as shown in [Figure 4-25](#).



**Figure 4-25: Select XCF04S Platform Flash PROM**

Before programming, choose the programming options available in [Figure 4-26](#). Checking the **Erase Before Programming** option erases the Platform Flash PROM completely before programming, ensuring that no previous data lingers. The **Verify** option checks that the PROM was correctly programmed and matches the downloaded configuration bitstream. Both these options are recommended even though they increase overall programming time.

The **Load FPGA** option immediately forces the FPGA to reconfigure after programming the Platform Flash PROM. The FPGA's configuration mode pins must be set for Master Serial mode, as defined in [Table 4-1, page 25](#). Click **OK** when finished.



*Figure 4-26: PROM Programming Options*

The iMPACT software indicates if programming was successful or not. If programming was successful and the Load FPGA option was left unchecked, push the PROG\_B push-button switch shown in [Figure 4-2, page 24](#) to force the FPGA to reconfigure from the newly programmed Platform Flash PROM. If the FPGA successfully configures, the DONE LED, also shown in [Figure 4-2](#), lights up.





## Character LCD Screen

### Overview

The Spartan-3E MicroBlaze Development Kit board has been designed with a 16 pin female header connector. The Spartan-3E MicroBlaze Development board is shipped with a 2x16 LCD display attached, but any standard LCD display can be attached to this connector.

The Spartan-3E MicroBlaze Development Kit board prominently features a 2-line by 16-character liquid crystal display (LCD). The FPGA controls the LCD via the 4-bit data interface or 8 bit data interface in shown [Figure 5-1](#).

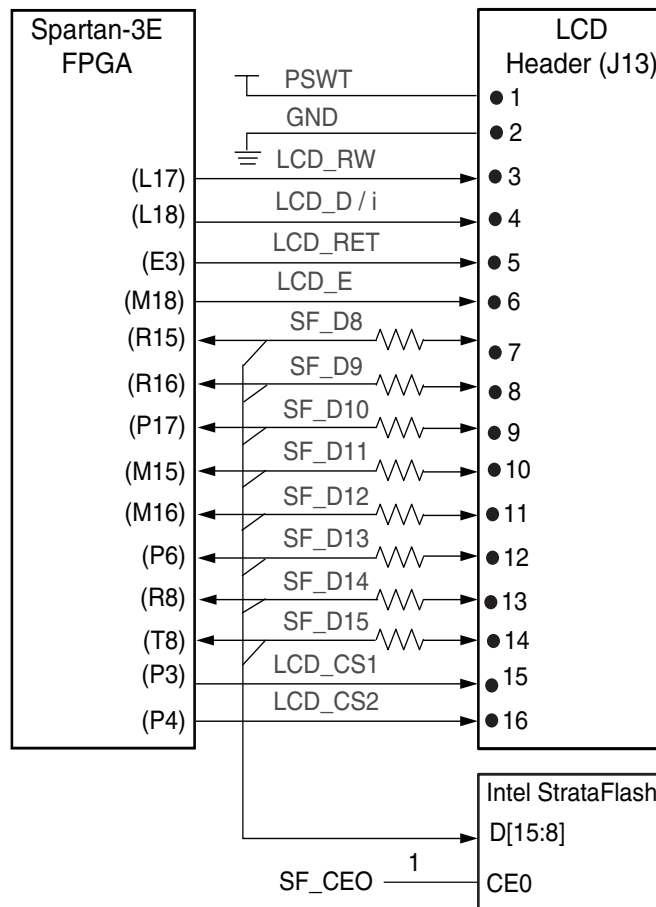


Figure 5-1: Character LCD Interface

Once mastered, the LCD is a practical way to display a variety of information using standard ASCII and custom characters. However, these displays are not fast. Scrolling the display at half-second intervals tests the practical limit for clarity. Compared with the 50 MHz clock available on the board, the display is slow. A PicoBlaze processor efficiently controls display timing plus the actual content of the display.

## Character LCD Interface Signals

Table 5-1 shows the interface character LCD interface signals.

Table 5-1: Character LCD Interface

Signal Name	FPGA Pin	Function	
SF_D<15>	T8	Data bit DB7	Shared with StrataFlash pins SF_D<15:8>
SF_D<14>	R8	Data bit DB6	
SF_D<13>	P6	Data bit DB5	
SF_D<12>	M16	Data bit DB4	
SF_D<11>	M15	Data bit DB3	
SF_D<10>	P17	Data bit DB2	
SF_D<9>	R16	Data bit DB1	
SF_D<8>	R15	Data bit DB0	
LCD_E	M18	Read/Write Enable Pulse 0: Disabled 1: Read/Write operation enabled	
LCD_RS	L18	Register Select 0: Instruction register during write operations. Busy Flash during read operations 1: Data for read or write operations	
LCD_RW	L17	Read/Write Control 0: WRITE, LCD accepts data 1: READ, LCD presents data	
LCD_RET	E3		
LCD_CS1	P3		
LCD_CS2	P4		

## Voltage Compatibility

The character LCD is powered by +5V. The FPGA I/O signals are powered by 3.3V. However, the FPGA's output levels are recognized as valid Low or High logic levels by the LCD. The LCD controller accepts 5V TTL signal levels and the 3.3V LVCMOS outputs provided by the FPGA meet the 5V TTL voltage level requirements.

The 390Ω series resistors on the data lines prevent overstressing on the FPGA and StrataFlash I/O pins when the character LCD drives a High logic value. The character LCD drives the data lines when LCD\_RW is High. Most applications treat the LCD as a write-only peripheral and never read from from the display.

## Interaction with Intel StrataFlash

As shown in Figure 5-1, the four LCD data signals are also shared with StrataFlash data lines SF\_D<11:8>. As shown in Table 5-2, the LCD/StrataFlash interaction depends on the application usage in the design. When the StrataFlash memory is disabled (SF\_CE0 = High), then the FPGA application has full read/write access to the LCD. Conversely, when LCD read operations are disabled (LCD\_RW = Low), then the FPGA application has full read/write access to the StrataFlash memory

Table 5-2: LCD/StrataFlash Control Interaction

SF_CE0	SF_BYTE	LCD_RW	Operation
1	X	X	StrataFlash disabled. Full read/write access to LCD.
X	X	0	LCD write access only. Full access to StrataFlash.
X	0	X	StrataFlash in byte-wide (x8) mode. Upper address lines are not used. Full access to both LCD and StrataFlash.

**Notes:**

1. 'X' indicates a don't care, can be either 0 or 1.

If the StrataFlash memory is in byte-wide (x8) mode (SF\_BYTE = Low), the FPGA application has full simultaneous read/write access to both the LCD and the StrataFlash memory. In byte-wide mode, the StrataFlash memory does not use the SF\_D<15:8> data lines.

## UCF Location Constraints

Figure 5-2 provides the UCF constraints for the Character LCD, including the I/O pin assignment and the I/O standard used.

```
# ==== Character LCD (LCD) ====
NET "LCD_E" LOC = "M18" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "LCD_DI" LOC = "L18" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "LCD_RW" LOC = "L17" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "LCD_RET" LOC = "E3" | IOSTANDARD = SSTL2_I ;
NET "LCD_CS1" LOC = "P3" | IOSTANDARD = SSTL2_I ;
NET "LCD_CS2" LOC = "P4" | IOSTANDARD = SSTL2_I ;

# LCD data connections are shared with StrataFlash connections SF_D<15:8>
NET "SF_D<8>" LOC = "R15" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_D<9>" LOC = "R16" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_D<10>" LOC = "P17" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_D<11>" LOC = "M15" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_D<12>" LOC = "M16" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_D<13>" LOC = "P6" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_D<14>" LOC = "R8" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_D<15>" LOC = "T8" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
```

UG257\_05\_02\_061306

Figure 5-2: UCF Location Constraints for the Character LCD

## LCD Controller

The 2 x 16 character LCD has an internal Sitronix [ST7066U](#) graphics controller that is functionally equivalent with the following devices.

- Samsung [S6A0069X](#) or KS0066U
- Hitachi HD44780
- SMOS SED1278

## Memory Map

The controller has three internal memory regions, each with a specific purpose. The display must be initialized before accessing any of these memory regions.

### DD RAM

The Display Data RAM (DD RAM) stores the character code to be displayed on the screen. Most applications interact primarily with DD RAM. The character code stored in a DD RAM location references a specific character bitmap stored either in the predefined [CG ROM](#) character set or in the user-defined [CG RAM](#) character set.

[Figure 5-3](#) shows the default address for the 32 character locations on the display. The upper line of characters is stored between addresses 0x00 and 0x0F. The second line of characters is stored between addresses 0x40 and 0x4F.

	Character Display Addresses																Undisplayed Addresses
1	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10 . . . 27
2	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	50 . . . 67
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17 . . . 40

UG257\_05\_03\_061206

*Figure 5-3: DD RAM Hexadecimal Addresses (No Display Shifting)*

Physically, there are 80 total character locations in DD RAM with 40 characters available per line. Locations 0x10 through 0x27 and 0x50 through 0x67 can be used to store other non-display data. Alternatively, these locations can also store characters that can only be displayed using controller's display shifting functions.

The [Set DD RAM Address](#) command initializes the address counter before reading or writing to DD RAM. Write DD RAM data using the [Write Data to CG RAM or DD RAM](#) command, and read DD RAM using the [Read Data from CG RAM or DD RAM](#) command.

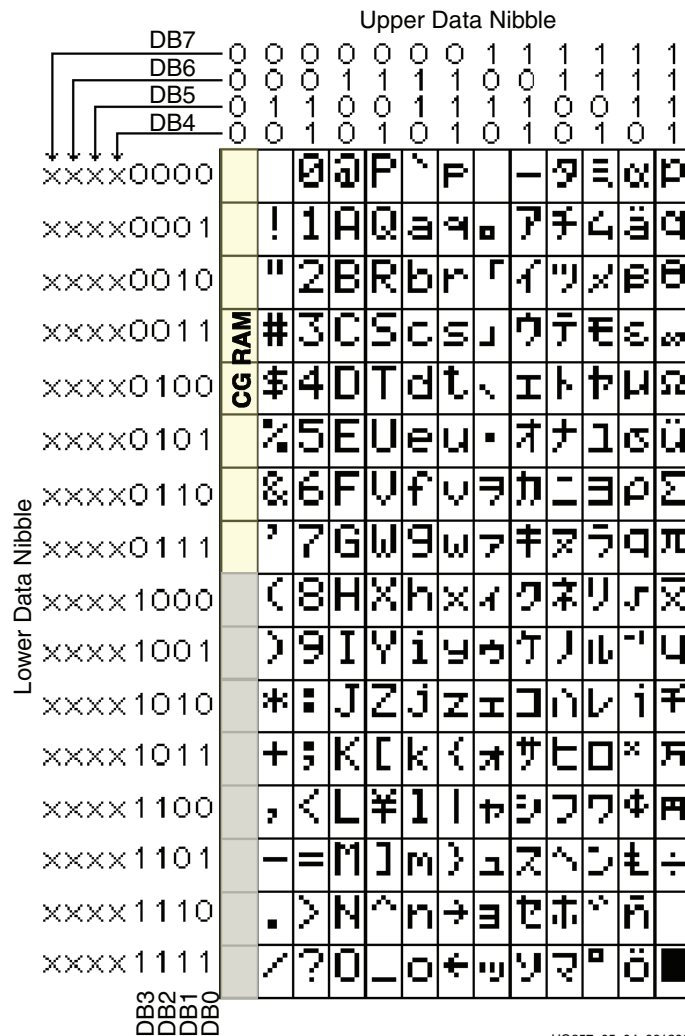
The DD RAM address counter either remains constant after read or write operations, or auto-increments or auto-decrements by one location, as defined by the I/D set by the [Entry Mode Set](#) command.

### CG ROM

The Character Generator ROM (CG ROM) contains the font bitmap for each of the predefined characters that the LCD screen can display, shown in [Figure 5-4](#). The character code stored in [DD RAM](#) for each character location subsequently references a position with the CG ROM. For example, a hexadecimal character code of 0x53 stored in a [DD RAM](#) location displays the character 'S'. The upper nibble of 0x53 equates to  $DB[7:4] = "0101"$

binary and the lower nibble equates to DB[3:0] = "0011" binary. As shown in Figure 5-4, the character 'S' appears on the screen.

English/Roman characters are stored in CG ROM at their equivalent ASCII code address.



UG257\_05\_04\_061206

Figure 5-4: LCD Character Set

The character ROM contains the ASCII English character set and Japanese kana characters.

The controller also provides for eight custom character bitmaps, stored in CG RAM. These eight custom characters are displayed by storing character codes 0x00 through 0x07 in a DD RAM location.

### CG RAM

The Character Generator RAM (CG RAM) provides space to create eight custom character bitmaps. Each custom character location consists of a 5-dot by 8-line bitmap, as shown in Figure 5-5.

The Set CG RAM Address command initializes the address counter before reading or writing to CG RAM. Write CG RAM data using the Write Data to CG RAM or DD RAM command, and read CG RAM using the Read Data from CG RAM or DD RAM command.

The CG RAM address counter can either remain constant after read or write operations, or auto-increments or auto-decrements by one location, as defined by the I/D set by the [Entry Mode Set](#) command.

[Figure 5-5](#) provides an example, creating a special *checkerboard* character. The custom character is stored in the fourth CG RAM character location, which is displayed when a DD RAM location is 0x03. To write the custom character, the CG RAM address is first initialized using the [Set CG RAM Address](#) command. The upper three address bits point to the custom character location. The lower three address bits point to the row address for the character bitmap. The [Write Data to CG RAM or DD RAM](#) command is used to write each character bitmap row. A '1' lights a bit on the display. A '0' leaves the bit unlit. Only the lower five data bits are used; the upper three data bits are *don't care* positions. The eighth row of bitmap data is usually left as all zeros to accommodate the cursor.

						Upper Nibble				Lower Nibble			
						Write Data to CG RAM or DD RAM							
A5	A4	A3	A2	A1	A0	D7	D6	D5	D4	D3	D2	D1	D0
Character Addresses			Row Addresses			Don't Care			Character Bitmap				
0	1	1	0	0	0	—	—	—	0	0	0	0	0
0	1	1	0	0	1	—	—	—	0	0	0	0	0
0	1	1	0	1	0	—	—	—	0	0	0	0	0
0	1	1	0	1	1	—	—	—	0	0	0	0	0
0	1	1	1	0	0	—	—	—	0	0	0	0	0
0	1	1	1	0	1	—	—	—	0	0	0	0	0
0	1	1	1	1	0	—	—	—	0	0	0	0	0
0	1	1	1	1	1	—	—	—	0	0	0	0	0

UG257\_05\_05\_061406

**Figure 5-5: Example Custom Checkerboard Character with Character Code 0x03**

## Command Set

[Table 5-3](#) summarizes the available LCD controller commands and bit definitions. Because the display is set up for 4-bit operation, each 8-bit command is sent as two 4-bit nibbles. The upper nibble is transferred first, followed by the lower nibble.

**Table 5-3: LCD Character Display Command Set**

Function	LCD_RS	LCD_RW	Upper Nibble				Lower Nibble				
			DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
<a href="#">Clear Display</a>	0	0	0	0	0	0	0	0	0	0	1
<a href="#">Return Cursor Home</a>	0	0	0	0	0	0	0	0	0	1	-
<a href="#">Entry Mode Set</a>	0	0	0	0	0	0	0	0	1	I/D	S
<a href="#">Display On/Off</a>	0	0	0	0	0	0	0	1	D	C	B
<a href="#">Cursor and Display Shift</a>	0	0	0	0	0	1	S/C	R/L	-	-	-
<a href="#">Function Set</a>	0	0	0	0	1	0	1	0	-	-	-

Table 5-3: LCD Character Display Command Set (Continued)

Function	LCD_RS	LCD_RW	Upper Nibble				Lower Nibble			
			DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Set CG RAM Address	0	0	0	1	A5	A4	A3	A2	A1	A0
Set DD RAM Address	0	0	1	A6	A5	A4	A3	A2	A1	A0
Read Busy Flag and Address	0	1	BF	A6	A5	A4	A3	A2	A1	A0
Write Data to CG RAM or DD RAM	1	0	D7	D6	D5	D4	D3	D2	D1	D0
Read Data from CG RAM or DD RAM	1	1	D7	D6	D5	D4	D3	D2	D1	D0

### Disabled

If the LCD\_E enable signal is Low, all other inputs to the LCD are ignored.

### Clear Display

Clear the display and return the cursor to the home position, the top-left corner.

This command writes a blank space (ASCII/ANSI character code 0x20) into all DD RAM addresses. The address counter is reset to 0, location 0x00 in DD RAM. Clears all option settings. The I/D control bit is set to 1 (increment address counter mode) in the [Entry Mode Set](#) command.

Execution Time: 82 μs – 1.64 ms

### Return Cursor Home

Return the cursor to the home position, the top-left corner. DD RAM contents are unaffected. Also returns the display being shifted to the original position, shown in [Figure 5-3](#).

The address counter is reset to 0, location 0x00 in DD RAM. The display is returned to its original status if it was shifted. The cursor or blink move to the top-left character location.

Execution Time: 40 μs – 1.6 ms

### Entry Mode Set

Sets the cursor move direction and specifies whether or not to shift the display.

These operations are performed during data reads and writes.

Execution Time: 40 μs

#### Bit DB1: (I/D) Increment/Decrement

0	Auto-decrement address counter. Cursor/blink moves to left.
1	Auto-increment address counter. Cursor/blink moves to right.



This bit either auto-increments or auto-decrements the DD RAM and CG RAM address counter by one location after each [Write Data to CG RAM or DD RAM](#) or [Read Data from CG RAM or DD RAM](#) command. The cursor or blink position moves accordingly.

#### Bit DB0: (S) Shift

0	Shifting disabled
1	During a DD RAM write operation, shift the entire display value in the direction controlled by Bit DB1 (I/D). Appears as though the cursor position remains constant and the display moves.

### Display On/Off

Display is turned on or off, controlling all characters, cursor and cursor position character (underscore) blink.

Execution Time: 40  $\mu$ s

#### Bit DB2: (D) Display On/Off

0	No characters displayed. However, data stored in DD RAM is retained
1	Display characters stored in DD RAM

#### Bit DB1: (C) Cursor On/Off

The cursor uses the five dots on the bottom line of the character. The cursor appears as a line under the displayed character.

0	No cursor
1	Display cursor

#### Bit DB0: (B) Cursor Blink On/Off

0	No cursor blinking
1	Cursor blinks on and off approximately every half second

### Cursor and Display Shift

Moves the cursor and shifts the display without changing DD RAM contents. Shift cursor position or display to the right or left without writing or reading display data.

This function positions the cursor in order to modify an individual character, or to scroll the display window left or right to reveal additional data stored in the DD RAM, beyond the 16th character on a line. The cursor automatically moves to the second line when it shifts beyond the 40th character location of the first line. The first and second line displays shift at the same time.

When the displayed data is shifted repeatedly, both lines move horizontally. The second display line does not shift into the first display line.

Execution Time: 40  $\mu$ s

Table 5-4: Shift Patterns According to S/C and R/L Bits

DB3 (S/C)	DB2 (R/L)	Operation
0	0	Shift the cursor position to the left. The address counter is decremented by one.
0	1	Shift the cursor position to the right. The address counter is incremented by one.
1	0	Shift the entire display to the left. The cursor follows the display shift. The address counter is unchanged.
1	1	Shift the entire display to the right. The cursor follows the display shift. The address counter is unchanged.

## Function Set

Sets interface data length, number of display lines, and character font.

The Starter Kit board supports a single function set with value 0x28.

Execution Time: 40  $\mu$ s

## Set CG RAM Address

Set the initial CG RAM address.

After this command, all subsequent read or write operations to the display are to or from CG RAM.

Execution Time: 40  $\mu$ s

## Set DD RAM Address

Set the initial DD RAM address.

After this command, all subsequent read or write operations to the display are to or from DD RAM. The addresses for displayed characters appear in [Figure 5-3](#).

Execution Time: 40  $\mu$ s

## Read Busy Flag and Address

Read the Busy flag (BF) to determine if an internal operation is in progress, and read the current address counter contents.

BF = 1 indicates that an internal operation is in progress. The next instruction is not accepted until BF is cleared or until the current instruction is allowed the maximum time to execute.

This command also returns the present value of address counter. The address counter is used for both CG RAM and DD RAM addresses. The specific context depends on the most recent [Set CG RAM Address](#) or [Set DD RAM Address](#) command issued.

Execution Time: 1  $\mu$ s

## Write Data to CG RAM or DD RAM

Write data into DD RAM if the command follows a previous [Set DD RAM Address](#) command, or write data into CG RAM if the command follows a previous [Set CG RAM Address](#) command.

After the write operation, the address is automatically incremented or decremented by 1 according to the [Entry Mode Set](#) command. The entry mode also determines display shift.

Execution Time: 40  $\mu$ s

## Read Data from CG RAM or DD RAM

Read data from DD RAM if the command follows a previous [Set DD RAM Address](#) command, or read data from CG RAM if the command follows a previous [Set CG RAM Address](#) command.

After the read operation, the address is automatically incremented or decremented by 1 according to the [Entry Mode Set](#) command. However, a display shift is not executed during read operations.

Execution Time: 40  $\mu$ s

## Operation

### Four-Bit Data Interface

The board uses a 4-bit data interface to the character LCD.

[Figure 5-6](#) illustrates a write operation to the LCD, showing the minimum times allowed for setup, hold, and enable pulse length relative to the 50 MHz clock (20 ns period) provided on the board.

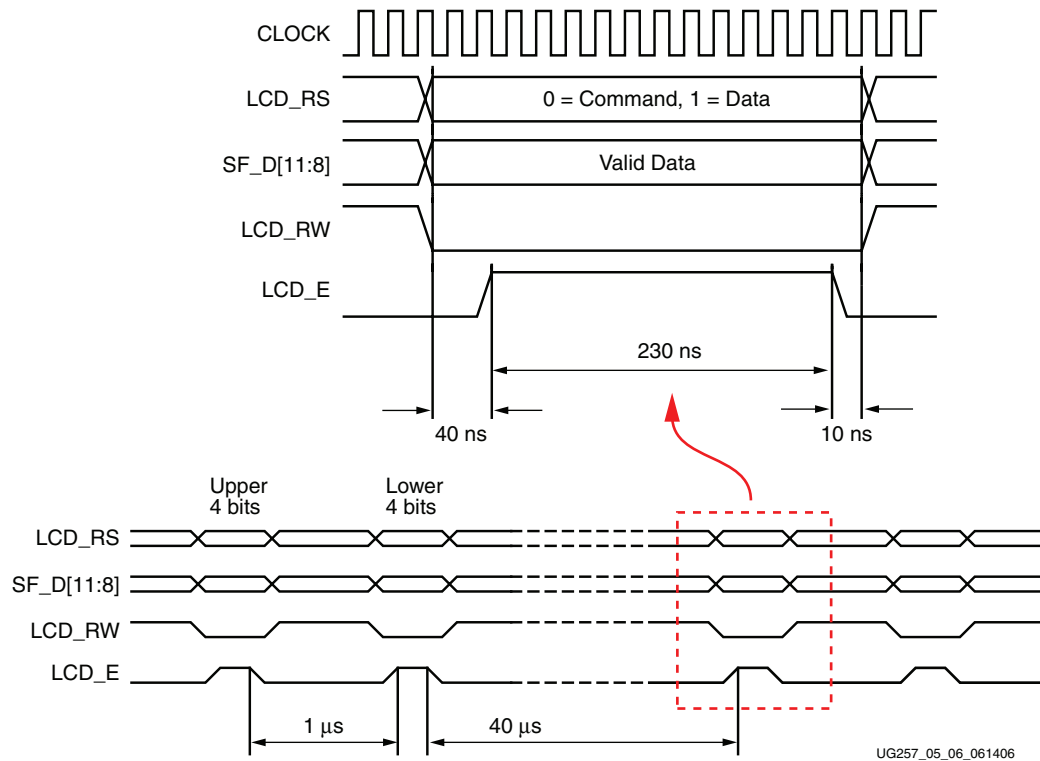


Figure 5-6: Character LCD Interface Timing

The data values on SF\_D<11:8>, and the register select (LCD\_RS) and the read/write (LCD\_RW) control signals must be set up and stable at least 40 ns before the enable LCD\_E goes High. The enable signal must remain High for 230 ns or longer—the equivalent of 12 or more clock cycles at 50 MHz.

In many applications, the LCD\_RW signal can be tied Low permanently because the FPGA generally has no reason to read information from the display.

## Transferring 8-Bit Data over the 4-Bit Interface

After initializing the display and establishing communication, all commands and data transfers to the character display are via 8 bits, transferred using two sequential 4-bit operations. Each 8-bit transfer must be decomposed into two 4-bit transfers, spaced apart by at least 1  $\mu$ s, as shown in [Figure 5-6](#). The upper nibble is transferred first, followed by the lower nibble. An 8-bit write operation must be spaced least 40  $\mu$ s before the next communication. This delay must be increased to 1.64 ms following a [Clear Display](#) command.

## Initializing the Display

After power-on, the display must be initialized to establish the required communication protocol. The initialization sequence is simple and ideally suited to the highly-efficient 8-bit [PicoBlaze](#) embedded controller. After initialization, the PicoBlaze controller is available for more complex control or computation beyond simply driving the display.

### Power-On Initialization

The initialization sequence first establishes that the FPGA application wishes to use the four-bit data interface to the LCD as follows:

- Wait 15 ms or longer, although the display is generally ready when the FPGA finishes configuration. The 15 ms interval is 750,000 clock cycles at 50 MHz.
- Write SF\_D<11:8> = 0x3, pulse LCD\_E High for 12 clock cycles.
- Wait 4.1 ms or longer, which is 205,000 clock cycles at 50 MHz.
- Write SF\_D<11:8> = 0x3, pulse LCD\_E High for 12 clock cycles.
- Wait 100  $\mu$ s or longer, which is 5,000 clock cycles at 50 MHz.
- Write SF\_D<11:8> = 0x3, pulse LCD\_E High for 12 clock cycles.
- Wait 40  $\mu$ s or longer, which is 2,000 clock cycles at 50 MHz.
- Write SF\_D<11:8> = 0x2, pulse LCD\_E High for 12 clock cycles.
- Wait 40  $\mu$ s or longer, which is 2,000 clock cycles at 50 MHz.

### Display Configuration

After the power-on initialization is completed, the four-bit interface is now established. The next part of the sequence configures the display:

- Issue a [Function Set](#) command, 0x28, to configure the display for operation on the Spartan-3E Starter Kit board.
- Issue an [Entry Mode Set](#) command, 0x06, to set the display to automatically increment the address pointer.
- Issue a [Display On/Off](#) command, 0x0C, to turn the display on and disables the cursor and blinking.

- Finally, issue a [Clear Display](#) command. Allow at least 1.64 ms (82,000 clock cycles) after issuing this command.

## Writing Data to the Display

To write data to the display, specify the start address, followed by one or more data values.

Before writing any data, issue a [Set DD RAM Address](#) command to specify the initial 7-bit address in the DD RAM. See [Figure 5-3](#) for DD RAM locations.

Write data to the display using a [Write Data to CG RAM or DD RAM](#) command. The 8-bit data value represents the look-up address into the CG ROM or CG RAM, shown in [Figure 5-4](#). The stored bitmap in the CG ROM or CG RAM drives the 5 x 8 dot matrix to represent the associated character.

If the address counter is configured to auto-increment, as described earlier, the application can sequentially write multiple character codes and each character is automatically stored and displayed in the next available location.

Continuing to write characters, however, eventually falls off the end of the first display line. The additional characters do not automatically appear on the second line because the DD RAM map is not consecutive from the first line to the second.

## Disabling the Unused LCD

If the FPGA application does not use the character LCD screen, drive the LCD\_E pin Low to disable it. Also drive the LCD\_RW pin Low to prevent the LCD screen from presenting data.

## Related Resources

- Initial Design for Spartan-3E MicroBlaze Development Kit (Reference Design)  
<http://www.xilinx.com/s3e1600e>
- PowerTip PC1602-D Character LCD (Basic Electrical and Mechanical Data)  
<http://www.powertipusa.com/pdf/pc1602d.pdf>
- Sitronix ST7066U Character LCD Controller  
<http://www.sitronix.com.tw/sitronix/product.nsf/Doc/ST7066U?OpenDocument>
- Detailed Data Sheet on PowerTip Character LCD  
<http://www.rapidelectronics.co.uk/images/siteimg/57-0910e.PDF>
- Samsung S6A0069X Character LCD Controller  
<http://www.samsung.com/Products/Semiconductor/DisplayDriverIC/MobileDDI/BWSTN/S6A0069X/S6A0069X.htm>

## VGA Display Port

The MicroBlaze Development Kit board includes a VGA display port via a J15 connector. Connect this port directly to most PC monitors or flat-panel LCDs using a standard monitor cable. As shown in Figure 6-1, the VGA connector is the left-most connector along the top of the board.

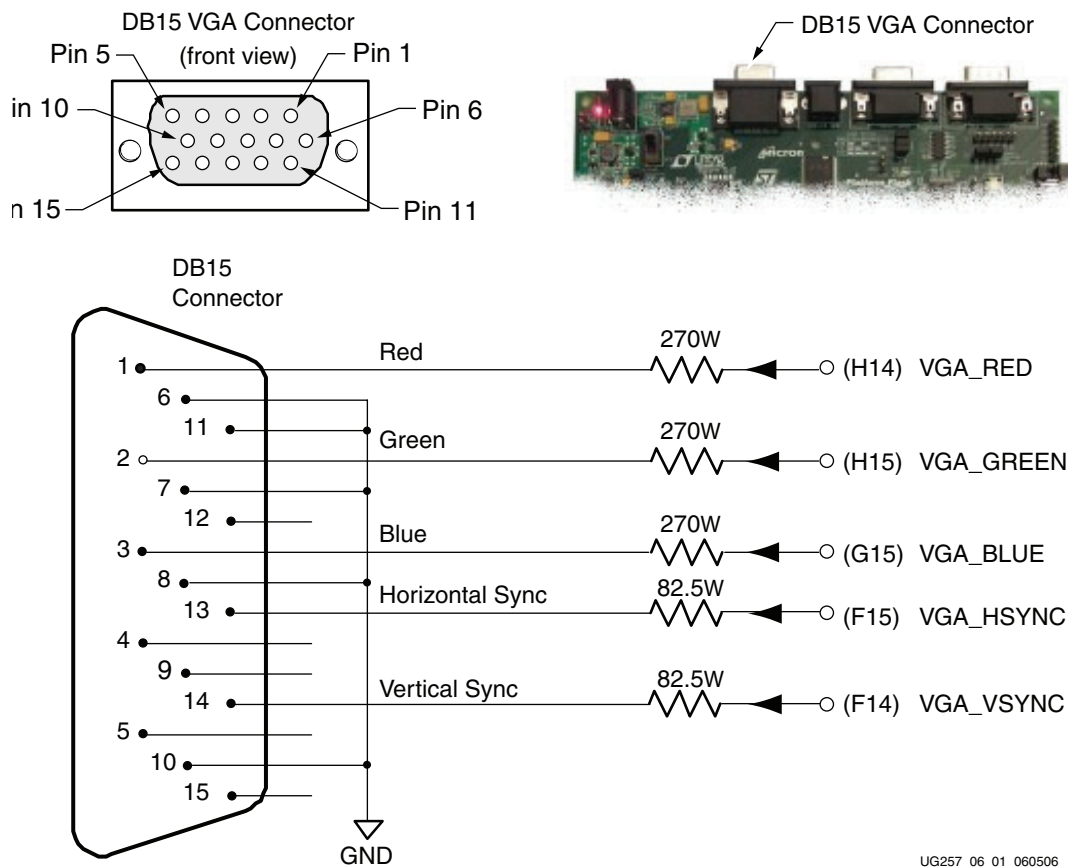


Figure 6-1: VGA Connections from Spartan-3E Starter Kit Board

The Spartan-3E FPGA directly drives the five VGA signals via resistors. Each color line has a series resistor, with one bit each for VGA\_RED, VGA\_GREEN, and VGA\_BLUE. The series resistor, in combination with the 75Ω termination built into the VGA cable, ensures that the color signals remain in the VGA-specified 0V to 0.7V range. The VGA\_HSYNC and VGA\_VSYNC signals use LVTTL or LVCMOS33 I/O standard drive levels. Drive

the VGA\_RED, VGA\_GREEN, and VGA\_BLUE signals High or Low to generate the eight colors shown in [Table 6-1](#).

**Table 6-1: 3-Bit Display Color Codes**

VGA_RED	VGA_GREEN	VGA_BLUE	Resulting Color
0	0	0	Black
0	0	1	Blue
0	1	0	Green
0	1	1	Cyan
1	0	0	Red
1	0	1	Magenta
1	1	0	Yellow
1	1	1	White

VGA signal timing is specified, published, copyrighted, and sold by the Video Electronics Standards Association (VESA). The following VGA system and timing information is provided as an example of how the FPGA might drive VGA monitor in 640 by 480 mode. For more precise information or for information on higher VGA frequencies, refer to documents available on the VESA website or other electronics websites (see [“Related Resources,”](#) page 57).

## Signal Timing for a 60 Hz, 640x480 VGA Display

CRT-based VGA displays use amplitude-modulated, moving electron beams (or cathode rays) to display information on a phosphor-coated screen. LCDs use an array of switches that can impose a voltage across a small amount of liquid crystal, thereby changing light permittivity through the crystal on a pixel-by-pixel basis. Although the following description is limited to CRT displays, LCDs have evolved to use the same signal timings as CRT displays. Consequently, the following discussion pertains to both CRTs and LCDs.

Within a CRT display, current waveforms pass through the coils to produce magnetic fields that deflect electron beams to transverse the display surface in a *raster* pattern, horizontally from left to right and vertically from top to bottom. As shown in [Figure 6-2](#), information is only displayed when the beam is moving in the *forward* direction—left to right and top to bottom—and not during the time the beam returns back to the left or top edge of the display. Much of the potential display time is therefore lost in *blanking* periods when the beam is reset and stabilized to begin a new horizontal or vertical display pass.

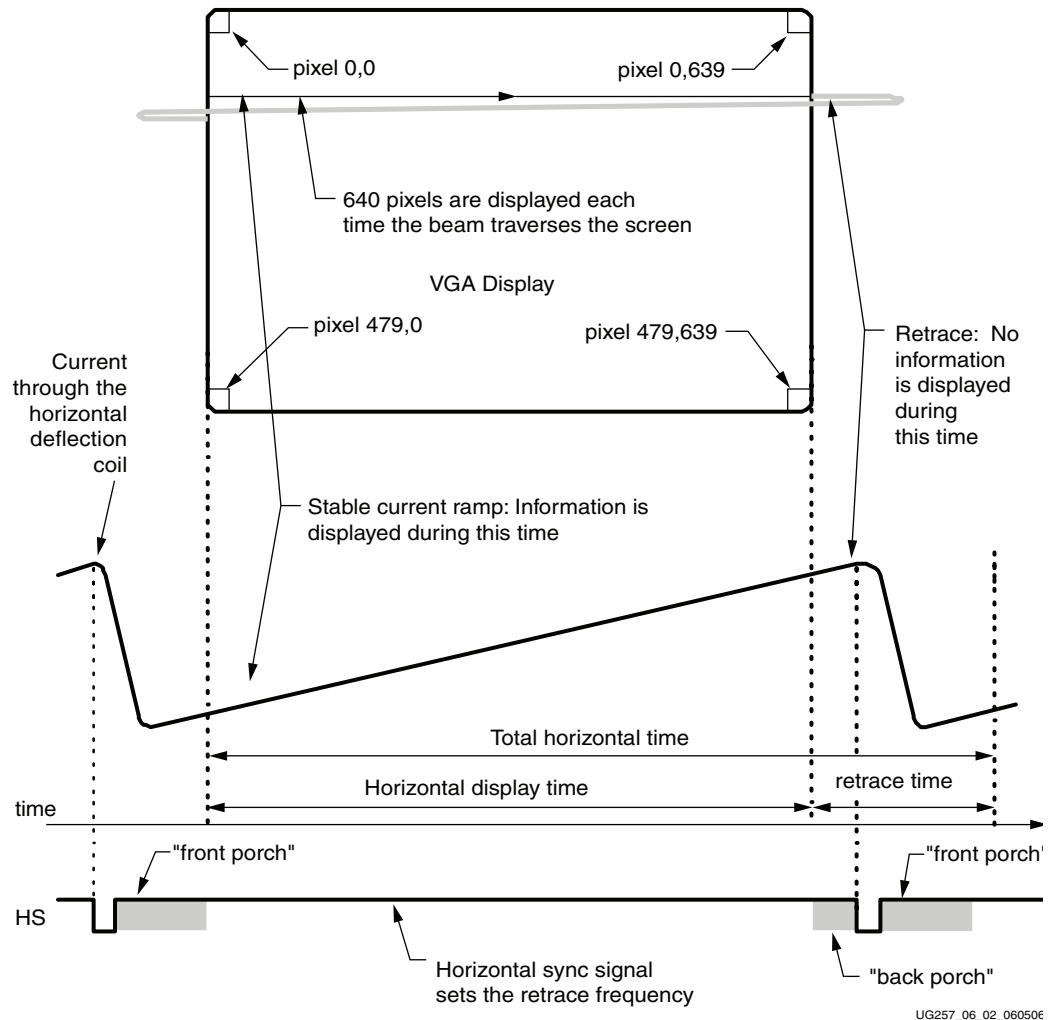


Figure 6-2: CRT Display Timing Example

The display resolution defines the size of the beams, the frequency at which the beam traces across the display, and the frequency at which the electron beam is modulated.

Modern VGA displays support multiple display resolutions, and the VGA controller dictates the resolution by producing timing signals to control the raster patterns. The controller produces TTL-level synchronizing pulses that set the frequency at which current flows through the deflection coils, and it ensures that pixel or video data is applied to the electron guns at the correct time.

Video data typically comes from a video refresh memory with one or more bytes assigned to each pixel location. The MicroBlaze Development Kit board uses three bits per pixel, producing one of the eight possible colors shown in Table 6-1. The controller indexes into the video data buffer as the beams move across the display. The controller then retrieves and applies video data to the display at precisely the time the electron beam is moving across a given pixel.

As shown in Figure 6-2, the VGA controller generates the horizontal sync (HS) and vertical sync (VS) timings signals and coordinates the delivery of video data on each pixel clock. The pixel clock defines the time available to display one pixel of information. The VS signal defines the refresh frequency of the display, or the frequency at which all information on the



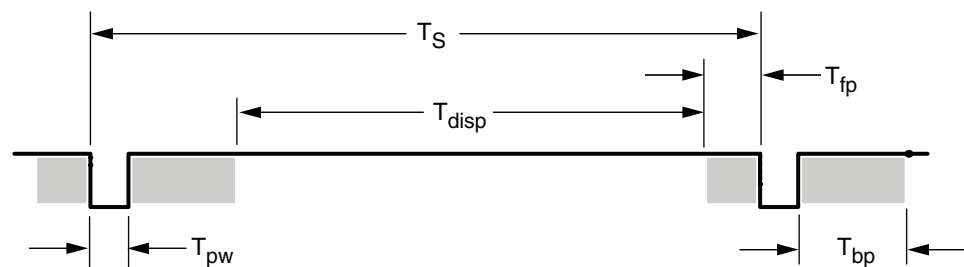
display is redrawn. The minimum refresh frequency is a function of the display's phosphor and electron beam intensity, with practical refresh frequencies in the 60 Hz to 120 Hz range. The number of horizontal lines displayed at a given refresh frequency defines the horizontal *retrace* frequency.

## VGA Signal Timing

The signal timings in Table 6-2 are derived for a 640-pixel by 480-row display using a 25 MHz pixel clock and 60 Hz  $\pm 1$  refresh. Figure 6-3 shows the relation between each of the timing symbols. The timing for the sync pulse width ( $T_{PW}$ ) and front and back porch intervals ( $T_{FP}$  and  $T_{BP}$ ) are based on observations from various VGA displays. The front and back porch intervals are the pre- and post-sync pulse times. Information cannot be displayed during these times.

Table 6-2: 640x480 Mode VGA Timing

Symbol	Parameter	Vertical Sync			Horizontal Sync	
		Time	Clocks	Lines	Time	Clocks
$T_S$	Sync pulse time	16.7 ms	416,800	521	32 $\mu$ s	800
$T_{DISP}$	Display time	15.36 ms	384,000	480	25.6 $\mu$ s	640
$T_{PW}$	Pulse width	64 $\mu$ s	1,600	2	3.84 $\mu$ s	96
$T_{FP}$	Front porch	320 $\mu$ s	8,000	10	640 ns	16
$T_{BP}$	Back porch	928 $\mu$ s	23,200	29	1.92 $\mu$ s	48



UG257\_06\_03\_060506

Figure 6-3: VGA Control Timing

Generally, a counter clocked by the pixel clock controls the horizontal timing. Decoded counter values generate the HS signal. This counter tracks the current pixel display location on a given row.

A separate counter tracks the vertical timing. The vertical-sync counter increments with each HS pulse and decoded values generate the VS signal. This counter tracks the current display row. These two continuously running counters form the address into a video display buffer. For example, the on-board DDR SDRAM provides an ideal display buffer.

No time relationship is specified between the onset of the HS pulse and the onset of the VS pulse. Consequently, the counters can be arranged to easily form video RAM addresses, or to minimize decoding logic for sync pulse generation.

## UCF Location Constraints

Figure 6-4 provides the UCF constraints for the VGA display port, including the I/O pin assignment, the I/O standard used, the output slew rate, and the output drive current.

NET "VGA_RED"	LOC = "H14"	IOSTANDARD = LVTTL	DRIVE = 8	SLEW = FAST ;
NET "VGA_GREEN"	LOC = "H15"	IOSTANDARD = LVTTL	DRIVE = 8	SLEW = FAST ;
NET "VGA_BLUE"	LOC = "G15"	IOSTANDARD = LVTTL	DRIVE = 8	SLEW = FAST ;
NET "VGA_HSYNC"	LOC = "F15"	IOSTANDARD = LVTTL	DRIVE = 8	SLEW = FAST ;
NET "VGA_VSYNC"	LOC = "F14"	IOSTANDARD = LVTTL	DRIVE = 8	SLEW = FAST ;

UG257\_06\_04\_060506

Figure 6-4: UCF Constraints for VGA Display Port

## Related Resources

- VESA  
<http://www.vesa.org>
- VGA timing information  
[http://www.epanorama.net/documents/pc/vga\\_timing.html](http://www.epanorama.net/documents/pc/vga_timing.html)



## RS-232 Serial Ports

---

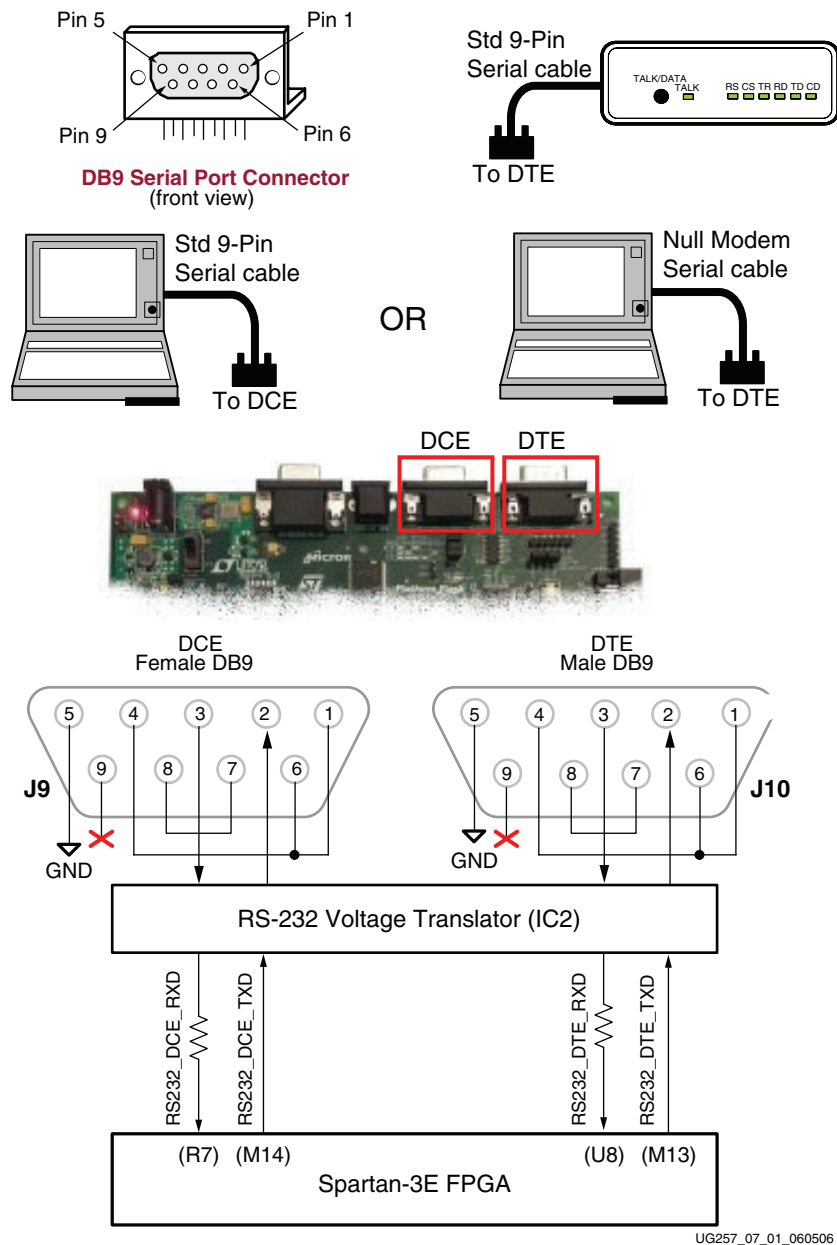
### Overview

As shown in [Figure 7-1](#), the MicroBlaze Development Kit board has two RS-232 serial ports: a female DB9 DCE connector and a male DTE connector. The DCE-style port connects directly to the serial port connector available on most personal computers and workstations via a standard straight-through serial cable. Null modem, gender changers, or crossover cables are not required.

Use the DTE-style connector to control other RS-232 peripherals, such as modems or printers, or perform simple loopback testing with the DCE connector.

[Figure 7-1](#) shows the connection between the FPGA and the two DB9 connectors. The FPGA supplies serial output data using LVTTTL or LVCMOS levels to the Maxim device, which in turn, converts the logic value to the appropriate RS-232 voltage level. Likewise, the Maxim device converts the RS-232 serial input data to LVTTTL levels for the FPGA. A series resistor between the Maxim output pin and the FPGA's RXD pin protects against accidental logic conflicts.

Hardware flow control is not supported on the connector. The port's DCD, DTR, and DSR signals connect together, as shown in [Figure 7-1](#). Similarly, the port's RTS and CTS signals connect together.



UG257\_07\_01\_060506

Figure 7-1: RS-232 Serial Ports

## UCF Location Constraints

Figure 7-2 and Figure 7-3 provide the UCF constraints for the DTE and DCE RS-232 ports, respectively, including the I/O pin assignment and the I/O standard used.

```
NET "RS232_DTE_RXD" LOC = "U8" | IOSTANDARD = LVTTTL ;  
NET "RS232_DTE_TXD" LOC = "M13" | IOSTANDARD = LVTTTL | DRIVE = 8 | SLEW = SLOW ;
```

UG257\_07\_02\_060506

Figure 7-2: UCF Location Constraints for DTE RS-232 Serial Port

```
NET "RS232_DCE_RXD" LOC = "R7" | IOSTANDARD = LVTTTL ;  
NET "RS232_DCE_TXD" LOC = "M14" | IOSTANDARD = LVTTTL | DRIVE = 8 | SLEW = SLOW ;
```

UG257\_07\_03\_060506

Figure 7-3: UCF Location Constraints for DCE RS-232 Serial Port



## PS/2 Mouse/Keyboard Port

The MicroBlaze Development Kit board includes a PS/2 mouse/keyboard port and the standard 6-pin mini-DIN connector, labeled J14 on the board. Figure 8-1 shows the PS/2 connector, and Table 8-1 shows the signals on the connector. Only pins 1 and 5 of the connector attach to the FPGA.

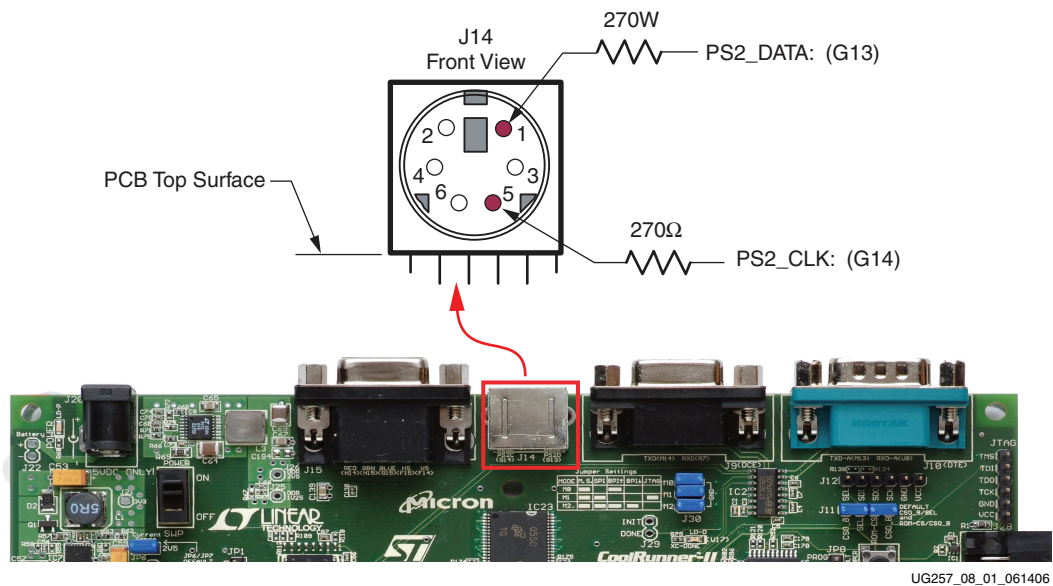


Figure 8-1: PS/2 Connector Location and Signals

Table 8-1: PS/2 Connector Pinout

PS/2 DIN Pin	Signal	FPGA Pin
1	DATA (PS2_DATA)	G13
2	Reserved	G13
3	GND	GND
4	+5V	—
5	CLK (PS2_CLK)	G14
6	Reserved	G14

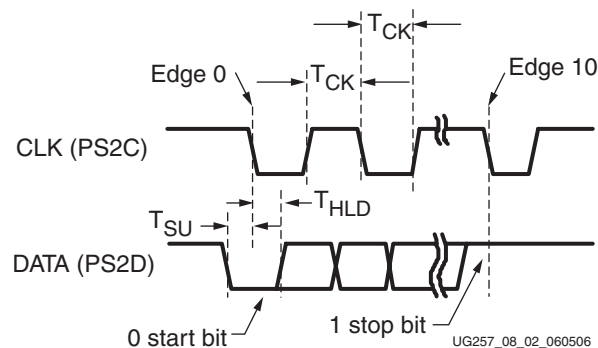


Both a PC mouse and keyboard use the two-wire PS/2 serial bus to communicate with a host device, the Spartan-3E FPGA in this case. The PS/2 bus includes both clock and data. Both a mouse and keyboard drive the bus with identical signal timings and both use 11-bit words that include a start, stop and odd parity bit. However, the data packets are organized differently for a mouse and keyboard. Furthermore, the keyboard interface allows bidirectional data transfers so the host device can illuminate state LEDs on the keyboard.

The PS/2 bus timing appears in [Table 8-2](#) and [Figure 8-2](#). The clock and data signals are only driven when data transfers occur; otherwise they are held in the idle state at logic High. The timing defines signal requirements for mouse-to-host communications and bidirectional keyboard communications. As shown in [Figure 8-2](#), the attached keyboard or mouse writes a bit on the data line when the clock signal is High, and the host reads the data line when the clock signal is Low.

**Table 8-2: PS/2 Bus Timing**

Symbol	Parameter	Min	Max
$T_{CK}$	Clock High or Low Time	30 $\mu$ s	50 $\mu$ s
$T_{SU}$	Data-to-clock Setup Time	5 $\mu$ s	25 $\mu$ s
$T_{HLD}$	Clock-to-data Hold Time	5 $\mu$ s	25 $\mu$ s



**Figure 8-2: PS/2 Bus Timing Waveforms**

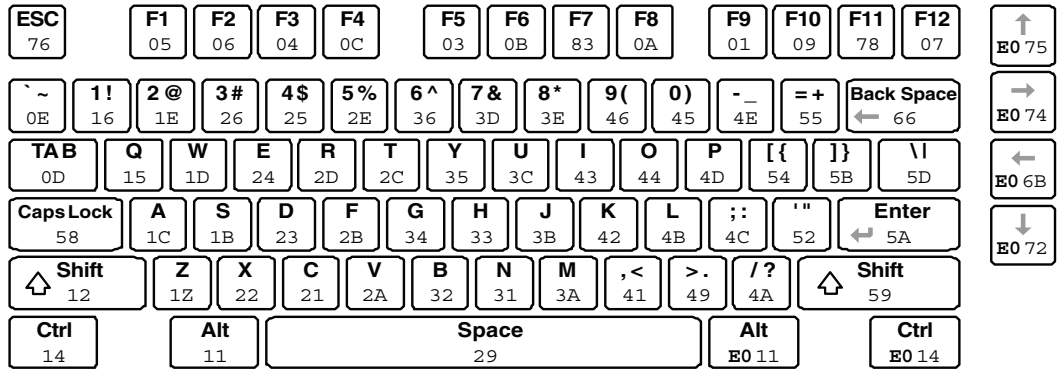
## Keyboard

The keyboard uses open-collector drivers so that either the keyboard or the host can drive the two-wire bus. If the host never sends data to the keyboard, then the host can use simple input pins.

A PS/2-style keyboard uses scan codes to communicate key press data. Nearly all keyboards in use today are PS/2 style. Each key has a single, unique scan code that is sent whenever the corresponding key is pressed. The scan codes for most keys appear in [Figure 8-3](#).

If the key is pressed and held, the keyboard repeatedly sends the scan code every 100 ms or so. When a key is released, the keyboard sends an “F0” key-up code, followed by the scan code of the released key. The keyboard sends the same scan code, regardless if a key has different *shift* and *non-shift* characters and regardless whether the Shift key is pressed or not. The host determines which character is intended.

Some keys, called extended keys, send an “E0” ahead of the scan code and furthermore, they might send more than one scan code. When an extended key is released, an “E0 F0” key-up code is sent, followed by the scan code.



UG257\_08\_03\_060506

Figure 8-3: PS/2 Keyboard Scan Codes

The host can also send commands and data to the keyboard. Table 8-3 provides a short list of some often-used commands.

Table 8-3: Common PS/2 Keyboard Commands

Command	Description																
ED	<p><b>Turn on/off Num Lock, Caps Lock, and Scroll Lock LEDs.</b> The keyboard acknowledges receipt of an “ED” command by replying with an “FA”, after which the host sends another byte to set LED status. The bit positions for the keyboard LEDs are shown below. Write a ‘1’ to the specific bit to illuminate the associated keyboard LED.</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>7</th> <th>6</th> <th>5</th> <th>4</th> <th>3</th> <th>2</th> <th>1</th> <th>0</th> </tr> </thead> <tbody> <tr> <td colspan="5" style="text-align: center;">Ignored</td> <td style="text-align: center;">Caps Lock</td> <td style="text-align: center;">Num Lock</td> <td style="text-align: center;">Scroll Lock</td> </tr> </tbody> </table>	7	6	5	4	3	2	1	0	Ignored					Caps Lock	Num Lock	Scroll Lock
7	6	5	4	3	2	1	0										
Ignored					Caps Lock	Num Lock	Scroll Lock										
EE	<b>Echo.</b> Upon receiving an echo command, the keyboard replies with the same scan code “EE”.																
F3	<b>Set scan code repeat rate.</b> The keyboard acknowledges receipt of an “F3” by returning an “FA”, after which the host sends a second byte to set the repeat rate.																
FE	<b>Resend.</b> Upon receiving a resend command, the keyboard resends the last scan code sent.																
FF	<b>Reset.</b> Resets the keyboard.																

The keyboard sends commands or data to the host only when both the data and clock lines are High, the Idle state.

Because the host is the *bus master*, the keyboard checks whether the host is sending data before driving the bus. The clock line can be used as a *clear to send* signal. If the host pulls the clock line Low, the keyboard must not send any data until the clock is released.

The keyboard sends data to the host in 11-bit words that contain a '0' start bit, followed by eight bits of scan code (LSB first), followed by an odd parity bit and terminated with a '1' stop bit. When the keyboard sends data, it generates 11 clock transitions at around 20 to 30 kHz, and data is valid on the falling edge of the clock as shown in Figure 8-2.

## Mouse

A mouse generates a clock and data signal when moved; otherwise, these signals remain High, indicating the Idle state. Each time the mouse is moved, the mouse sends three 11-bit words to the host. Each of the 11-bit words contains a '0' start bit, followed by 8 data bits (LSB first), followed by an odd parity bit, and terminated with a '1' stop bit. Each data transmission contains 33 total bits, where bits 0, 11, and 22 are '0' start bits, and bits 10, 21, and 32 are '1' stop bits. The three 8-bit data fields contain movement data as shown in Figure 8-4. Data is valid at the falling edge of the clock, and the clock period is 20 to 30 kHz.

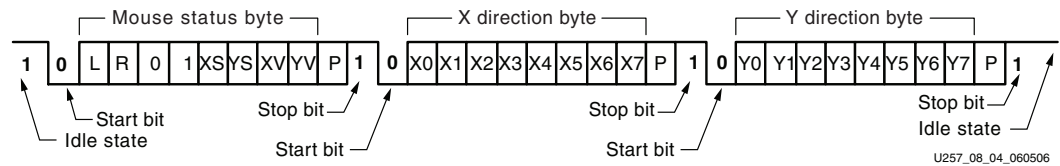


Figure 8-4: PS/2 Mouse Transaction

A PS/2-style mouse employs a relative coordinate system (see Figure 8-5), wherein moving the mouse to the right generates a positive value in the X field, and moving to the left generates a negative value. Likewise, moving the mouse up generates a positive value in the Y field, and moving it down represents a negative value. The XS and YS bits in the status byte define the sign of each value, where a '1' indicates a negative value.

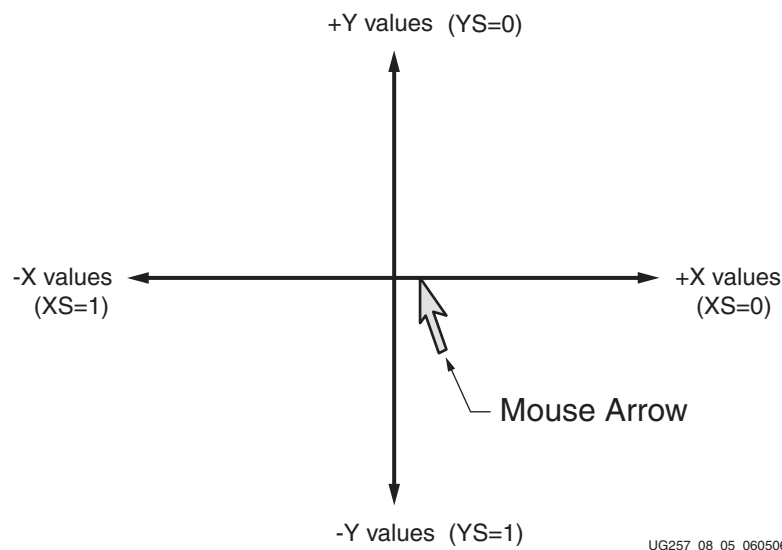


Figure 8-5: The Mouse Uses a Relative Coordinate System to Track Movement

The magnitude of the X and Y values represent the rate of mouse movement. The larger the value, the faster the mouse is moving. The XV and YV bits in the status byte indicate when

the X or Y values exceed their maximum value, an overflow condition. A '1' indicates when an overflow occurs. If the mouse moves continuously, the 33-bit transmissions repeat every 50 ms or so.

The L and R fields in the status byte indicate Left and Right button presses. A '1' indicates that the associated mouse button is being pressed.

## Voltage Supply

The PS/2 port on the MicroBlaze Development Kit board is powered by 5V. Although the Spartan-3E FPGA is not a 5V-tolerant device, it can communicate with a 5V device using series current-limiting resistors, as shown in [Figure 8-1](#).

## UCF Location Constraints

[Figure 8-6](#) provides the UCF constraints for the PS/2 port connecting, including the I/O pin assignment and the I/O standard used.

```
NET "PS2_CLK" LOC = "G14" | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = SLOW ;  
NET "PS2_DATA" LOC = "G13" | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = SLOW ;
```

U257\_08\_06\_060506

*Figure 8-6: UCF Location Constraints for PS/2 Port*

## Related Resources

- PS/2 Mouse/Keyboard Protocol  
<http://www.computer-engineering.org/ps2protocol/>
- PS/2 Keyboard Interface  
<http://www.computer-engineering.org/ps2keyboard/>
- PS/2 Mouse Interface  
<http://www.computer-engineering.org/ps2mouse/>



## Digital to Analog Converter (DAC)

The MicroBlaze Development Kit board includes an SPI-compatible, four-channel, serial Digital-to-Analog Converter (DAC). The DAC device is a Linear Technology LTC2624 quad DAC with 12-bit unsigned resolution. The four outputs from the DAC appear on the J5 header, which uses the Digilent 6-pin [Peripheral Module](#) format. The DAC and the header are located immediately above the Ethernet RJ-45 connector, as shown in [Figure 9-1](#).

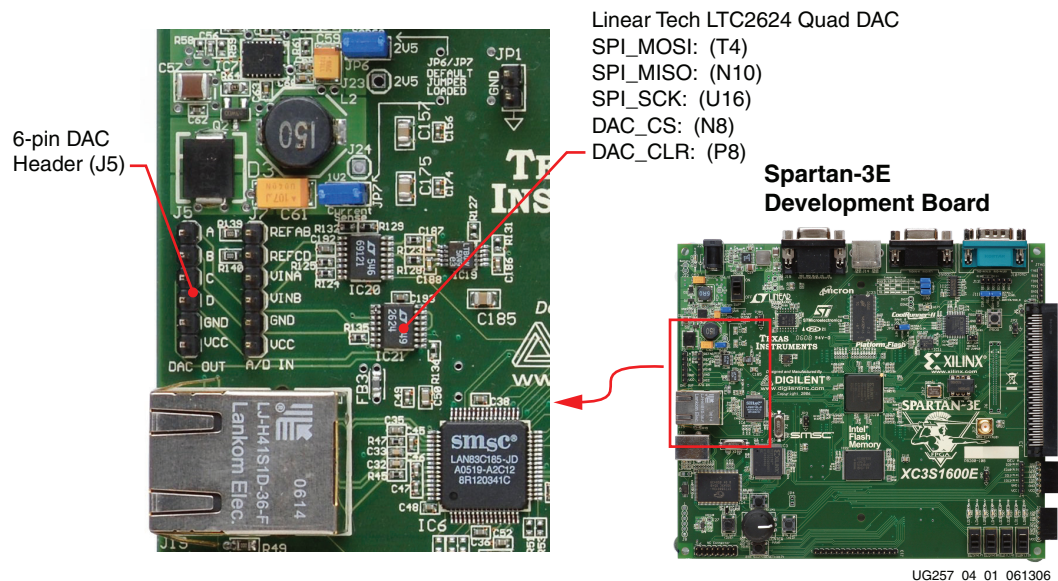


Figure 9-1: Digital-to-Analog Converter and Associated Header

### SPI Communication

As shown in [Figure 9-2](#), the FPGA uses a Serial Peripheral Interface (SPI) to communicate digital values to each of the four DAC channels. The SPI bus is a full-duplex, synchronous, character-oriented channel employing a simple four-wire interface. A bus master—the FPGA in this example—drives the bus clock signal (SPI\_SCK) and transmits serial data (SPI\_MOSI) to the selected bus slave—the DAC in this example. At the same time, the bus slave provides serial data (SPI\_MISO) back to the bus master.

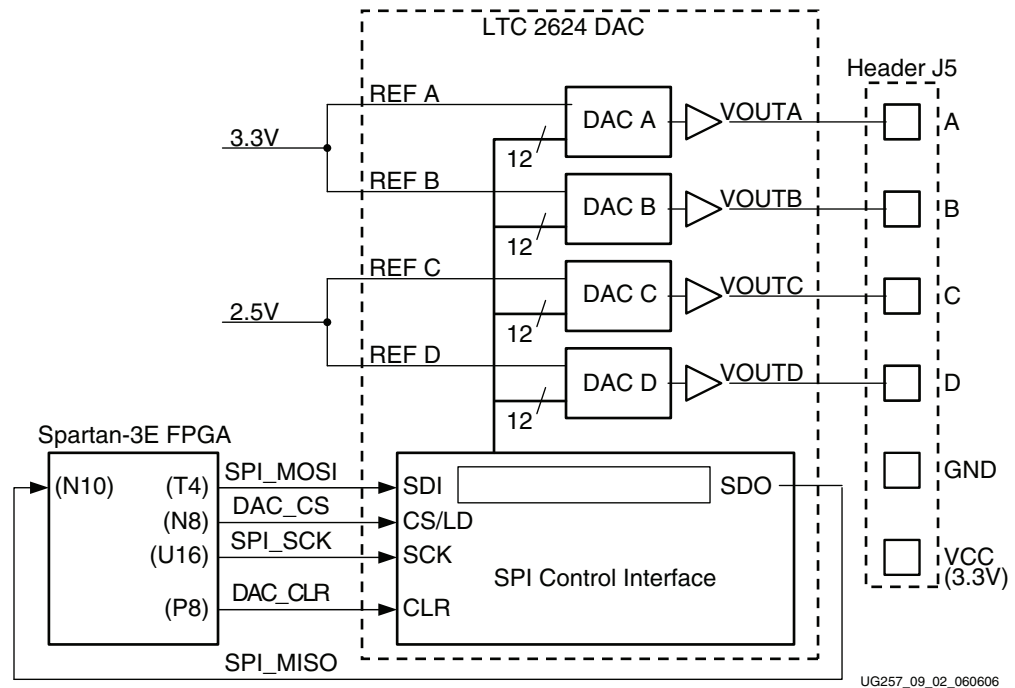


Figure 9-2: Digital-to-Analog Connection Schematics

## Interface Signals

Table 9-1 lists the interface signals between the FPGA and the DAC. The SPI\_MOSI, SPI\_MISO, and SPI\_SCK signals are shared with other devices on the SPI bus. The DAC\_CS signal is the active-Low slave select input to the DAC. The DAC\_CLR signal is the active-Low, asynchronous reset input to the DAC.

Table 9-1: DAC Interface Signals

Signal	FPGA Pin	Direction	Description
SPI_MOSI	T4	FPGA→DAC	Serial data: Master Output, Slave Input
DAC_CS	N8	FPGA→DAC	Active-Low chip-select. Digital-to-analog conversion starts when signal returns High.
SPI_SCK	U16	FPGA→DAC	Clock
DAC_CLR	P8	FPGA→DAC	Asynchronous, active-Low reset input
SPI_MISO	N10	FPGA←DAC	Serial data: Master Input, Slave Output

The serial data output from the DAC is primarily used to cascade multiple DACs. This signal can be ignored in most applications although it does demonstrate full-duplex communication over the SPI bus.

## Disable Other Devices on the SPI Bus to Avoid Contention

The SPI bus signals are shared by other devices on the board. It is vital that other devices are disabled when the FPGA communicates with the DAC to avoid bus contention.

Table 9-2 provides the signals and logic values required to disable the other devices.

Although the StrataFlash PROM is a parallel device, its least-significant data bit is shared with the SPI\_MISO signal.

Table 9-2: Disabled Devices on the SPI Bus

Signal	Disabled Device	Disable Value
SPI_SS_B	SPI serial Flash	1
AMP_CS	Programmable pre-amplifier	1
AD_CONV	Analog-to-Digital Converter (ADC)	0
SF_CE0	StrataFlash Parallel Flash PROM	1
FPGA_INIT_B	Platform Flash PROM	1

## SPI Communication Details

Figure 9-3 shows a detailed example of the SPI bus timing. Each bit is transmitted or received relative to the SPI\_SCK clock signal. The bus is fully static and supports clocks rate up to the maximum of 50 MHz. However, check all timing parameters using the LTC2624 data sheet if operating at or close to the maximum speed.

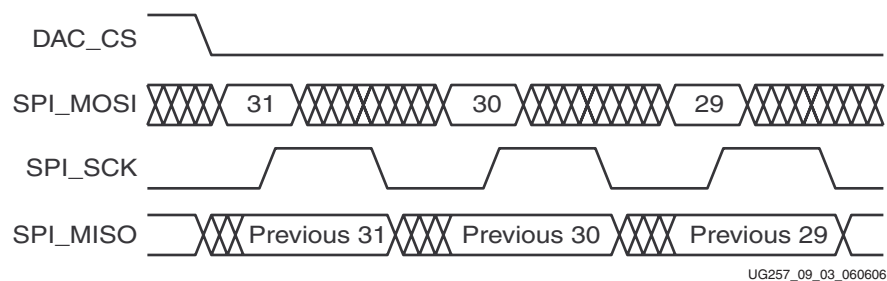


Figure 9-3: SPI Communication Waveforms

After driving the DAC\_CS slave select signal Low, the FPGA transmits data on the SPI\_MOSI signal, MSB first. The LTC2624 captures input data (SPI\_MOSI) on the rising edge of SPI\_SCK; the data must be valid for at least 4 ns relative to the rising clock edge.

The LTC2624 DAC transmits its data on the SPI\_MISO signal on the falling edge of SPI\_SCK. The FPGA captures this data on the next rising SPI\_SCK edge. The FPGA must read the first SPI\_MISO value on the first rising SPI\_SCK edge after DAC\_CS goes Low. Otherwise, bit 31 is missed.

After transmitting all 32 data bits, the FPGA completes the SPI bus transaction by returning the DAC\_CS slave select signal High. The High-going edge starts the actual digital-to-analog conversion process within the DAC.

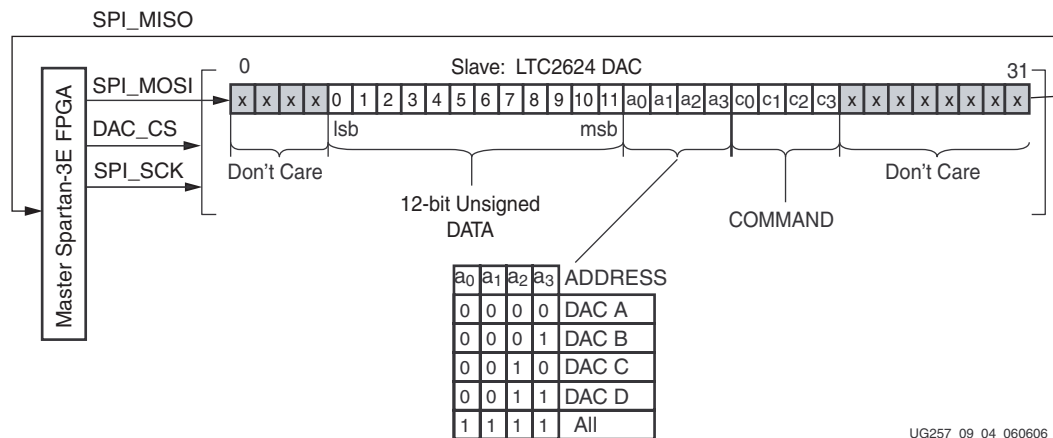
## Communication Protocol

Figure 9-4 shows the communications protocol required to interface with the LTC2624 DAC. The DAC supports both a 24-bit and 32-bit protocol. The 32-bit protocol is shown.

Inside the D/A converter, the SPI interface is formed by a 32-bit shift register. Each 32-bit command word consists of a command, an address, followed by data value. As a new command enters the DAC, the previous 32-bit command word is echoed back to the



master. The response from the DAC can be ignored although it is a useful to confirm correct communication.



UG257\_09\_04\_060606

Figure 9-4: SPI Communications Protocol to LTC2624 DAC

The FPGA first sends eight dummy or “don’t care” bits, followed by a 4-bit command. The most commonly used command with the board is COMMAND[3:0] = “0011”, which immediately updates the selected DAC output with the specified data value. Following the command, the FPGA selects one or all the DAC output channels via a 4-bit address field. Following the address field, the FPGA sends a 12-bit unsigned data value that the DAC converts to an analog value on the selected output(s). Finally, four additional dummy or *don’t care* bits pad the 32-bit command word.

## Specifying the DAC Output Voltage

As shown in Figure 9-2, each DAC output level is the analog equivalent of a 12-bit unsigned digital value,  $D[11:0]$ , written by the FPGA to the DAC via the SPI interface.

The voltage on a specific output is generally described in Equation 9-1. The reference voltage,  $V_{\text{REFERENCE}}$ , is different between the four DAC outputs. Channels A and B use a 3.3V reference voltage and Channels C and D use a 2.5V reference. The reference voltages themselves have a  $\pm 5\%$  tolerance, so there will be slight corresponding variances in the output voltage.

$$V_{\text{OUT}} = \frac{D[11:0]}{4096} \times V_{\text{REFERENCE}} \quad \text{Equation 9-1}$$

### DAC Outputs A and B

Equation 9-2 provides the output voltage equation for DAC outputs A and B. The reference voltage associated with DAC outputs A and B is  $3.3V \pm 5\%$ .

$$V_{\text{OUTA}} = \frac{D[11:0]}{4096} \times (3.3V \pm 5\%) \quad \text{Equation 9-2}$$

## DAC Outputs C and D

Equation 9-3 provides the output voltage equation for DAC outputs A and B. The reference voltage associated with DAC outputs A and B is  $2.5V \pm 5\%$ .

$$V_{OUTC} = \frac{D[11:0]}{4096} \times (2.5V \pm 5\%) \quad \text{Equation 9-3}$$

## UCF Location Constraints

Figure 9-5 provides the UCF constraints for the DAC interface, including the I/O pin assignment and the I/O standard used.

```
NET "SPI_MISO" LOC = "N10" | IOSTANDARD = LVCMOS33 ;
NET "SPI_MOSI" LOC = "T4" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;
NET "SPI_SCK" LOC = "U16" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;
NET "DAC_CS" LOC = "N8" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;
NET "DAC_CLR" LOC = "P8" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;
```

UG257\_09\_05\_060606

Figure 9-5: UCF Location Constraints for the DAC Interface

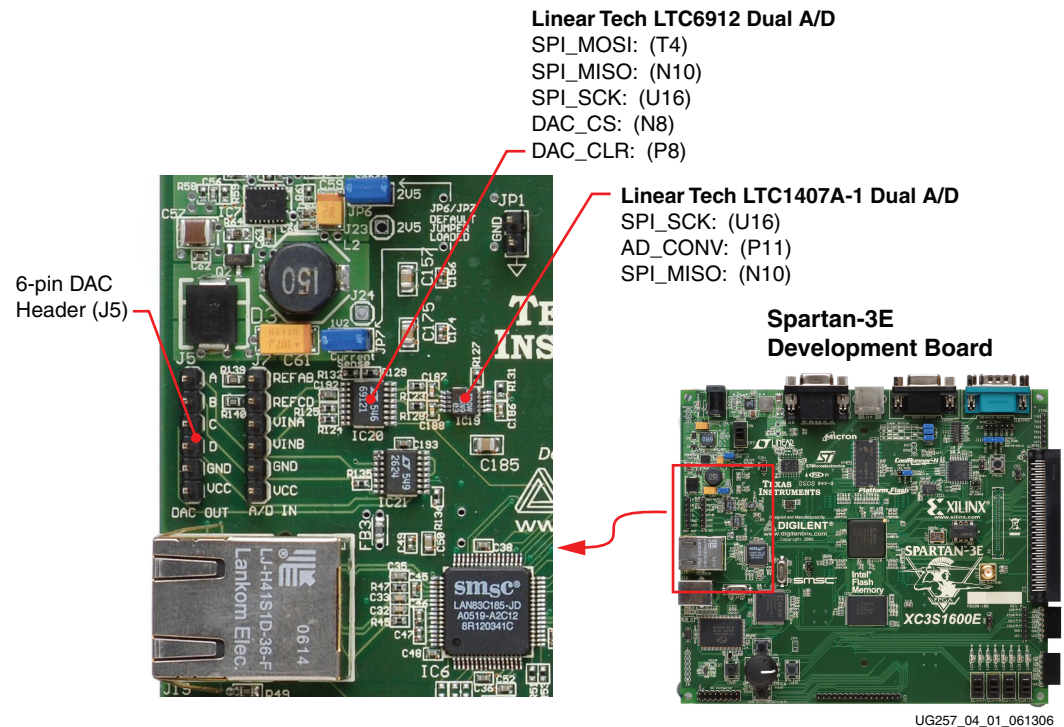
## Related Resources

- LTC2624 Quad DAC Data Sheet
- <http://www.linear.com/pc/downloadDocument.do?navId=H0,C1,C1155,C1005,C1156,P2048,D2170>
- PicoBlaze Based D/A Converter Control for the Spartan-3E Starter Kit (Reference Design)
- <http://www.xilinx.com/sp3e1600e>
- Xilinx PicoBlaze Soft Processor
- <http://www.xilinx.com/picoblaze>
- Digilent, Inc. Peripheral Modules  
<http://www.digilentinc.com/Products/Catalog.cfm?Nav1=Products&Nav2=Peripheral&Cat=Peripheral>



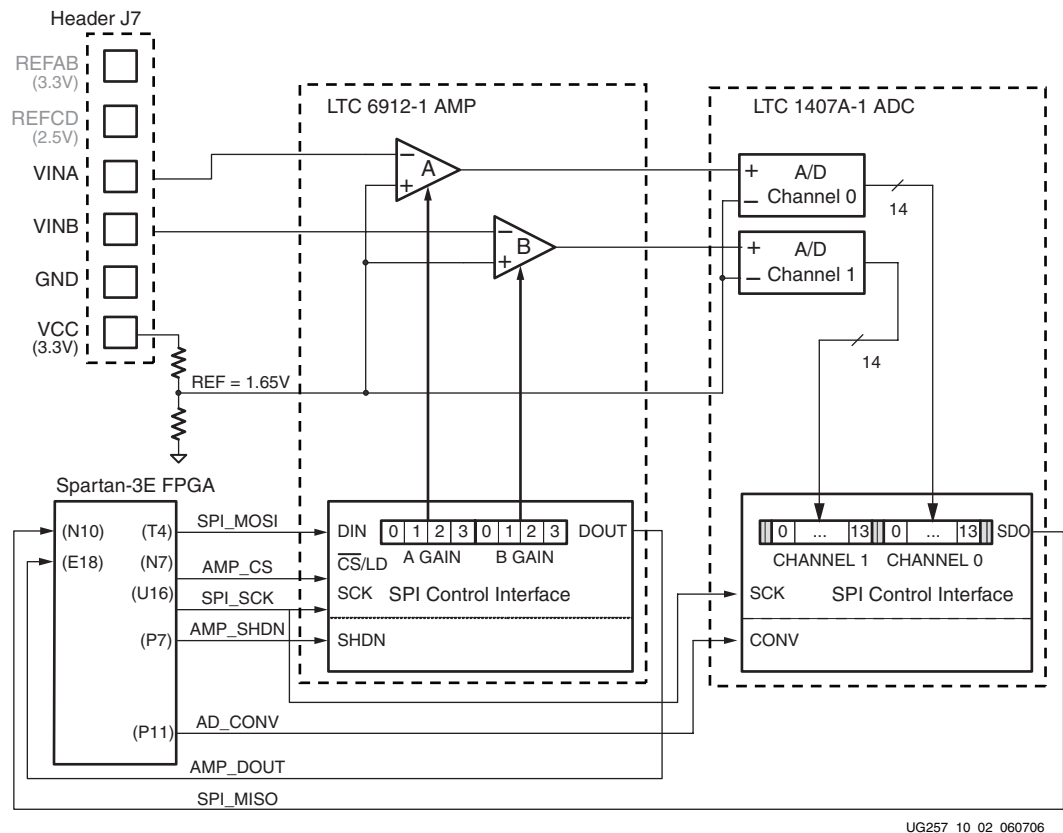
## Analog Capture Circuit

The MicroBlaze Development Kit board includes a two-channel analog capture circuit, consisting of a programmable scaling pre-amplifier and an analog-to-digital converter (ADC), as shown in [Figure 10-1](#). Analog inputs are supplied on the J7 header.



**Figure 10-1: Two-Channel Analog Capture Circuit**

The analog capture circuit consists of a Linear Technology LTC6912-1 programmable pre-amplifier that scales the incoming analog signal on header J7 (see [Figure 10-2](#)). The output of pre-amplifier connects to a Linear Technology LTC1407A-1 ADC. Both the pre-amplifier and the ADC are serially programmed or controlled by the FPGA.



UG257\_10\_02\_060706

Figure 10-2: Detailed View of Analog Capture Circuit

## Digital Outputs from Analog Inputs

The analog capture circuit converts the analog voltage on VINA or VINB and converts it to a 14-bit digital representation, D[13:0], as expressed by Equation 10-1.

$$D[13:0] = GAIN \times \frac{(V_{IN} - 1.65V)}{1.25V} \times 8192 \quad \text{Equation 10-1}$$

The GAIN is the current setting loaded into the programmable pre-amplifier. The various allowable settings for GAIN and allowable voltages applied to the VINA and VINB inputs appear in Table 10-2.

The reference voltage for the amplifier and the ADC is 1.65V, generated via a voltage divider shown in Figure 10-2. Consequently, 1.65V is subtracted from the input voltage on VINA or VINB.

The maximum range of the ADC is  $\pm 1.25V$ , centered around the reference voltage, 1.65V. Hence, 1.25V appears in the denominator to scale the analog input accordingly.

Finally, the ADC presents a 14-bit, two's complement digital output. A 14-bit, two's complement number represents values between  $-2^{13}$  and  $2^{13}-1$ . Therefore, the quantity is scaled by 8192, or  $2^{13}$ .

See "Programmable Pre-Amplifier" to control the GAIN settings on the programmable pre-amplifier.

The reference design files provide more information on converting the voltage applied on VINA or VINB to a digital representation (see “[Related Resources](#),” page 81).

## Programmable Pre-Amplifier

The LTC6912-1 provides two independent inverting amplifiers with programmable gain. The purpose of the amplifier is to scale the incoming voltage on VINA or VINB so that it maximizes the conversion range of the DAC, namely  $1.65 \pm 1.25V$ .

### Interface

[Table 10-1](#) lists the interface signals between the FPGA and the amplifier. The SPI\_MOSI, SPI\_MISO, and SPI\_SCK signals are shared with other devices on the SPI bus. The AMP\_CS signal is the active-Low slave select input to the amplifier.

**Table 10-1: AMP Interface Signals**

Signal	FPGA Pin	Direction	Description
SPI_MOSI	T4	FPGA→AD	Serial data: Master Output, Slave Input. Presents 8-bit programmable gain settings, as defined in <a href="#">Table 10-2</a> .
AMP_CS	N7	FPGA→AMP	Active-Low chip-select. The amplifier gain is set when signal returns High.
SPI_SCK	U16	FPGA→AMP	Clock
AMP_SHDN	P7	FPGA→AMP	Active-High shutdown, reset
AMP_DOUT	E18	FPGA←AMP	Serial data. Echoes previous amplifier gain settings. Can be ignored in most applications.

### Programmable Gain

Each analog channel has an associated programmable gain amplifier (see [Figure 10-2](#)). Analog signals presented on the VINA or VINB inputs on header J7 are amplified relative to 1.65V. The 1.65V reference is generated using a voltage divider of the 3.3V voltage supply.

The gain of each amplifier is programmable from -1 to -100, as shown in [Table 10-2](#).

**Table 10-2: Programmable Gain Settings for Pre-Amplifier**

Gain	A3	A2	A1	A0	Input Voltage Range	
	B3	B2	B1	B0	Minimum	Maximum
0	0	0	0	0		
-1	0	0	0	1	0.4	2.9
-2	0	0	1	0	1.025	2.275
-5	0	0	1	1	1.4	1.9
-10	0	1	0	0	1.525	1.775
-20	0	1	0	1	1.5875	1.7125

Table 10-2: Programmable Gain Settings for Pre-Amplifier (Continued)

Gain	A3	A2	A1	A0	Input Voltage Range	
	B3	B2	B1	B0	Minimum	Maximum
-50	0	1	1	0	1.625	1.675
-100	0	1	1	1	1.6375	1.6625

## SPI Control Interface

Figure 10-3 highlights the SPI-based communications interface with the amplifier. The gain for each amplifier is sent as an 8-bit command word, consisting of two 4-bit fields. The most-significant bit, B3, is sent first.

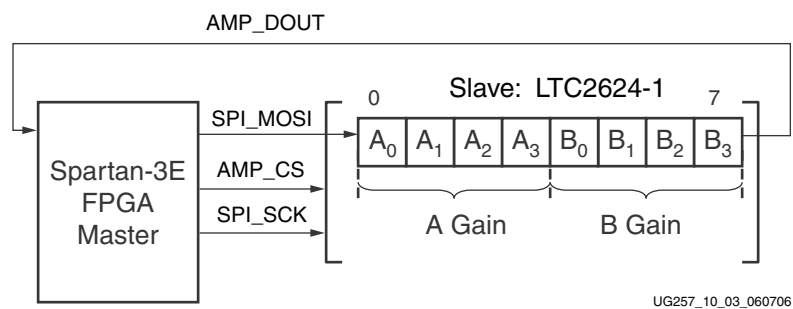


Figure 10-3: SPI Serial Interface to Amplifier

The AMP\_DOUT output from the amplifier echoes the previous gain settings. These values can be ignored for most applications.

The SPI bus transaction starts when the FPGA asserts AMP\_CS Low (see Figure 10-4). The amplifier captures serial data on SPI\_MOSI on the rising edge of the SPI\_SCK clock signal. The amplifier presents serial data on AMP\_DOUT on the falling edge of SPI\_SCK.

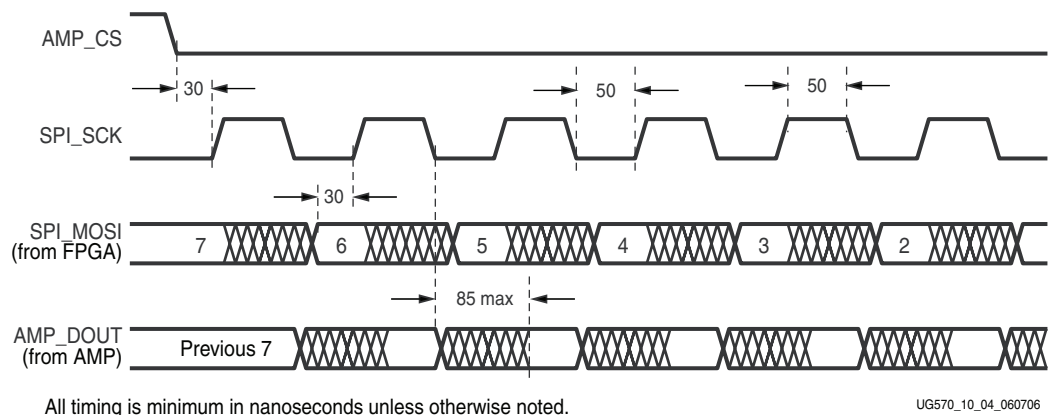


Figure 10-4: SPI Timing When Communicating with Amplifier

The amplifier interface is relatively slow, supporting only about a 10 MHz clock frequency.

## UCF Location Constraints

Figure 10-5 provides the User Constraint File (UCF) constraints for the amplifier interface, including the I/O pin assignment and I/O standard used.

```
NET "SPI_MOSI" LOC = "T4" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 6 ;
NET "AMP_CS" LOC = "N7" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 6 ;
NET "SPI_SCK" LOC = "U16" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;
NET "AMP_SHDN" LOC = "P7" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 6 ;
NET "AMP_DOUT" LOC = "E18" | IOSTANDARD = LVCMOS33 ;
```

UG570\_10\_05\_060706

Figure 10-5: UCF Location Constraints for the DAC Interface

## Analog to Digital Converter (ADC)

The LTC1407A-1 provides two ADCs. Both analog inputs are sampled simultaneously when the AD\_CONV signal is applied.

### Interface

Table 10-3 lists the interface signals between the FPGA and the ADC. The SPI\_MOSI, SPI\_MISO, and SPI\_SCK signals are shared with other devices on the SPI bus. The DAC\_CS signal is the active-Low slave select input to the DAC. The DAC\_CLR signal is the active-Low, asynchronous reset input to the DAC.

Table 10-3: ADC Interface Signals

Signal	FPGA Pin	Direction	Description
SPI_SCK	U16	FPGA→ADC	Clock
AD_CONV	P11	FPGA→ADC	Active-High shutdown and reset.
SPI_MISO	N10	FPGA←ADC	Serial data: Master Input, Serial Output. Presents the digital representation of the sample analog values as two 14-bit two's complement binary values.

### SPI Control Interface

Figure 10-6 provides an example SPI bus transaction to the ADC.

When the AD\_CONV signal goes High, the ADC simultaneously samples both analog channels. The results of this conversion are not presented until the next time AD\_CONV is asserted, a latency of one sample. The maxim sample rate is approximately 1.5 MHz.

The ADC presents the digital representation of the sampled analog values as a 14-bit, two's complement binary value.



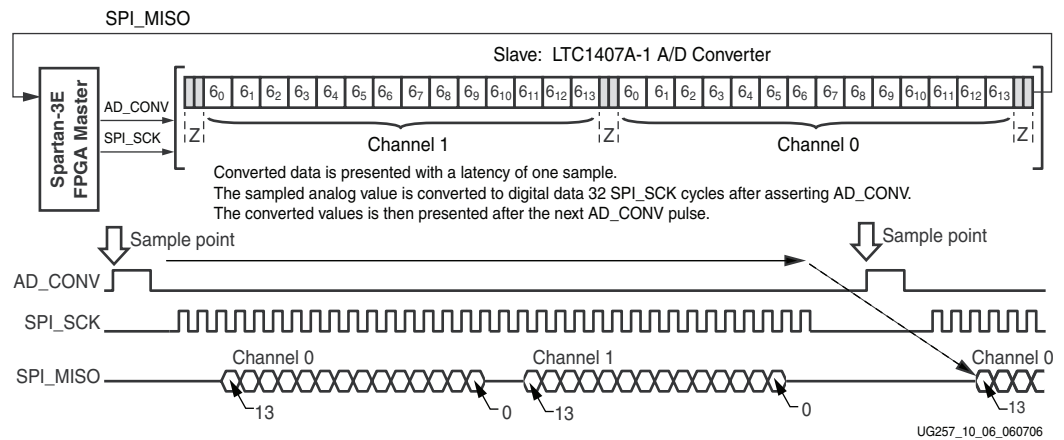


Figure 10-6: Analog-to-Digital Conversion Interface

Figure 10-7 shows detailed transaction timing. The AD\_CONV signal is not a traditional SPI slave select enable. Be sure to provide enough SPI\_SCK clock cycles so that the ADC leaves the SPI\_MISO signal in the high-impedance state. Otherwise, the ADC blocks communication to the other SPI peripherals. As shown in Figure 10-6, use a 34-cycle communications sequence. The ADC 3-states its data output for two clock cycles before and after each 14-bit data transfer.

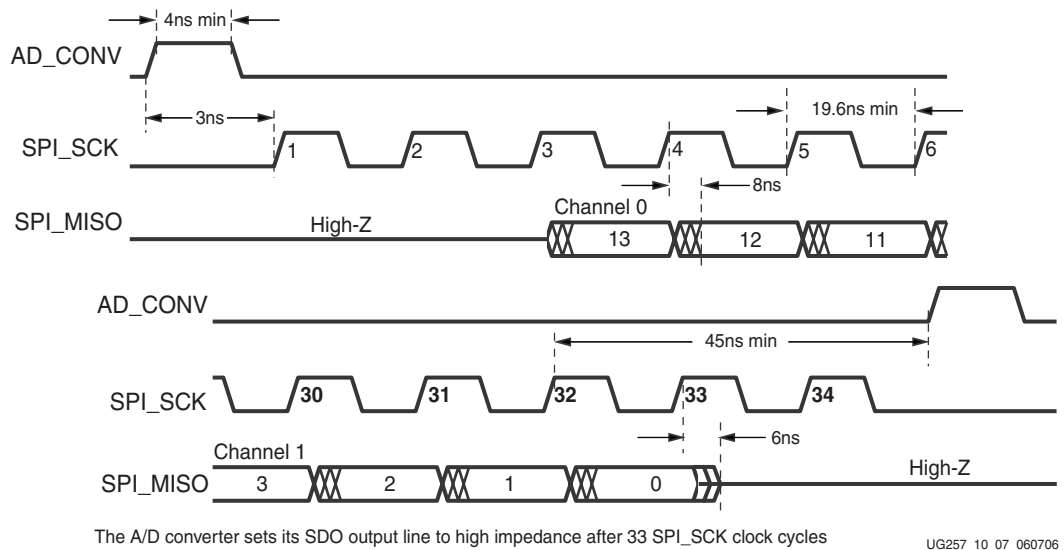


Figure 10-7: Detailed SPI Timing to ADC

## UCF Location Constraints

Figure 10-8 provides the User Constraint File (UCF) constraints for the amplifier interface, including the I/O pin assignment and I/O standard used.

```

NET "AD_CONV" LOC = "P11" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 6 ;
NET "SPI_SCK" LOC = "U16" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;
NET "SPI_MISO" LOC = "N10" | IOSTANDARD = LVCMOS33 ;
    
```

UG257\_10\_08\_061406

Figure 10-8: UCF Location Constraints for the ADC Interface

## Disable Other Devices on the SPI Bus to Avoid Contention

The SPI bus signals are shared by other devices on the board. It is vital that other devices are disabled when the FPGA communicates with the AMP or ADC to avoid bus contention. Table 10-4 provides the signals and logic values required to disable the other devices. Although the StrataFlash PROM is a parallel device, its least-significant data bit is shared with the SPI\_MISO signal. The Platform Flash PROM is only potentially enabled if the FPGA is set up for Master Serial mode configuration.

Table 10-4: Disable Other Devices on SPI Bus

Signal	Disabled Device	Disable Value
SPI_SS_B	SPI Serial Flash	1
AMP_CS	Programmable Pre-Amplifier	1
DAC_CS	DAC	1
SF_CE0	StrataFlash Parallel Flash PROM	1
FPGA_INIT_B	Platform Flash PROM	1

## Connecting Analog Inputs

Connect AC signals to VINA or VINB via a DC blocking capacitor.

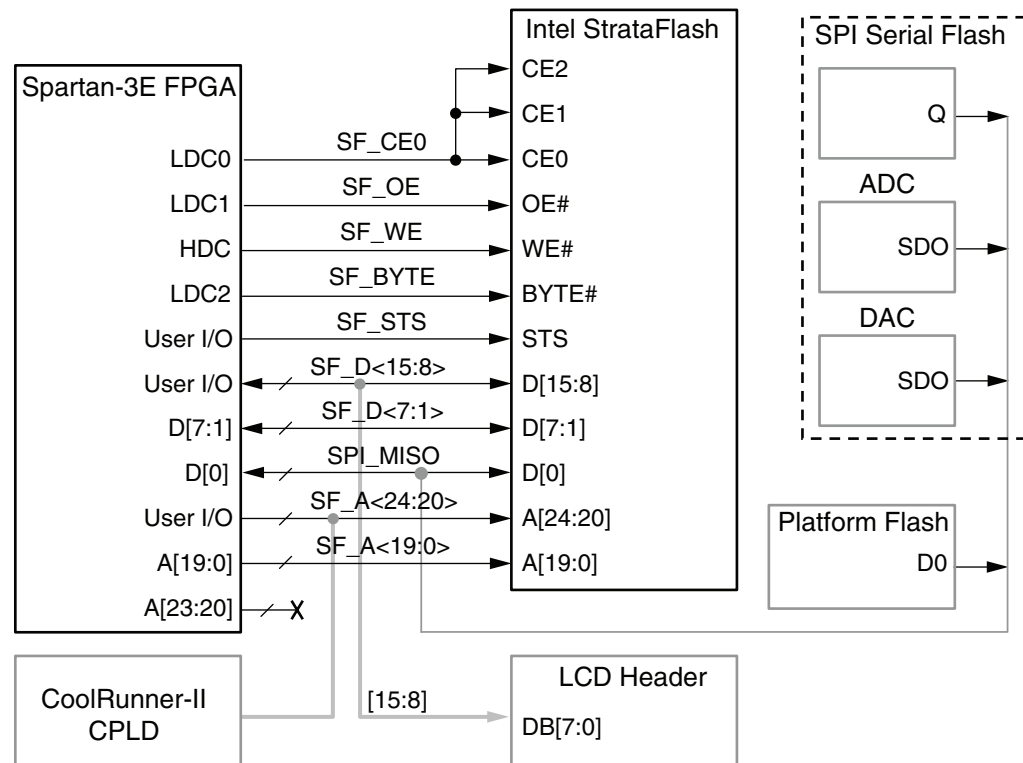
## Related Resources

- Amplifier and A/D Converter Control for the Spartan-3E Starter Kit (Reference Design)
- <http://www.xilinx.com/sp3e1600e>
- Xilinx PicoBlaze Soft Processor
- <http://www.xilinx.com/picoblaze>
- LTC6912 Dual Programmable Gain Amplifiers with Serial Digital Interface
- <http://www.linear.com/pc/downloadDocument.do?navId=H0,C1,C1154,C1009,C1121,P7596,D5359>
- LTC1407A-1 Serial 14-bit Simultaneous Sampling ADCs with Shutdown
- <http://www.linear.com/pc/downloadDocument.do?navId=H0,C1,C1155,C1001,C1158,P2420,D1295>



## Intel StrataFlash Parallel NOR Flash PROM

As shown in [Figure 11-1](#), the MicroBlaze Development Kit boards includes a 128 Mbit (16 Mbyte) Intel StrataFlash parallel NOR Flash PROM. As indicated, some of the StrataFlash connections are shared with other components on the board.



UG257\_11\_01\_062106

**Figure 11-1: Connections to Intel StrataFlash Flash Memory**

The StrataFlash PROM provides various functions:

- Stores a single FPGA configuration in the StrataFlash device.
- Stores two different FPGA configurations in the StrataFlash device and dynamically switch between the two using the Spartan-3E FPGA's MultiBoot feature.
- Stores and executes MicroBlaze processor code directly from the StrataFlash device.

- Stores MicroBlaze processor code in the StrataFlash device and shadows the code into the DDR memory before executing the code.
- Stores non-volatile data from the FPGA.

## StrataFlash Connections

[Table 11-1](#) shows the connections between the FPGA and the StrataFlash device.

Although the XC1600E FPGA only requires just slightly under 6 Mbits per configuration image, the FPGA-to-StrataFlash interface on the board support up to a 256 Mbit StrataFlash. The MicroBlaze Development Kit board ships with a 128 Mbit device. Address line SF\_A24 is not used.

In general, the StrataFlash device connects to the XC1600E to support Byte Peripheral Interface (BPI) configuration. The upper four address bits from the FPGA, A[23:19] do not connect directly to the StrataFlash device. Instead, the XC2C64 CPLD controls the pins during configuration. As described in [Table 11-1](#) and [Shared Connections](#), some of the StrataFlash connections are shared with other components on the board.

**Table 11-1: FPGA-to-StrataFlash Connections**

Category	StrataFlash Signal Name	FPGA Pin Number	Function
Address	SF_A24	A11	Shared with XC2C64A CPLD. The CPLD actively drives these pins during FPGA configuration, as described in <a href="#">Chapter 16, "XC2C64A CoolRunner-II CPLD"</a> . Also connects to FPGA user-I/O pins. SF_A24 is the same as FX2 connector signal FX2_IO<32>.
	SF_A23	N11	
	SF_A22	V12	
	SF_A21	V13	
	SF_A20	T12	
	SF_A19	V15	Connects to FPGA pins A[19:0] to support the BPI configuration.
	SF_A18	U15	
	SF_A17	T16	
	SF_A16	U18	
	SF_A15	T17	
	SF_A14	R18	
	SF_A13	T18	
	SF_A12	L16	
	SF_A11	L15	
	SF_A10	K13	
	SF_A9	K12	
	SF_A8	K15	
	SF_A7	K14	
	SF_A6	J17	
	SF_A5	J16	
	SF_A4	J15	
	SF_A3	J14	
	SF_A2	J12	
	SF_A1	J13	
SF_A0	H17		

Table 11-1: FPGA-to-StrataFlash Connections

Category	StrataFlash Signal Name	FPGA Pin Number	Function
Data	SF_D15	T8	Upper 8 bits of a 16-bit halfword when StrataFlash is configured for x16 data (SF_BYTE=High). Connects to FPGA user I/O.
	SF_D14	R8	
	SF_D13	P6	
	SF_D12	M16	
	SF_D11	M15	
	SF_D10	P17	
	SF_D9	R16	
	SF_D8	R15	
	SF_D7	N9	Upper 7 bits of a data byte or lower 8 bits of a 16-bit halfword. Connects to FPGA pins D[7:1] to support the BPI configuration.
	SF_D6	M9	
	SF_D5	R9	
	SF_D4	U9	
	SF_D3	V9	
	SF_D2	R10	
	SF_D1	P10	
	SPI_MISO	N10	Bit 0 of data byte and 16-bit halfword. Connects to FPGA pin D0/DIN to support the BPI configuration. Shared with other SPI peripherals and Platform Flash PROM.
Control	SF_CE0	D16	StrataFlash Chip Enable. Connects to FPGA pin LDC0 to support the BPI configuration.
	SF_WE	D17	StrataFlash Write Enable. Connects to FPGA pin HDC to support the BPI configuration.
	SF_OE	C18	StrataFlash Chip Enable. Connects to FPGA pin LDC1 to support the BPI configuration.
	SF_BYTE	C17	StrataFlash Byte Enable. Connects to FPGA pin LDC2 to support the BPI configuration. 0: x8 data 1: x16 data
	SF_STS	B18	StrataFlash Status signal. Connects to FPGA user-I/O pin.

## Shared Connections

Besides the connections to the FPGA, the StrataFlash memory shares some connections to other components.

### Character LCD

The LCD supports an 8-bit or a 4-bit data interface. The eight display data connections are also shared with the SF\_D<15:8> signals on the StrataFlash PROM. As shown in [Table 11-2](#), the FPGA controls access to the StrataFlash PROM or the character LCD using the SF\_CE0 and LCD\_RW signals.

**Table 11-2: FPGA Control for StrataFlash and LCD**

SF_CE0	LCD_RW	Function
1	1	The FPGA reads from the character LCD.
0	0	The FPGA accesses the StrataFlash PROM.

### Xilinx XC2C64A CPLD

The Xilinx XC2C64A CoolRunner CPLD controls the five upper StrataFlash address lines, SF\_A<24:20> during configuration. The four upper BPI-mode address lines from the FPGA, A<23:20> are not connected. Instead, four FPGA user-I/O pins connect to the StrataFlash PROM upper address lines SF\_A<23:0>. See [Chapter 16, “XC2C64A CoolRunner-II CPLD”](#) for more information.

The most-significant address line, SF\_A<24>, is not physically used on the 16 Mbyte StrataFlash PROM. It is provided for upward migration to a larger StrataFlash PROM in the same package footprint. Likewise, the SF\_A<24> signal is also connected to the FX2\_IO<32> signal on the FX2 expansion connector.

### SPI Data Line

The least-significant StrataFlash data line, SF\_D<0>, is shared with data output signals from serial SPI peripherals, SPI\_MISO, and the serial output from the Platform Flash PROM as shown in [Table 11-3](#). To avoid contention, the FPGA application must ensure that only one data source is active at any time.

**Table 11-3: Possible Contention on SPI\_MISO (SF\_D<0>) Data**

Condition	Function
FPGA_M2 = Low FPGA_M1 = Low FPGA_M0 = Low INIT_B = High	Platform Flash outputs data on D0.
SF_CE0 = Low SF_OE = Low	StrataFlash outputs data.
AD_CONV = High SPI_SCK	Serial data is clocked out of the A/D converter
DAC_CS = Low SPI_SCK	DAC outputs previous command in response to SPI_SCK transitions.



## UCF Location Constraints

### Address

Figure 11-2 provides the UCF constraints for the StrataFlash address pins, including the I/O pin assignment and the I/O standard used.

```

NET "SF_A<24>" LOC = "A11" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_A<23>" LOC = "N11" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_A<22>" LOC = "V12" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_A<21>" LOC = "V13" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_A<20>" LOC = "T12" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_A<19>" LOC = "V15" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_A<18>" LOC = "U15" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_A<17>" LOC = "T16" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_A<16>" LOC = "U18" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_A<15>" LOC = "T17" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_A<14>" LOC = "R18" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_A<13>" LOC = "T18" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_A<12>" LOC = "L16" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_A<11>" LOC = "L15" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_A<10>" LOC = "K13" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_A<9>" LOC = "K12" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_A<8>" LOC = "K15" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_A<7>" LOC = "K14" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_A<6>" LOC = "J17" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_A<5>" LOC = "J16" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_A<4>" LOC = "J15" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_A<3>" LOC = "J14" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_A<2>" LOC = "J12" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_A<1>" LOC = "J13" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_A<0>" LOC = "H17" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;

```

UG257\_11\_02\_060706

Figure 11-2: UCF Location Constraints for StrataFlash Address Inputs

### Data

Figure 11-3 provides the UCF constraints for the StrataFlash data pins, including the I/O pin assignment and the I/O standard used.

```

NET "SF_D<15>" LOC = "T8" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_D<14>" LOC = "R8" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_D<13>" LOC = "P6" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_D<12>" LOC = "M16" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_D<11>" LOC = "M15" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_D<10>" LOC = "P17" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_D<9>" LOC = "R16" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_D<8>" LOC = "R15" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_D<7>" LOC = "N9" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_D<6>" LOC = "M9" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_D<5>" LOC = "R9" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_D<4>" LOC = "U9" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_D<3>" LOC = "V9" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_D<2>" LOC = "R10" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_D<1>" LOC = "P10" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SPI_MISO" LOC = "N10" | IOSTANDARD = LVCMOS33 | DRIVE = 6 | SLEW = SLOW ;

```

UG257\_11\_03\_060706

Figure 11-3: UCF Location Constraints for StrataFlash Data I/Os

## Control

Figure 11-4 provides the UCF constraints for the StrataFlash control pins, including the I/O pin assignment and the I/O standard used.

```
NET "SF_BYTE" LOC = "C17" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_CEO" LOC = "D16" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_OE" LOC = "C18" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_STB" LOC = "B18" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_WE" LOC = "D17" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
```

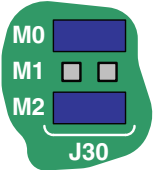
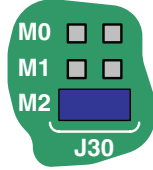
UG257\_11\_04\_060706

Figure 11-4: UCF Location Constraints for StrataFlash Control Pins

## Setting the FPGA Mode Select Pins

Set the FPGA configuration mode pins for either BPI Up or BPI down mode, as shown in Table 11-4. See

Table 11-4: Selecting BPI-Up or BPI-Down Configuration Modes (Header J30 in Chapter 4, “FPGA Configuration Options”, Figure 4-2)

Configuration Mode	Mode Pins M2:M1:M0	FPGA Configuration Image in StrataFlash	Jumper Settings
BPI Up	0:1:0	FPGA starts at address 0 and increments through address space. The CPLD controls address lines A[24:20] during BPI configuration.	
BPI Down	0:1:1	FPGA starts at address 0xFF_FFFF and decrements through address space. The CPLD controls address lines A[24:20] during BPI configuration.	

## Related Resources

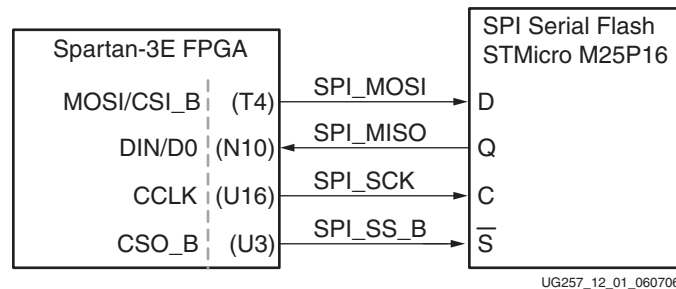
- Intel J3 StrataFlash Data Sheet  
<http://www.intel.com/design/flcomp/products/j3/techdocs.htm#datasheets>
- Application Note 827, Intel StrataFlash® Memory (J3) to Xilinx Spartan-3E FPGA Design Guide  
<http://www.intel.com/design/flcomp/applnots/307257.htm>



## SPI Serial Flash

The MicroBlaze Development Kit board includes a STMicroelectronics M25P16 16 Mbit SPI serial Flash, useful in a variety of applications. The SPI Flash provides an alternative means to configure the FPGA—a new feature of Spartan-3E FPGAs as shown in [Figure 12-1](#). The SPI Flash is also available to the FPGA after configuration for a variety of purposes, such as:

- Simple non-volatile data storage
- Storage for identifier codes, serial numbers, IP addresses, etc.
- Storage of MicroBlaze processor code that can be shadowed into DDR SDRAM.



**Figure 12-1: Spartan-3E FPGAs Have an Optional SPI Flash Configuration Interface**

**Table 12-1: SPI Flash Interface Signals**

Signal	FPGA Pin	Direction	Description
SPI_MOSI	T4	FPGA → SPI	Serial data: Master Output, Slave Input
SPI_MISO	N10	FPGA ← SPI	Serial data: Master Input, Slave Output
SPI_SCK	U16	FPGA → SPI	Clock
SPI_SS_B	U3	FPGA → SPI	Asynchronous, active-Low slave select input

## UCF Location Constraints

Figure 12-2 provides the UCF constraints for the SPI serial Flash PROM, including the I/O pin assignment and the I/O standard used.

```
# some connections shared with SPI Flash, DAC, ADC, and AMP
NET "SPI_MISO" LOC = "N10" | IOSTANDARD = LVCMOS33 ;
NET "SPI_MOSI" LOC = "T4" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 6 ;
NET "SPI_SCK" LOC = "U16" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 6 ;
NET "SPI_SS_B" LOC = "U3" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 6 ;
NET "SPI_ALT_CS_JP11" LOC = "R12" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 6 ;
```

UG257\_12\_02\_060806

Figure 12-2: UCF Location Constraints for SPI Flash Connections

## Configuring from SPI Flash

To configure the FPGA from SPI Flash, the FPGA mode select pins must be set appropriately and the SPI Flash must contain a valid configuration image.

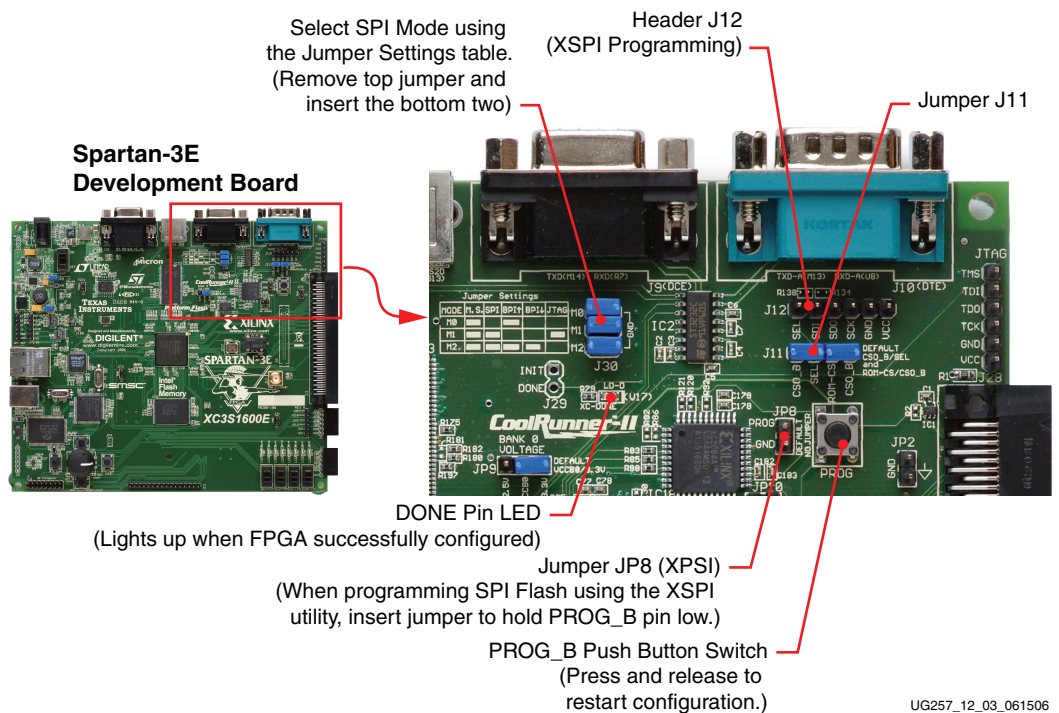
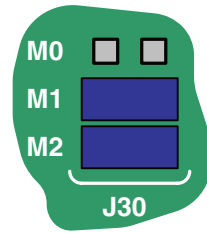


Figure 12-3: Configuration Options for SPI Mode

## Setting the FPGA Mode Select Pins

Set the FPGA configuration mode pins for SPI mode, as shown in Figure 12-4. The location of the configuration mode jumpers (J30) appears in Figure 12-3.



UG257\_12\_04\_061506

Figure 12-4: Set Mode Pins for SPI Mode

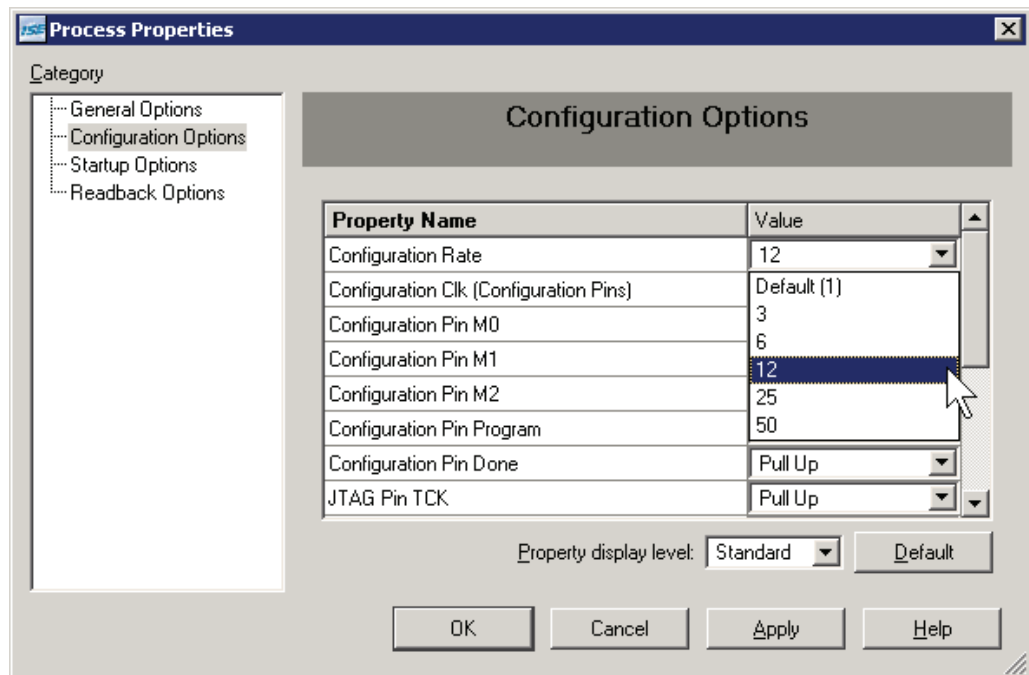
## Creating an SPI Serial Flash PROM File

The following steps describe how to format an FPGA bitstream for an SPI Serial Flash PROM.

### Setting the Configuration Clock Rate

The FPGA supports a 12 MHz configuration clock rate when connected to an M25P16 SPI serial Flash. Set the **Properties for Generate Programming File** so that the **Configuration Rate** is 12, as shown in Figure 12-5. See “Generating the FPGA Configuration Bitstream File” in the **FPGA Configuration Options** chapter for a more detailed description.

Regenerate the FPGA bitstream programming file with the new settings.

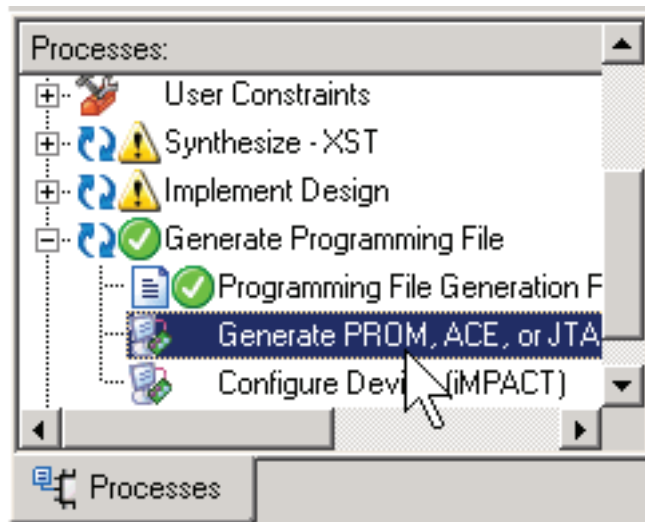


UG257\_12\_05\_060806

Figure 12-5: Set Configuration Rate to 12 MHz When Using the M25P16 SPI Flash

## Formatting an SPI Flash PROM File

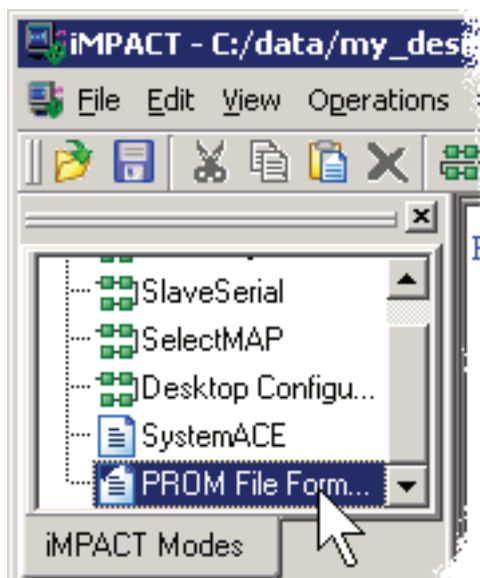
After generating the program file, double-click **Generate PROM, ACE, or JTAG File** to launch the iMPACT software, as shown in [Figure 12-6](#).



UG257\_12\_06\_060806

*Figure 12-6: Double-Click Generate PROM, ACE, or JTAG File*

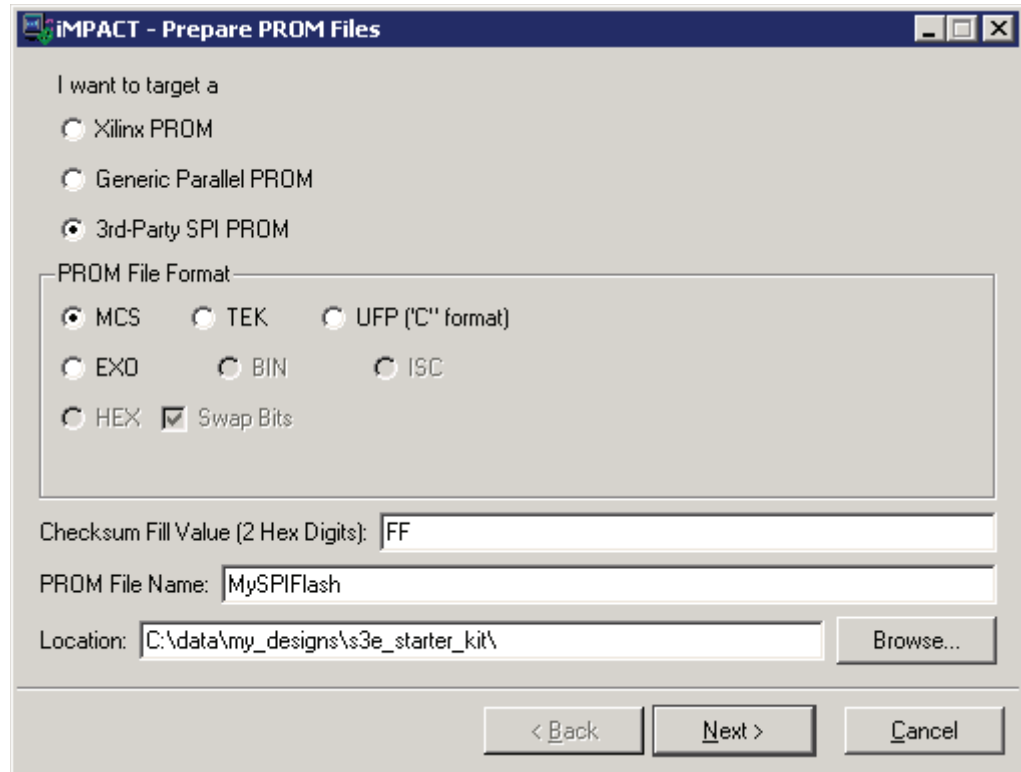
After iMPACT starts, double-click **PROM File Formatter**, as shown in [Figure 12-7](#).



UG257\_12\_07\_060806

*Figure 12-7: Double-Click PROM File Formatter*

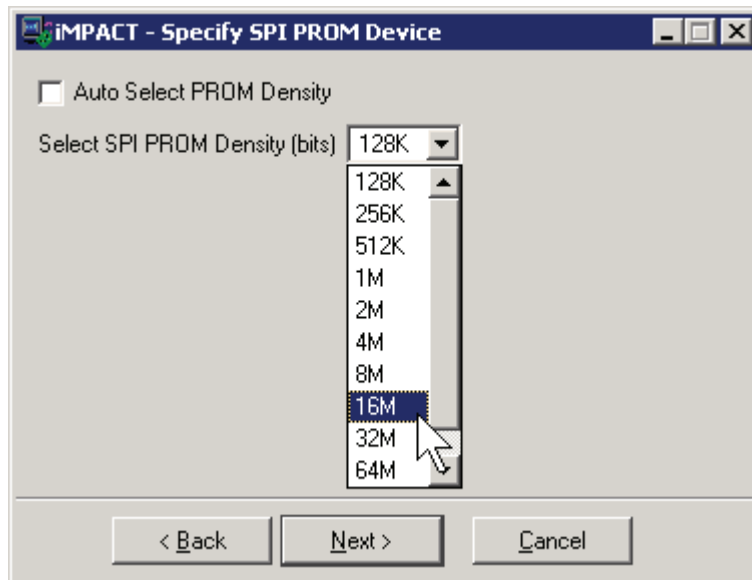
Choose **3rd Party SPI PROM** as the target PROM type, as shown in [Figure 12-8](#). Select from any of the PROM File Formats; the Intel Hex format (**MCS**) is popular. The PROM Formatter automatically swaps the bit direction as SPI Flash PROMs shift out the most-significant bit (MSB) first. Enter the **Location** of the directory and the **PROM File Name**. Click **Next** > when finished.



UG257\_12\_08\_060806

Figure 12-8: Choose the PROM Target Type, the, Data Format, and File Location

The Spartan-3E Starter Kit board has a 16 Mbit SPI serial Flash PROM. Select **16M** from the drop list, as shown in Figure 12-9. Click **Next >**.

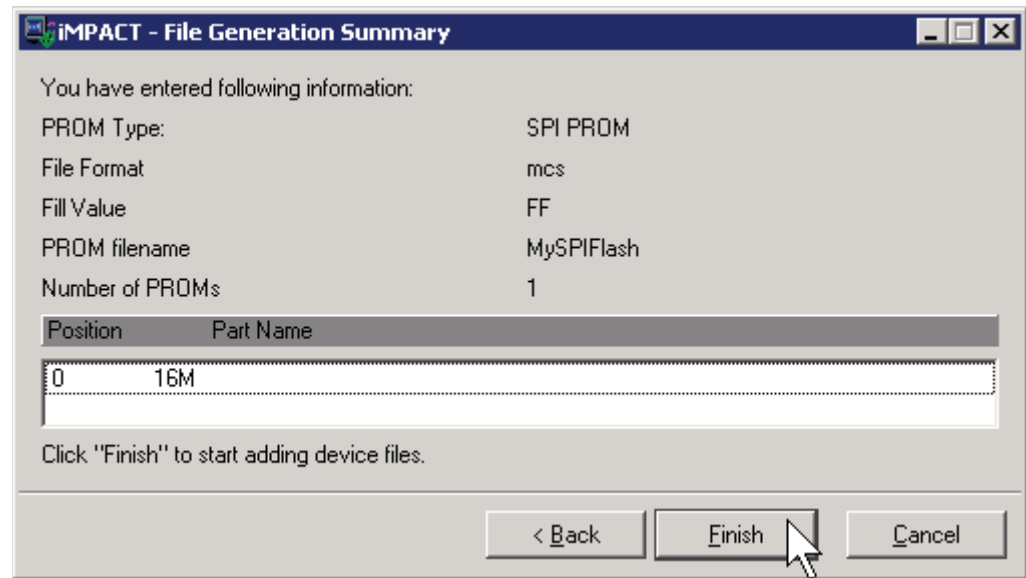


UG257\_12\_09\_060806

Figure 12-9: Choose 16M



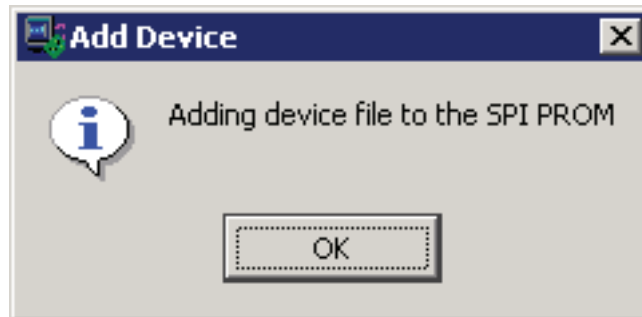
The PROM Formatter then echoes the settings, as shown in [Figure 12-10](#). Click **Finish**.



UG257\_12\_10\_060806

*Figure 12-10: Click Finish after Entering PROM Formatter Settings*

The PROM Formatter then prompts for the name(s) of the FPGA configuration bitstream file. As shown in [Figure 12-11](#), click **OK** to start selecting files. Select an FPGA bitstream file (\*.bit). Choose **No** after selecting the last FPGA file. Finally, click **OK** to continue.



UG257\_12\_11\_060806

*Figure 12-11: Enter FPGA Configuration Bitstream File(s)*

When PROM formatting is complete, the iMPACT software presents the present settings by showing the PROM, the select FPGA bitstream(s), and the amount of PROM space consumed by the bitstream. [Figure 12-12](#) shows an example for a single XC1600E FPGA bitstream stored in an XCF04S Platform Flash PROM.

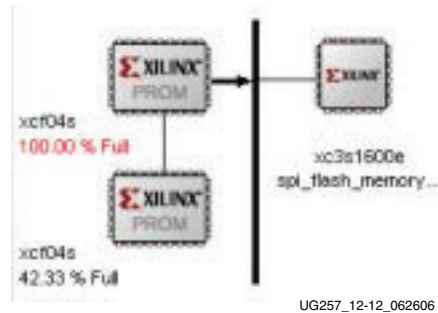


Figure 12-12: PROM Formatting Completed

To generate the actual PROM file, click **Operations** → **Generate File** as shown in Figure 12-13.

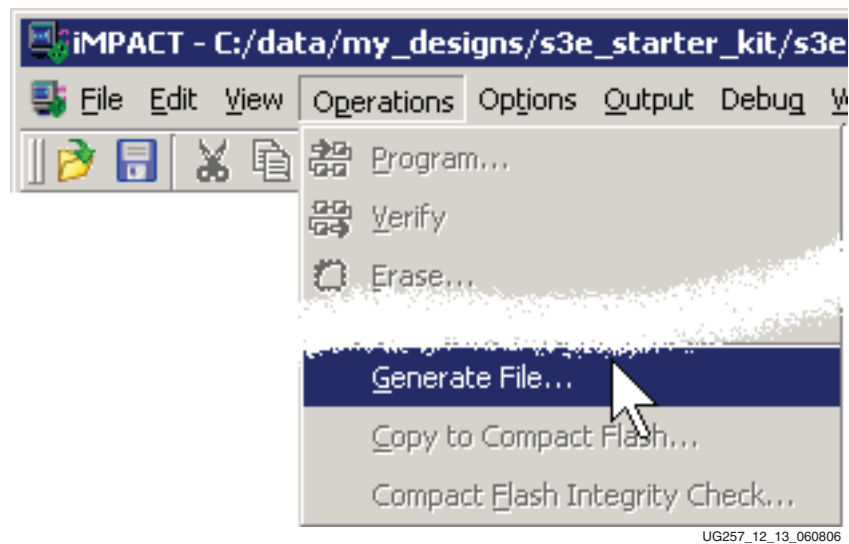
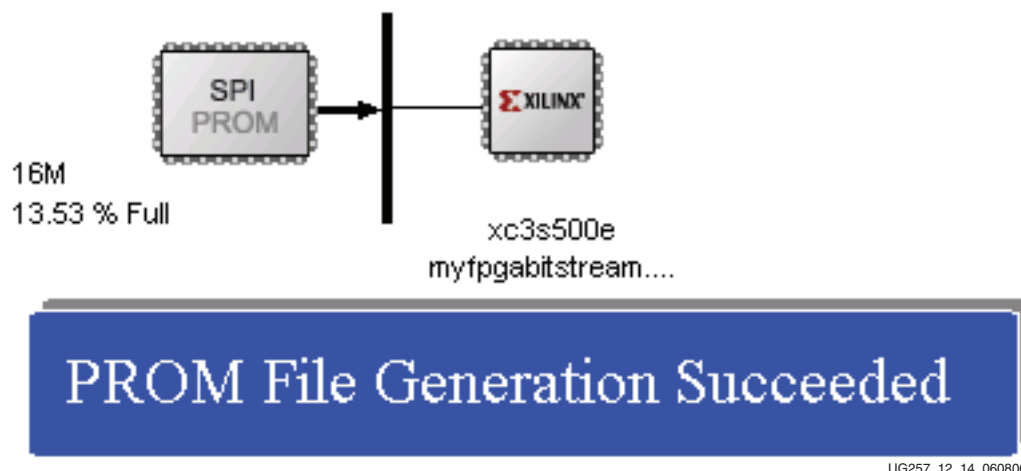


Figure 12-13: Click **Operations** → **Generate File** to Create the Formatted PROM File

As shown in Figure 12-14, the iMPACT software indicates that the PROM file was successfully created. The PROM Formatter creates an output file based on the settings shown in Figure 12-8. In this example, the output file is called **MySPIFlash.mcs**.



UG257\_12\_14\_060806

Figure 12-14: PROM File Formatter Succeeded

## Downloading the Design to SPI Flash

There are multiple methods to program the SPI Flash, as listed below.

- Use the XSPI programming software provided with XAPP445. Download the SPI Flash via the parallel port using a JTAG parallel programming cable (not provided with the kit).
- Use the PicoBlaze based SPI Flash programmer reference designs. Use a terminal emulator, such as Hyperlink, to download SPI Flash programming data via the PC's serial port to the FPGA. The embedded PicoBlaze processor then programs the attached SPI serial Flash. See "[Related Resources](#)," page 104.
- Via the FPGA's JTAG chain, use a JTAG tool to program the SPI Flash connected to the FPGA. See the link to the Universal Scan SPI Flash programming tutorial in "[Related Resources](#)," page 104.
- Additional programming support will be provided in the ISE 8.2i software.

## Downloading the SPI Flash using XSPI

The following steps describe how to download the SPI Flash PROM using the XSPI programming utility.

### Download and Install the XSPI Programming Utility

Download application note XAPP445 and the associated XSPI programming software (see "[Related Resources](#)," page 104). Unzip the XSPI software onto the PC.

### Attach a JTAG Parallel Programming Cable

The XSPI programming utility uses a JTAG parallel programming cable, such as:

- [Xilinx Parallel Cable IV](#) with flying leads
- Digilent JTAG3 programming cable

These cables are not provided with the MicroBlaze Development Kit board, but can be purchased separately, either from the Xilinx Online Store or from Digilent, Inc. (see "[Related Resources](#)," page 104).

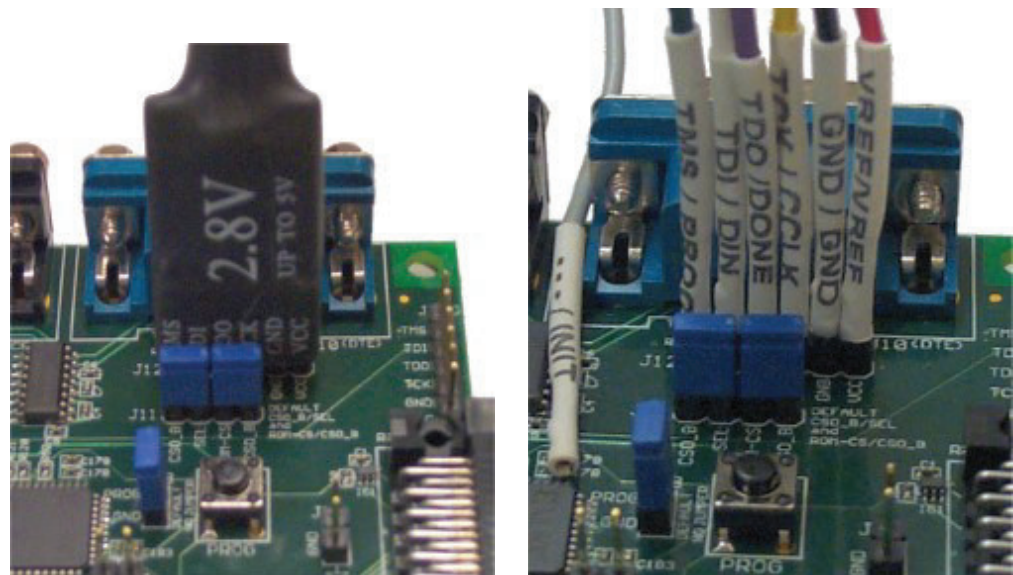
First, turn off the power on the Spartan-3E Starter Kit board.

If the USB cable is attached to the board, disconnect it. Simultaneously connecting both the USB cable and the parallel cable to the PC confuses the iMPACT software.

Connect one end of the JTAG parallel programming cable to the parallel printer port of the PC.

Connect the JTAG end of the cable to Header J12, as shown in [Figure 12-15\(a\)](#). The physical location of Header J12 is more clearly shown in [Figure 12-3, page 92](#). The J12 header connects directly to the SPI Flash pins; it is not connected to the JTAG chain.

The JTAG3 cable directly mounts to Header J12. The labels on the JTAG3 cable face toward the J11 jumpers. If using flying leads, they must be connected as shown in [Figure 12-15\(b\)](#) and [Table 12-2](#). Note the color coding for the leads. The gray INIT lead is left unconnected.



a) JTAG3 Parallel Connector

b) Parallel Cable III or Parallel Cable IV with Flying Leads

UG257\_12\_15\_060806

Figure 12-15: Attaching a JTAG Parallel Programming Cable to the Board

Table 12-2: Cable Connections to J12 Header

Cable and Labels	Connections					
J12 Header Label	SEL	SDI	SDO	SCK	GND	VCC
JTAG3 Cable Label	TMS	TDI	TDO	TCK	GND	VCC
Flying Leads Label	TMS/ PROG	TDI/ DIN	TDO/ DONE	TCK/ CCLK	GND/ GND	VREF/ VREF

### Insert Jumper on JP8 and Hold PROG\_B Low

The JTAG parallel programming cable directly accesses the SPI Flash pins. To avoid signal contention with the FPGA, ensure that the connecting FPGA pins are high-impedance. Force the FPGA's PROG\_B pin Low by installing a jumper on JP8, next to the PROG push button, as shown in [Figure 12-16](#). See [Figure 12-3, page 92](#) to locate jumper JP8 and surrounding landmarks.



a) No Jumper: FPGA Operational (default)    b) Jumper Installed: FPGA Held in Configuration State, I/Os in High Impedance

UG257\_12\_16\_061506

Figure 12-16: Installing the JP8 Jumper Holds the FPGA in Configuration State

Re-apply power to the MicroBlaze Development Kit board.

## Programming the SPI Flash with the XSPI Software

Open a command prompt or DOS box and change to the XSPI installation directory.

The XSPI installation software also includes a short user guide, in addition to XAPP445. Type **xspi** at the prompt to view quick help.

Type the following command at the prompt to program the SPI Flash using the SPI-formatted Flash file generated earlier. This verifies that the SPI Flash is indeed an M25P16 SPI Flash and then erases, programs, and finally verifies the Flash.

```
C:\xspi>xspi -spi_dev m25p16 -spi_epv -mcs -i MySPIFlash.mcs -o output.txt
```

A disclaimer notice appears on the screen. Press the Enter key to continue. The entire programming process takes slightly longer than a minute, as shown in [Figure 12-17](#).

```

-==< Press ENTER to accept notice and continue >==
Start   : Mon Feb 27 13:37:07 2006

==> Checking SPI device [STMicro_M25P16_ver_00100] ID code(s)
- density = [2097152] bytes
          = [16777216] bits
- mfg_code = [0x20]
- memory_type = [0x20]
- density_code = [0x15]

+-----+
| Device ID code(s) check ====> [ OK ] |
+-----+

=> Operation: Erase
=> Operation: Program and Verify using file [MySPIFlash.mcs]
Programmed [283776] of [283776] bytes (w/ polling)
Verified   [283776] of [283776] bytes (0 errors)

--> Total byte mismatches [0] (see [temp.txt])
Finish   : Mon Feb 27 13:38:22 2006
Elapsed clock time (00:01:15) = 75 seconds

```

UG257\_12\_17\_060806

Figure 12-17: Programming the M25P16 SPI Flash with the XSPI Programming Utility

After programming the SPI Flash, remove jumper JP8, as shown in Figure 12-16(a). If properly programmed, the FPGA then configures itself from the SPI Flash PROM and the DONE LED lights. The DONE LED is shown in Figure 12-3, page 92.

## Additional Design Details

Figure 12-18 provides additional details of the SPI Flash interface used on the Spartan-3E Starter Kit board. In most applications, this interface is as simple as that shown in Figure 12-1, page 91. The Spartan-3E Starter Kit board, however, supports a variety of configuration options and demonstrates additional Spartan-3E capabilities.

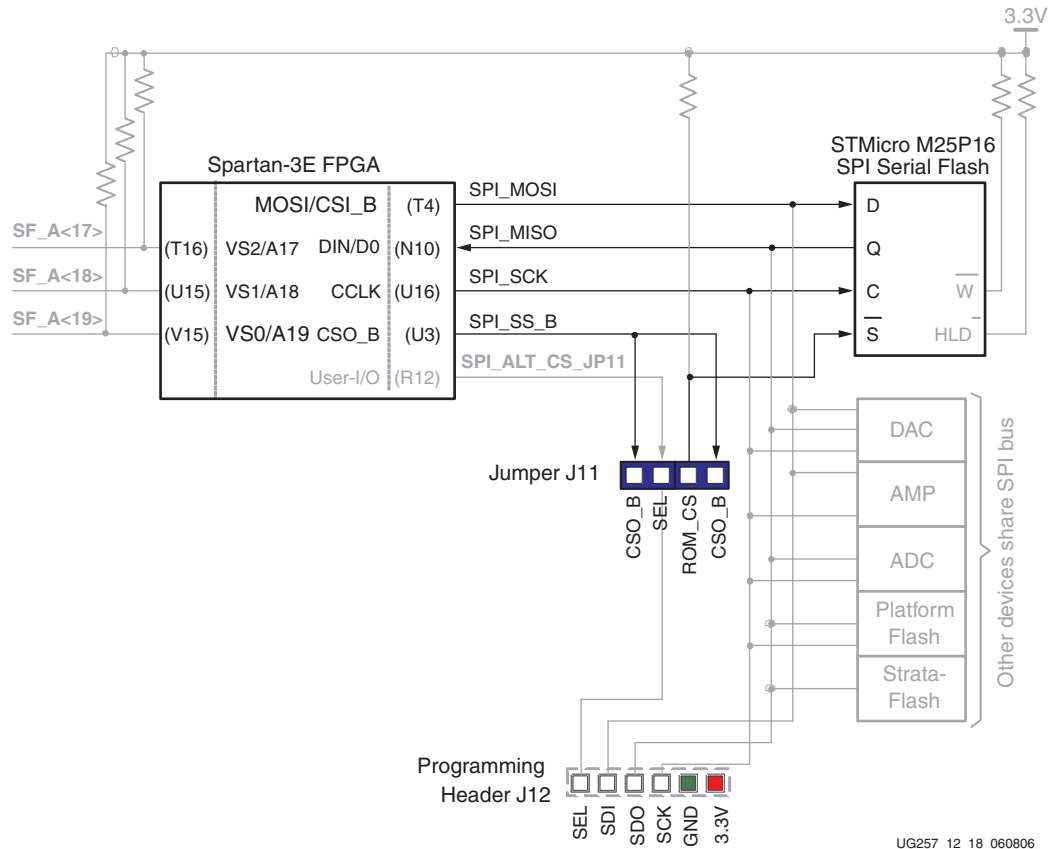


Figure 12-18: Additional SPI Flash Interface Design Details

## Shared SPI Bus with Peripherals

After configuration, the SPI Flash configuration pins are available to the application. On the Spartan-3E Starter Kit board, the SPI bus is shared by other SPI-capable peripheral devices, as shown in Figure 12-18. To access the SPI Flash memory after configuration, the FPGA application must disable the other devices on the shared SPI bus. Table 12-3 shows the signal names and disable values for the other devices.

Table 12-3: Disable Other Devices on SPI Bus

Signal	Disabled Device	Disable Value
DAC_CS	Digital-to-Analog Converter (DAC)	1
AMP_CS	Programmable Pre-Amplifier	1

Table 12-3: Disable Other Devices on SPI Bus

Signal	Disabled Device	Disable Value
AD_CONV	Analog-to-Digital Converter (ADC)	0
SF_CE0	StrataFlash Parallel Flash PROM	1
FPGA_INIT_B	Platform Flash PROM	1

## Other SPI Flash Control Signals

The M25P16 SPI Flash has two additional control inputs. The active-Low write protect input (W) and the active-Low bus hold input (HLD) are unused and pulled High via an external pull-up resistor.

## Variant Select Pins, VS[2:0]

When in SPI configuration mode, the FPGA samples the value on three pins, labeled VS[2:0], to determine which SPI read command to issue to the SPI Flash. For the M25P16 Flash, VS[2:0]=<1:1:1> issues the correct command sequence. The VS[2:0] pins are pulled High externally via pull-up resistors to 3.3V. The VS[2:0] pins are also parallel NOR Flash address lines A[19:17] in the FPGA's BPI configuration mode and these signals also connect to the StrataFlash parallel Flash PROM. After SPI configuration, the VS[2:0] pins become user-programmable I/O pins, allowing full access to the StrataFlash PROM, despite that the FPGA configured from SPI Flash.

## Jumper Block J11

In SPI configuration mode, the FPGA selects the attached SPI Flash by asserting the CSO\_B pin Low. On the MicroBlaze Development Kit board, the CSO\_B pin drives into the jumper J11 block. This jumper block provides the option to move the on-board SPI Flash to a different select line (SPI\_ALT\_CS\_JP11). This way, a different SPI Flash device can be tested by changing the JP11 jumper settings and connecting the alternate SPI Flash on Header JP12. By default, both jumpers are inserted on jumper block header J11.

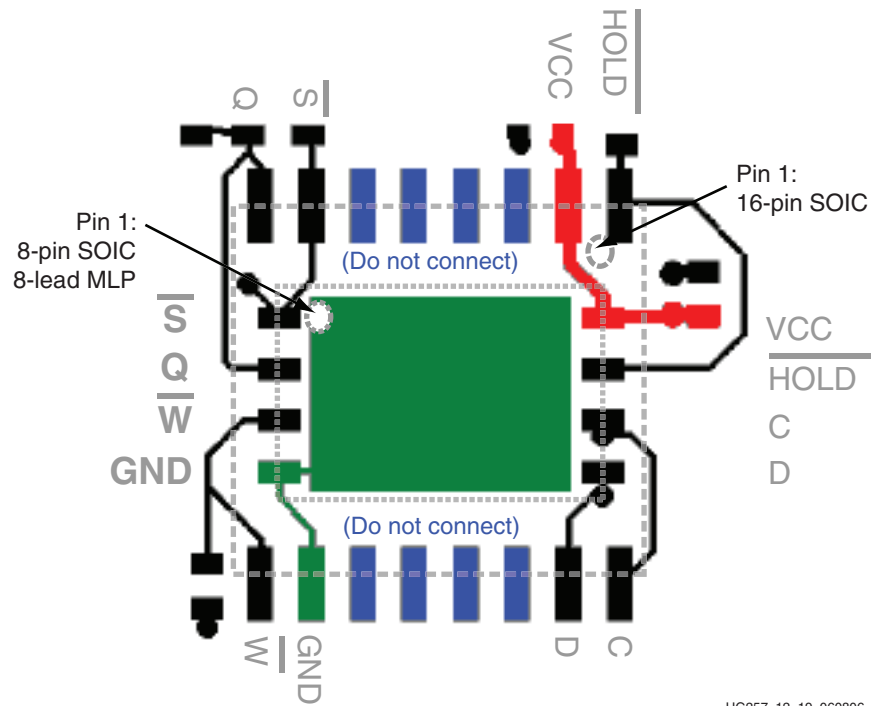
## Programming Header J12

As shown in [Figure 12-15, page 99](#), Header J12 accepts a JTAG parallel programming cable to program the on-board SPI Flash.

## Multi-Package Layout

STMicroelectronics was rather clever when they defined the package layout for the M25Pxx SPI serial Flash family. The Spartan-3E Starter Kit board supports all three of the package types used for the 16 Mbit device, as shown in [Figure 12-19](#). By default, the board ships with the 8-lead, 8x6 mm MLP package. The multi-package layout also supports the 8-pin SOIC package and the 16-pin SOIC package. Pin 1 for the 8-pin SOIC and MLP packages is located in the top-left corner. However, pin 1 for the 16-pin SOIC package is located in the top-right corner, because the package is rotated 90°. The 16-pin SOIC package also have four pins on each side that do not connect on the board. These pins must be left floating. Why support multiple packages? In a word, flexibility. The multi-package layout provides ...

- **Density migration between smaller- and larger-density SPI Flash PROMs.** Not all SPI Flash densities are available in all packages. The SPI Flash migration strategy follows nicely with the pinout migration provided by Xilinx FPGAs.
- **Consistent configuration PROM layout when migrating between FPGA densities.** The Spartan-3E FPGA's FG320 package footprint supports the XC3S500E, the XC3S1200E, and the XC3S1600E FPGA devices without modification. The SPI Flash multi-package layout allows comparable flexibility in the associated configuration PROM. Ship the optimally-sized SPI Flash memory for the FPGA mounted on the board.
- **Supply security.** If a certain SPI Flash density is not available in the desired package, switch to a different package style or to a different density to secure availability.



UG257\_12\_19\_060806

Figure 12-19: Multi-Package Layout for the STMicroelectronics M25Pxx Family



## Related Resources

- XAPP445: *Configuring Spartan-3E Xilinx FPGAs with SPI Flash Memories*
- [http://www.xilinx.com/xlnx/xweb/xil\\_publications\\_display.jsp?category=Application+Notes/FPGA+Features+and+Design/Configuration&show=xapp445.pdf](http://www.xilinx.com/xlnx/xweb/xil_publications_display.jsp?category=Application+Notes/FPGA+Features+and+Design/Configuration&show=xapp445.pdf)
- XSPI SPI Flash Programming Utility
- [http://www.xilinx.com/xlnx/xweb/xil\\_publications\\_display.jsp?category=Application+Notes/FPGA+Features+and+Design/Configuration&show=xapp445.pdf](http://www.xilinx.com/xlnx/xweb/xil_publications_display.jsp?category=Application+Notes/FPGA+Features+and+Design/Configuration&show=xapp445.pdf)
- [Xilinx Parallel Cable IV](#) with Flying Leads
- <http://www.xilinx.com/xlnx/xebiz/productview.jsp?sGlobalNavPick=&category=-19314>
- Digilent JTAG3 Programming Cable
- <http://www.digilentinc.com/Products/Catalog.cfm?Nav1=Products&Nav2=Cables&Cat=Cable>
- STMicroelectronics M25P16 SPI Serial Flash Data Sheet
- <http://www.st.com/stonline/books/pdf/docs/10027.pdf>
- AN1579: Compatibility between the SO8 Package and the MLP Package for the M25Pxx in Your Application
- <http://www.st.com/stonline/products/literature/an/9540.pdf>
- PicoBlaze SPI Serial Flash Programmer, via RS-232 (Reference Design)
- <http://www.xilinx.com/s3estarter>
- Using Serial Flash on the Spartan-3E Starter Kit Board (Reference Design)
- <http://www.xilinx.com/s3estarter>
- Universal Scan SPI Flash Programming via JTAG Training Video
- <http://www.ricreations.com/JTAG-Software-Downloads.htm>

## DDR SDRAM

The MicroBlaze Development Kit board includes a 512 Mbit (32M x 16) Micron Technology DDR SDRAM (MT46V32M16) with a 16-bit data interface, as shown in Figure 13-1. All DDR SDRAM interface pins connect to the FPGA's I/O Bank 3 on the FPGA. I/O Bank 3 and the DDR SDRAM are both powered by 2.5V, generated by an LTC3412 regulator from the board's 5V supply input. The 1.25V reference voltage, common to the FPGA and DDR SDRAM, is generated using a resistor voltage divider from the 2.5V rail.

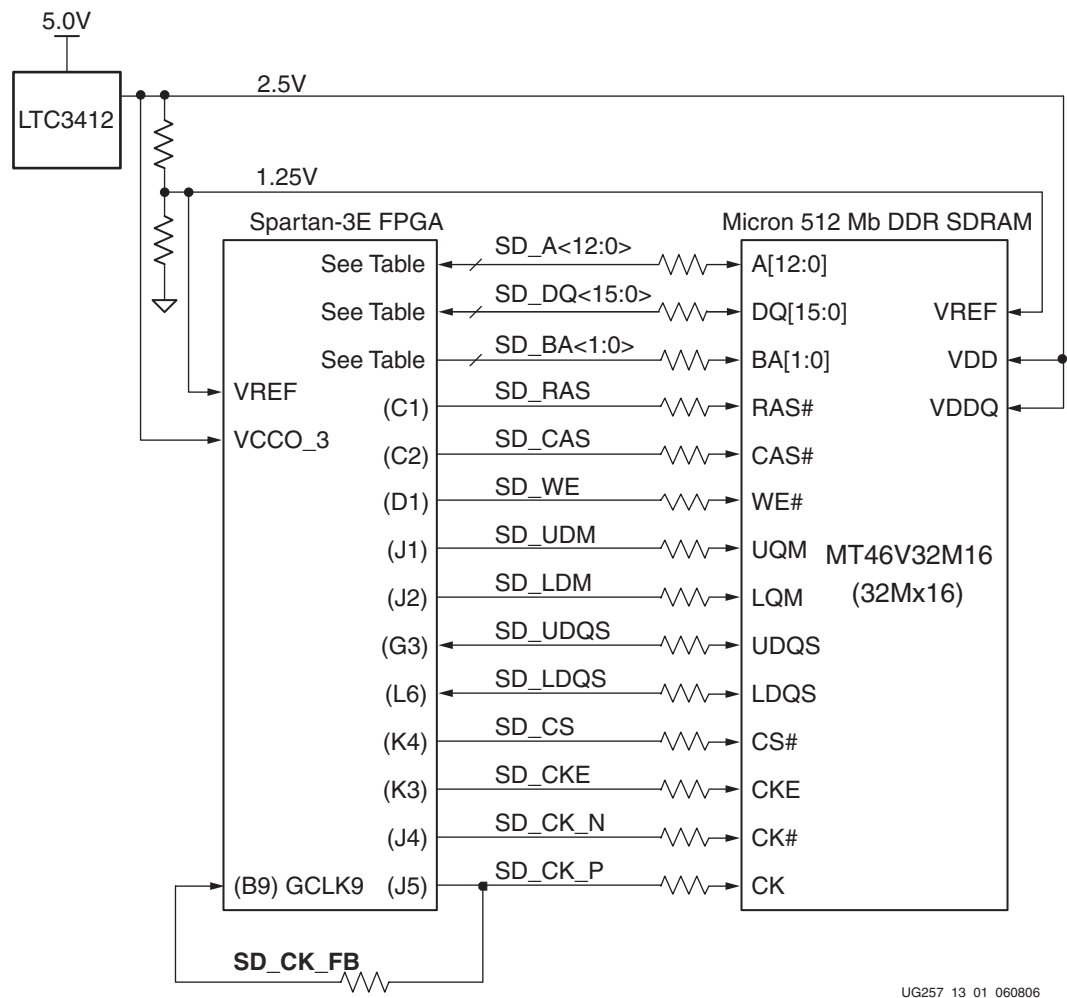


Figure 13-1: FPGA Interface to Micron 512 Mbit DDR SDRAM

All DDR SDRAM interface signals are terminated.

The differential clock pin SD\_CK\_P is fed back into FPGA pin B9 in I/O Bank 0 to have best access to one of the FPGA's Digital Clock Managers (DCMs). This path is required when using the MicroBlaze OPB DDR controller. The MicroBlaze OPB DDR SDRAM controller IP core documentation is also available from within the EDK 8.1i development software (see "Related Resources," page 109).

## DDR SDRAM Connections

Table 13-1 shows the connections between the FPGA and the DDR SDRAM.

Table 13-1: **FPGA-to-DDR SDRAM Connections**

Category	DDR SDRAM Signal Name	FPGA Pin Number	Function
Address	SD_A12	P2	Address inputs
	SD_A11	N5	
	SD_A10	T2	
	SD_A9	N4	
	SD_A8	H2	
	SD_A7	H1	
	SD_A6	H3	
	SD_A5	H4	
	SD_A4	E4	
	SD_A3	P1	
	SD_A2	R2	
	SD_A1	R3	
	SD_A0	T1	

**Table 13-1: FPGA-to-DDR SDRAM Connections (Continued)**

Category	DDR SDRAM Signal Name	FPGA Pin Number	Function
Data	SD_DQ15	H5	Data input/output
	SD_DQ14	H6	
	SD_DQ13	G5	
	SD_DQ12	G6	
	SD_DQ11	F2	
	SD_DQ10	F1	
	SD_DQ9	E1	
	SD_DQ8	E2	
	SD_DQ7	M6	
	SD_DQ6	M5	
	SD_DQ5	M4	
	SD_DQ4	M3	
	SD_DQ3	L4	
	SD_DQ2	L3	
	SD_DQ1	L1	
	SD_DQ0	L2	
Control	SD_BA1	K6	Bank address inputs
	SD_BA0	K5	
	SD_RAS	C1	Command inputs
	SD_CAS	C2	
	SD_WE	D1	
	SD_CK_N	J4	Differential clock input
	SD_CK_P	J5	
	SD_CKE	K3	Active-High clock enable input
	SD_CS	K4	Active-Low chip select input
	SD_UDM	J1	Data Mask. Upper and Lower data masks
	SD_LDM	J2	
	SD_UDQS	G3	Data Strobe. Upper and Lower data strobes
	SD_LDQS	L6	
	SD_CK_FB	B9	SDRAM clock feedback into top DCM within FPGA. Used by some DDR SDRAM controller cores

## UCF Location Constraints

### Address

Figure 13-2 provides the User Constraint File (UCF) constraints for the DDR SDRAM address pins, including the I/O pin assignment and the I/O standard used.

```

NET "SD_A<12>" LOC = "P2" | IOSTANDARD = SSTL2_I ;
NET "SD_A<11>" LOC = "N5" | IOSTANDARD = SSTL2_I ;
NET "SD_A<10>" LOC = "T2" | IOSTANDARD = SSTL2_I ;
NET "SD_A<9>" LOC = "N4" | IOSTANDARD = SSTL2_I ;
NET "SD_A<8>" LOC = "H2" | IOSTANDARD = SSTL2_I ;
NET "SD_A<7>" LOC = "H1" | IOSTANDARD = SSTL2_I ;
NET "SD_A<6>" LOC = "H3" | IOSTANDARD = SSTL2_I ;
NET "SD_A<5>" LOC = "H4" | IOSTANDARD = SSTL2_I ;
NET "SD_A<4>" LOC = "E4" | IOSTANDARD = SSTL2_I ;
NET "SD_A<3>" LOC = "P1" | IOSTANDARD = SSTL2_I ;
NET "SD_A<2>" LOC = "R2" | IOSTANDARD = SSTL2_I ;
NET "SD_A<1>" LOC = "R3" | IOSTANDARD = SSTL2_I ;
NET "SD_A<0>" LOC = "T1" | IOSTANDARD = SSTL2_I ;

```

UG257\_13\_02\_060806

Figure 13-2: UCF Location Constraints for DDR SDRAM Address Inputs

### Data

Figure 13-3 provides the User Constraint File (UCF) constraints for the DDR SDRAM data pins, including the I/O pin assignment and I/O standard used.

```

NET "SD_DQ<15>" LOC = "H5" | IOSTANDARD = SSTL2_I ;
NET "SD_DQ<14>" LOC = "H6" | IOSTANDARD = SSTL2_I ;
NET "SD_DQ<13>" LOC = "G5" | IOSTANDARD = SSTL2_I ;
NET "SD_DQ<12>" LOC = "G6" | IOSTANDARD = SSTL2_I ;
NET "SD_DQ<11>" LOC = "F2" | IOSTANDARD = SSTL2_I ;
NET "SD_DQ<10>" LOC = "F1" | IOSTANDARD = SSTL2_I ;
NET "SD_DQ<9>" LOC = "E1" | IOSTANDARD = SSTL2_I ;
NET "SD_DQ<8>" LOC = "E2" | IOSTANDARD = SSTL2_I ;
NET "SD_DQ<7>" LOC = "M6" | IOSTANDARD = SSTL2_I ;
NET "SD_DQ<6>" LOC = "M5" | IOSTANDARD = SSTL2_I ;
NET "SD_DQ<5>" LOC = "M4" | IOSTANDARD = SSTL2_I ;
NET "SD_DQ<4>" LOC = "M3" | IOSTANDARD = SSTL2_I ;
NET "SD_DQ<3>" LOC = "L4" | IOSTANDARD = SSTL2_I ;
NET "SD_DQ<2>" LOC = "L3" | IOSTANDARD = SSTL2_I ;
NET "SD_DQ<1>" LOC = "L1" | IOSTANDARD = SSTL2_I ;
NET "SD_DQ<0>" LOC = "L2" | IOSTANDARD = SSTL2_I ;

```

UG257\_13\_03\_060806

Figure 13-3: UCF Location Constraints for DDR SDRAM Data I/Os

## Control

Figure 13-4 provides the User Constraint File (UCF) constraints for the DDR SDRAM control pins, including the I/O pin assignment and the I/O standard used.

```

NET "SD_BA<0>" LOC = "K5" | IOSTANDARD = SSTL2_I ;
NET "SD_BA<1>" LOC = "K6" | IOSTANDARD = SSTL2_I ;
NET "SD_CAS" LOC = "C2" | IOSTANDARD = SSTL2_I ;
NET "SD_CK_N" LOC = "J4" | IOSTANDARD = SSTL2_I ;
NET "SD_CK_P" LOC = "J5" | IOSTANDARD = SSTL2_I ;
NET "SD_CKE" LOC = "K3" | IOSTANDARD = SSTL2_I ;
NET "SD_CS" LOC = "K4" | IOSTANDARD = SSTL2_I ;

NET "SD_LDM" LOC = "J2" | IOSTANDARD = SSTL2_I ;
NET "SD_LDQS" LOC = "L6" | IOSTANDARD = SSTL2_I ;
NET "SD_RAS" LOC = "C1" | IOSTANDARD = SSTL2_I ;
NET "SD_UDM" LOC = "J1" | IOSTANDARD = SSTL2_I ;
NET "SD_UDQS" LOC = "G3" | IOSTANDARD = SSTL2_I ;
NET "SD_WE" LOC = "D1" | IOSTANDARD = SSTL2_I ;
# Path to allow connection to top DCM connection
NET "SD_CK_FB" LOC = "B9" | IOSTANDARD = LVCMOS33 ;

```

UG257\_13\_04\_060806

Figure 13-4: UCF Location Constraints for DDR SDRAM Control Pins

## Reserve FPGA VREF Pins

Five pins in I/O Bank 3 are dedicated as voltage reference inputs, VREF. These pins cannot be used for general-purpose I/O in a design. Prohibit the software from using these pins with the constraints provided in Figure 13-5.

```

# Prohibit VREF pins
CONFIG PROHIBIT = D2;
CONFIG PROHIBIT = G4;
CONFIG PROHIBIT = J6;
CONFIG PROHIBIT = L5;
CONFIG PROHIBIT = R4;

```

UG257\_13\_05\_060806

Figure 13-5: UCF Location Constraints for StrataFlash Control Pins

## Related Resources

- Xilinx Embedded Design Kit (EDK)  
[http://www.xilinx.com/ise/embedded\\_design\\_prod/platform\\_studio.htm](http://www.xilinx.com/ise/embedded_design_prod/platform_studio.htm)
- MT46V32M16 (32M x 16) DDR SDRAM Data Sheet  
<http://download.micron.com/pdf/datasheets/dram/ddr/512MBDDRx4x8x16.pdf>
- MicroBlaze OPB Double Data Rate (DDR) SDRAM Controller (v2.00b)  
[http://www.xilinx.com/bvdocs/ipcenter/data\\_sheet/opb\\_ddr.pdf](http://www.xilinx.com/bvdocs/ipcenter/data_sheet/opb_ddr.pdf)



# 10/100 Ethernet Physical Layer Interface

The MicroBlaze Development Kit board includes a Standard Microsystems LAN83C185 10/100 Ethernet physical layer (PHY) interface and an RJ-45 connector, as shown in Figure 14-1. With an Ethernet Media Access Controller (MAC) implemented in the FPGA, the board can optionally connect to a standard Ethernet network. All timing is controlled from an on-board 25 MHz crystal oscillator.

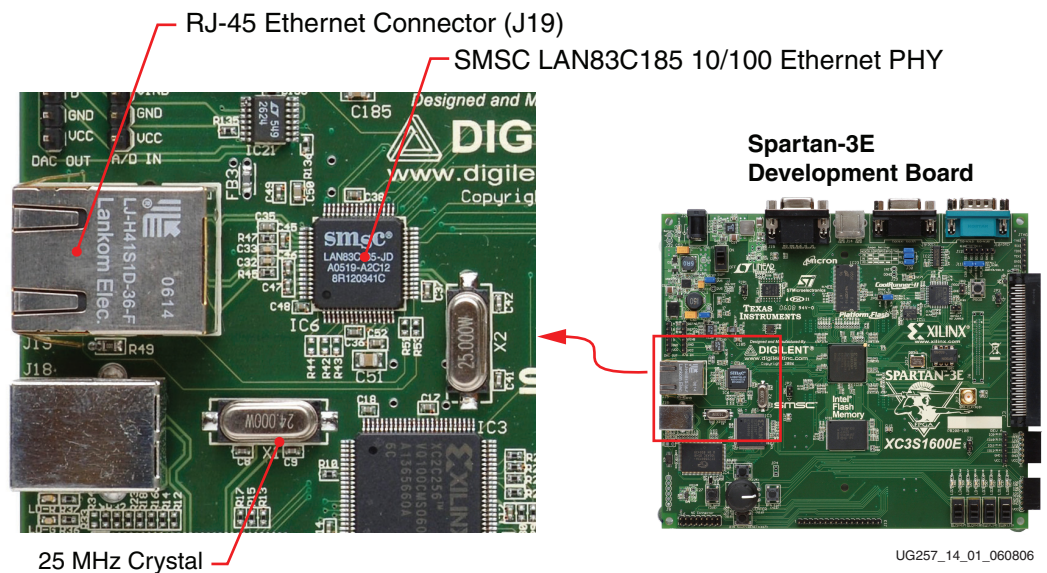


Figure 14-1: 10/100 Ethernet PHY with RJ-45 Connector



## Ethernet PHY Connections

The FPGA connects to the LAN83C185 Ethernet PHY using a standard Media Independent Interface (MII), as shown in [Figure 14-2](#). A more detailed description of the interface signals, including the FPGA pin number, appears in [Table 14-1](#).

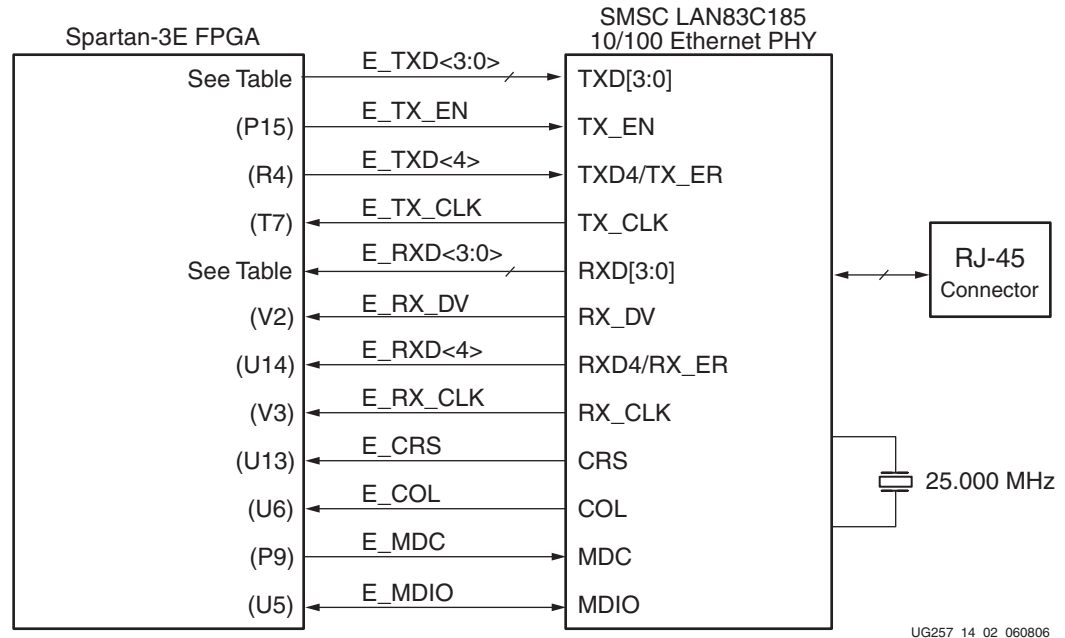


Figure 14-2: FPGA Connects to Ethernet PHY via MII

Table 14-1: FPGA Connections to the LAN83C185 Ethernet PHY

Signal Name	FPGA Pin Number	Function
E_TXD<4>	R6	Transmit Data to the PHY. E_TXD<4> is also the MII Transmit Error.
E_TXD<3>	T5	
E_TXD<2>	R5	
E_TXD<1>	T15	
E_TXD<0>	R11	
E_TX_EN	P15	Transmit Enable.
E_TX_CLK	T7	Transmit Clock. 25 MHz in 100Base-TX mode, and 2.5 MHz in 10Base-T mode.
E_RXD<4>	U14	Receive Data from PHY.
E_RXD<3>	V14	
E_RXD<2>	U11	
E_RXD<1>	T11	
E_RXD<0>	V8	
E_RX_DV	V2	Receive Data Valid.

Table 14-1: FPGA Connections to the LAN83C185 Ethernet PHY (Continued)

Signal Name	FPGA Pin Number	Function
E_RX_CLK	V3	Receive Clock. 25 MHz in 100Base-TX mode, and 2.5 MHz in 10Base-T mode.
E_CRS	U13	Carrier Sense
E_COL	U6	MII Collision Detect.
E_MDC	P9	Management Clock. Serial management clock.
E_MDIO	U5	Management Data Input/Output.

## MicroBlaze Ethernet IP Cores

The Ethernet PHY is primarily intended for use with MicroBlaze applications. As such, an Ethernet MAC is part of the EDK Platform Studio's Base System Builder. Both the full Ethernet MAC and the *Lite* version are available for evaluation, as shown in Figure 14-3. The Ethernet Lite MAC controller core uses fewer FPGA resources and is ideal for applications that do not require support for interrupts, back-to-back data transfers, and statistics counters.

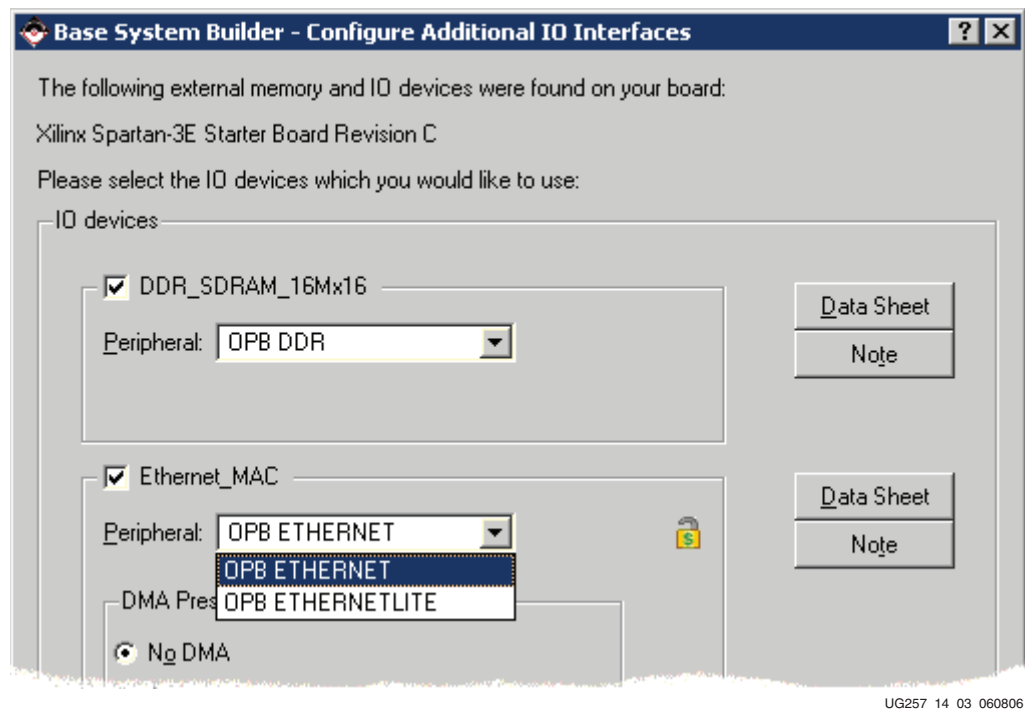


Figure 14-3: Ethernet MAC IP Cores for the Spartan-3E Starter Kit Board

The Ethernet MAC core requires design constraints to meet the required performance. Refer to the OPB Ethernet MAC data sheet (v1.02) for details. The OPB bus clock frequency must be 65 MHz or higher for 100 Mbps Ethernet operations and 6.5 MHz or faster for 10 Mbps Ethernet operations.

The hardware evaluation versions of the Ethernet MAC cores operate for approximately eight hours in silicon before timing out. To order the full version of the core, visit the Xilinx website at:

[http://www.xilinx.com/ipcenter/processor\\_central/processor\\_ip/10-100emac/10-100emac\\_order\\_register.htm](http://www.xilinx.com/ipcenter/processor_central/processor_ip/10-100emac/10-100emac_order_register.htm)

## UCF Location Constraints

Figure 14-4 provides the UCF constraints for the 10/100 Ethernet PHY interface, including the I/O pin assignment and the I/O standard used.

```

NET "E_COL"      LOC = "U6" | IOSTANDARD = LVCMOS33 ;
NET "E_CRS"      LOC = "U13" | IOSTANDARD = LVCMOS33 ;
NET "E_MDC"      LOC = "P9" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;
NET "E_MDIO"     LOC = "U5" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;
NET "E_RX_CLK"   LOC = "V3" | IOSTANDARD = LVCMOS33 ;
NET "E_RX_DV"    LOC = "V2" | IOSTANDARD = LVCMOS33 ;
NET "E_RXD<0>"  LOC = "V8" | IOSTANDARD = LVCMOS33 ;
NET "E_RXD<1>"  LOC = "T11" | IOSTANDARD = LVCMOS33 ;
NET "E_RXD<2>"  LOC = "U11" | IOSTANDARD = LVCMOS33 ;
NET "E_RXD<3>"  LOC = "V14" | IOSTANDARD = LVCMOS33 ;
NET "E_RXD<4>"  LOC = "U14" | IOSTANDARD = LVCMOS33 ;
NET "E_TX_CLK"   LOC = "T7" | IOSTANDARD = LVCMOS33 ;
NET "E_TX_EN"    LOC = "P15" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;
NET "E_TXD<0>"  LOC = "R11" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;
NET "E_TXD<1>"  LOC = "T15" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;
NET "E_TXD<2>"  LOC = "R5" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;
NET "E_TXD<3>"  LOC = "T5" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;
NET "E_TXD<4>"  LOC = "R6" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;

```

UG257\_14\_04\_060806

Figure 14-4: UCF Location Constraints for 10/100 Ethernet PHY Inputs

## Related Resources

- Standard Microsystems SMSC LAN83C185 10/100 Ethernet PHY
- <http://www.smsc.com/main/catalog/lan83c185.html>
- Xilinx OPB Ethernet Media Access Controller (EMAC) (v1.02a)
- [http://www.xilinx.com/bvdocs/ipcenter/data\\_sheet/opb\\_ethernet.pdf](http://www.xilinx.com/bvdocs/ipcenter/data_sheet/opb_ethernet.pdf)
- Xilinx OPB Ethernet Lite Media Access Controller (v1.01a)
- The Ethernet Lite MAC controller core uses fewer FPGA resources and is ideal for applications that do not require support for interrupts, back-to-back data transfers, and statistics counters.
- [http://www.xilinx.com/bvdocs/ipcenter/data\\_sheet/opb\\_ethernetlite.pdf](http://www.xilinx.com/bvdocs/ipcenter/data_sheet/opb_ethernetlite.pdf)
- EDK 8.1i Documentation
- [http://www.xilinx.com/ise/embedded/edk\\_docs.htm](http://www.xilinx.com/ise/embedded/edk_docs.htm)

## Expansion Connectors

The MicroBlaze Development Kit board provides a variety of expansion connectors for easy interface flexibility to other off-board components. The board includes the following I/O expansion headers (see [Figure 15-1](#)):

- A Hirose 100-pin edge connector with 43 associated FPGA user-I/O pins, including up to 15 differential LVDS I/O pairs and two Input-only pairs
- Three 6-pin Peripheral Module connections
- Landing pads for an Agilent or Tektronix connectorless probe

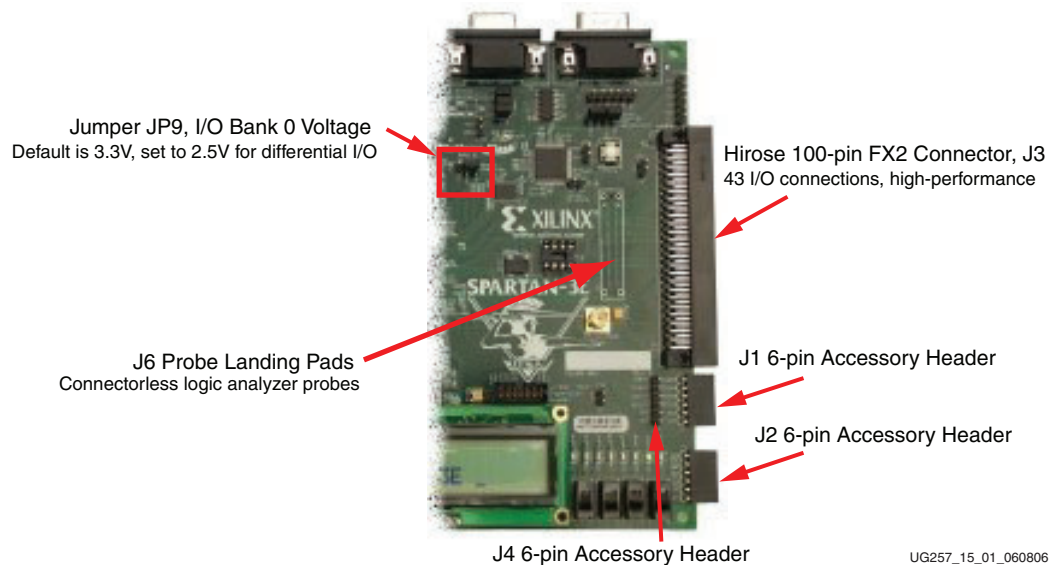


Figure 15-1: Expansion Headers

UG257\_15\_01\_060806

### Hirose 100-pin FX2 Edge Connector (J3)

A 100-pin edge connector is located along the right edge of the board (see [Figure 15-1](#)). This connector is a Hirose FX2-100P-1.27DS header with 1.27 mm pitch. Throughout the documentation, this connector is called the FX2 connector.

As shown in [Figure 15-2](#), 43 FPGA I/O pins interface to the FX2 connector. All but five of these pins are true, bidirectional I/O pins capable of driving or receiving signals. Five pins, FX2\_IP<38:35> and FX2\_IP<40> are Input-only pins on the FPGA. These pins are highlighted in light green in [Table 15-1](#) and cannot drive the FX2 connector but can receive signals.

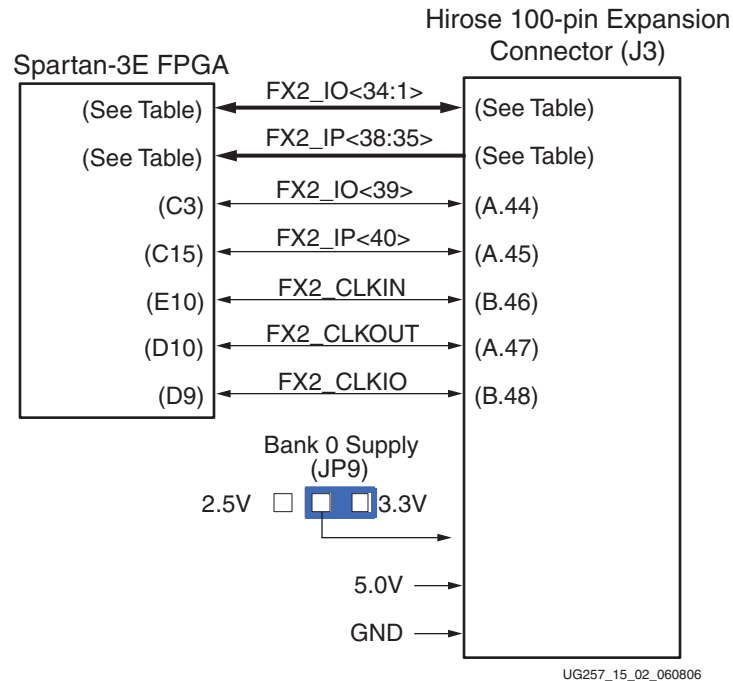


Figure 15-2: FPGA Connections to the Hirose 100-pin Edge Connector

Three signals are reserved primarily as clock signals between the board and FX2 connector, although all three connect to full I/O pins.

## Voltage Supplies to the Connector

The MicroBlaze Development Kit board provides power to the Hirose 100-pin FX connector and any attached board via two supplies (see Figure 15-2). The 5.0V supply provides a voltage source for any 5V logic on the attached board or alternately provides power to any voltage regulators on the attached board.

A separate supply provides the same voltage at that applied to the FPGA's I/O Bank 0. All FPGA I/Os that interface to the Hirose connector are in Bank 0. The I/O Bank 0 supply is 3.3V by default. However, the voltage level can be changed to 2.5V using jumper JP9. Some FPGA I/O standards—especially the differential standards such as RSDS and LVDS—require a 2.5V output supply voltage.

To support high-speed signals across the connector, a majority of pins on the B-side of the FX2 connector are tied to GND.

## Connector Pinout and FPGA Connections

Table 15-1 shows the pinout for the Hirose 100-pin FX2 connector and the associated FPGA pin connections. The FX2 connect has two rows of connectors, both with 50 connections each, shown in the table using light yellow shading.

Table 15-1 also highlights the shared connections to the eight discrete LEDs, the three 6-pin Accessory Headers (J1, J2, and J4), and the connectorless debugging header (J6).

Table 15-1: Hirose 100-pin FX2 Connector Pinout and FPGA Connections (J3)

Signal Name	FPGA Pin	Shared Header Connections		FX2 Connector		FPGA Pin	Signal Name
		LED	J6	A (top)	B (bottom)		
	VCCO_0			1	1		SHIELD
	VCCO_0			2	2	GND	GND
TMS_B				3	3		TDO_XC2C
JTSEL				4	4		TCK_B
TDO_FX2				5	5	GND	GND
FX2_IO1	B4		◆	6	6	GND	GND
FX2_IO2	A4		◆	7	7	GND	GND
FX2_IO3	D5		◆	8	8	GND	GND
FX2_IO4	C5		◆	9	9	GND	GND
FX2_IO5	A6		◆	10	10	GND	GND
FX2_IO6	B6		◆	11	11	GND	GND
FX2_IO7	E7		◆	12	12	GND	GND
FX2_IO8	F7		◆	13	13	GND	GND
FX2_IO9	D7		◆	14	14	GND	GND
FX2_IO10	C7		◆	15	15	GND	GND
FX2_IO11	F8		◆	16	16	GND	GND
FX2_IO12	E8		◆	17	17	GND	GND
FX2_IO13	F9		◆	18	18	GND	GND
FX2_IO14	E9		◆	19	19	GND	GND
FX2_IO15	D11		◆	20	20	GND	GND
FX2_IO16	C11		◆	21	21	GND	GND
FX2_IO17	F11		◆	22	22	GND	GND
FX2_IO18	E11		◆	23	23	GND	GND
FX2_IO19	E12			24	24	GND	GND
FX2_IO20	F12			25	25	GND	GND
FX2_IO21	A13			26	26	GND	GND
FX2_IO22	B13			27	27	GND	GND
FX2_IO23	A14			28	28	GND	GND
FX2_IO24	B14			29	29	GND	GND
FX2_IO25	C14			30	30	GND	GND
FX2_IO26	D14			31	31	GND	GND
FX2_IO27	A16			32	32	GND	GND

Table 15-1: Hirose 100-pin FX2 Connector Pinout and FPGA Connections (J3)

Signal Name	FPGA Pin	Shared Header Connections		FX2 Connector		FPGA Pin	Signal Name
		LED	J6	A (top)	B (bottom)		
FX2_IO28	B16			33	33	GND	GND
FX2_IO29	E13			34	34	GND	GND
FX2_IO30	C4			35	35	GND	GND
FX2_IO31	B11			36	36	GND	GND
FX2_IO32	A11			37	37	GND	GND
FX2_IO33	A8	LED7		38	38	GND	GND
FX2_IO34	G9	LED6		39	39	GND	GND
FX2_IP35	A7	LED5		40	40	GND	GND
FX2_IP36	D13	LED4		41	41	GND	GND
FX2_IP37	E6	LED3		42	42	GND	GND
FX2_IP38	D6	LED2		43	43	GND	GND
FX2_IO39	C3	LED1		44	44	GND	GND
FX2_IP40	C15			45	45	GND	GND
GND	GND			46	46	E10	FX2_CLKIN
FX2_CLKO UT	D10			47	47	GND	GND
GND	GND			48	48	D9	FX2_CLKIO
5.0V				49	49		5.0V
5.0V				50	50		SHIELD

## Compatible Board

The following board is compatible with the FX2 connector on the MicroBlaze Development Kit board:

- VDEC1 Video Decoder Board from Digilent, Inc.  
<http://www.digilentinc.com/Products/Detail.cfm?Prod=VDEC1>

## Mating Receptacle Connectors

The MicroBlaze Development Kit board uses a Hirose FX2-100P-1.27DS header connector. The header mates with any compatible 100-pin receptacle connector, including board-mounted and non-locking cable connectors.

## Differential I/O

The Hirose FX2 connector, header J3, supports up to 15 differential I/O pairs and two input-only pairs using either the LVDS or RSDS I/O standards, as listed in Table 15-2. All I/O pairs support differential input termination (DIFF\_TERM) as described in the

Spartan-3E data sheet. Select pairs have optional landing pads for external termination resistors.

These signals are not routed with matched differential impedance, as would be required for ultimate performance. However, all traces have similar lengths to minimize skew.

**Table 15-2: Differential I/O Pairs**

Differential Pair	Signal Name	FPGA Pins	FPGA Pin Name	Direction	DIFF_TERM	External Resistor Designator
1	FX2_IO1	B4	IO_L24N_0	I/O	Yes	
	FX2_IO2	A4	IO_L24P_0	I/O	Yes	
2	FX2_IO3	D5	IO_L23N_0	I/O	Yes	
	FX2_IO4	C5	IO_L23P_0	I/O	Yes	
3	FX2_IO5	A6	IO_L20N_0	I/O	Yes	
	FX2_IO6	B6	IO_L20P_0	I/O	Yes	
4	FX2_IO7	E7	IO_L19N_0	I/O	Yes	
	FX2_IO8	F7	IO_L19P_0	I/O	Yes	
5	FX2_IO9	D7	IO_L18N_0	I/O	Yes	
	FX2_IO10	C7	IO_L18P_0	I/O	Yes	
6	FX2_IO11	F8	IO_L17N_0	I/O	Yes	
	FX2_IO12	E8	IO_L17P_0	I/O	Yes	
7	FX2_IO13	F9	IP_L15N_0	I/O	Yes	
	FX2_IO14	E9	IP_L15P_0	I/O	Yes	
8	FX2_IO15	D11	IP_L09N_0	I/O	Yes	
	FX2_IO16	C11	IP_L09P_0	I/O	Yes	
9	FX2_IO17	F11	IO_L08N_0	I/O	Yes	R202
	FX2_IO18	E11	IO_L08P_0	I/O	Yes	
10	FX2_IO19	E12	IO_L06N_0	I/O	Yes	R203
	FX2_IO20	F12	IO_L06P_0	I/O	Yes	
11	FX2_IO21	A13	IO_L05P_0	I/O	Yes	R204
	FX2_IO22	B13	IO_L05N_0	I/O	Yes	
12	FX2_IO23	A14	IO_L04N_0	I/O	Yes	R205
	FX2_IO24	B14	IO_L04P_0	I/O	Yes	
13	FX2_IO25	C14	IO_L03N_0	I/O	Yes	R206
	FX2_IO26	D14	IO_L03P_0	I/O	Yes	
14	FX2_IO27	A16	IO_L01N_0	I/O	Yes	R207
	FX2_IO28	B16	IO_L01P_0	I/O	Yes	



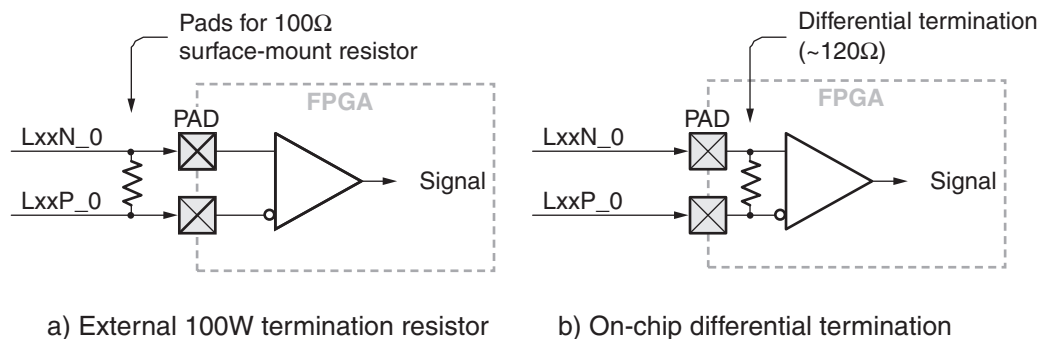
Table 15-2: Differential I/O Pairs (Continued)

Differential Pair	Signal Name	FPGA Pins	FPGA Pin Name	Direction	DIFF_TERM	External Resistor Designator
15	FX2_IP35	D12	IP_L07N_0	Input		R208
	FX2_IP36	C12	IP_L07P_0	Input		
16	FX2_IP37	A15	IP_L02N_0	Input		R209
	FX2_IP38	B15	IP_L02P_0	Input		
17	FX2_CLKIN	E10	IO_L11N_0/ GCLK5	I/O	Yes	R210
	FX2_CLKOUT	D10	IO_L11P_0/ GCLK4	I/O	Yes	

## Using Differential Inputs

LVDS and RSDS differential inputs require input termination. Two options are available. The first option is to use external termination resistors, as shown in Figure 15-3a. The board provides landing pads for external 100Ω termination resistors. The resistors are not loaded on the board as shipped. The resistor reference designators are labeled on the silkscreen, as listed in Table 15-2. The landing pads are located on both the top- and bottom-side of the board, between the FPGA and the FX2 connector. The resistors are not loaded on the board as shipped. External termination is always required when using differential input pairs 15 and 16.

The second option, shown in Figure 15-3b, is a Spartan-3E feature called on-chip differential termination, which uses the DIFF\_TERM attribute available on differential I/O signals. Each differential I/O pin includes a circuit that behaves like an internal termination resistor of approximately 120Ω. On-chip differential termination is only available on I/O pairs, not on Input-only pairs like pairs 15 and 16 in Table 15-2.



UG257\_15\_03\_060806

Figure 15-3: Differential Input Termination Options

Figure 15-4 and Figure 15-5 show the locations of the differential input termination resistor landing pads on the top and bottom side of the board. Table 15-2 indicates which resistor is associated with a specific differential pair.

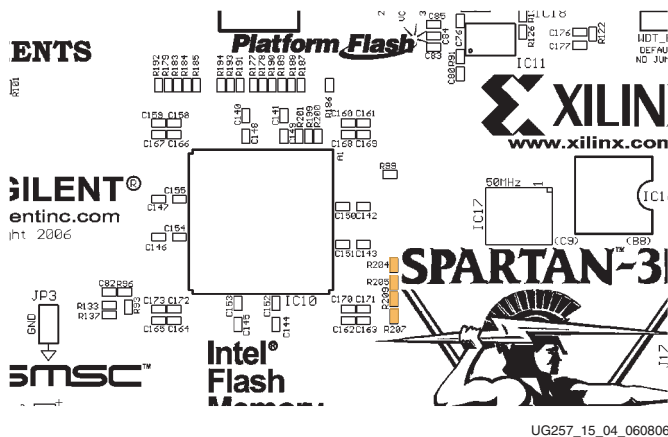


Figure 15-4: Location of Termination Resistor Pads on Top Side of Board

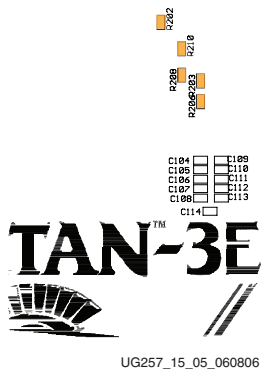


Figure 15-5: Location of Termination Resistor Pads on Bottom Side of Board

### Using Differential Outputs

Differential input signals do not require any special voltage. LVDS and RSDS differential outputs signals, on the other hand, require a 2.5V supply on I/O Bank 0. The board provides the option to power I/O Bank 0 with either 3.3V or 2.5V. [Figure 15-1, page 115](#) highlights the location of jumper JP9.

If using differential outputs on the FX2 connector, set jumper JP9 to 2.5V. If the jumper is not set correctly, the outputs switch correctly but the signal levels are out of specification.

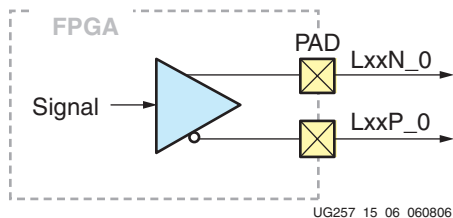


Figure 15-6: Differential Outputs

## UCF Location Constraints

Figure 15-7 provides the UCF constraints for the FX2 connector, including the I/O pin assignment and the I/O standard used, assuming that all connections use single-ended I/O standards. These header connections are shared with the 6-pin accessory headers, as shown in Figure 15-11, page 124.

```
# ===== FX2 Connector (FX2) =====
NET "FX2_CLKIN" LOC = "E10" | IOSTANDARD = LVCMOS33 ;
NET "FX2_CLKIO" LOC = "D9" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_CLKOUT" LOC = "D10" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_IO<1>" LOC = "B4" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_IO<2>" LOC = "A4" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_IO<3>" LOC = "D5" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_IO<4>" LOC = "C5" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_IO<5>" LOC = "A6" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_IO<6>" LOC = "B6" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_IO<7>" LOC = "E7" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_IO<8>" LOC = "F7" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_IO<9>" LOC = "D7" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_IO<10>" LOC = "C7" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_IO<11>" LOC = "F8" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_IO<12>" LOC = "E8" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_IO<13>" LOC = "F9" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_IO<14>" LOC = "E9" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_IO<15>" LOC = "D11" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_IO<16>" LOC = "C11" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_IO<17>" LOC = "F11" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_IO<18>" LOC = "E11" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_IO<19>" LOC = "E12" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_IO<20>" LOC = "F12" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_IO<21>" LOC = "A13" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_IO<22>" LOC = "B13" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_IO<23>" LOC = "A14" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_IO<24>" LOC = "B14" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_IO<25>" LOC = "C14" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_IO<26>" LOC = "D14" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_IO<27>" LOC = "A16" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_IO<28>" LOC = "B16" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_IO<29>" LOC = "E13" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_IO<30>" LOC = "C4" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_IO<31>" LOC = "B11" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_IO<32>" LOC = "A11" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
# The discrete LEDs are shared with the following 8 FX2 connections #
NET "FX2_IO<33>" LOC = "A8" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ; # LED7
# NET "FX2_IO<34>" LOC = "G9" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ; # LED6
# NET "FX2_IP<35>" LOC = "A7" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ; # LED5
# NET "FX2_IP<36>" LOC = "D13" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ; # LED4
# NET "FX2_IP<37>" LOC = "E6" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ; # LED3
# NET "FX2_IP<38>" LOC = "D6" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ; # LED2
# NET "FX2_IO<39>" LOC = "C3" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ; # LED1
#NET "FX2_IP<40>" LOC = "C15" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
```

UG257\_15\_07\_062106

Figure 15-7: UCF Location Constraints for Accessory Headers

## Six-Pin Accessory Headers

The 6-pin accessory headers provide easy I/O interface expansion using the various Digilent Peripheral Modules (see “[Related Resources](#),” page 126). The location of the 6-pin headers is provided in [Figure 15-1](#), page 115.

### Header J1

The J1 header, shown in [Figure 15-8](#), is the top-most 6-pin connector along the right edge of the board. It uses a female 6-pin 90° socket. Four FPGA pins connect to the J1 header, J1<4:1>. The board supplies 3.3V to the accessory board mounted in the J1 socket on the bottom pin.

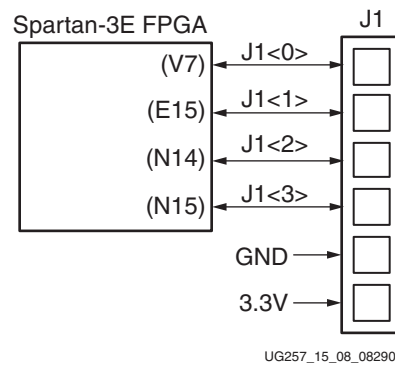


Figure 15-8: FPGA Connections to the J1 Accessory Header

### Header J2

The J2 header, shown in [Figure 15-9](#), is the bottom-most 6-pin connector along the right edge of the board. It uses a female 6-pin 90° socket. Four FPGA pins connect to the J2 header, J2<4:1>. The board supplies 3.3V to the accessory board mounted in the J4 socket on the bottom pin.

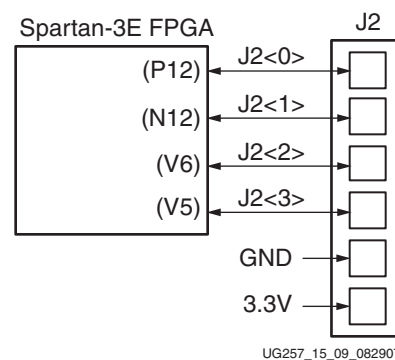


Figure 15-9: FPGA Connections to the J2 Accessory Header

## Header J4

The J4 header, shown in [Figure 15-10](#), is located immediately to the left of the J1 header. It uses a 6-pin header consisting of 0.1-inch centered stake pins. Four FPGA pins connect to the J4 header, J4<4:1>. Four FPGA pins connect to the J4<4:1> header. The board supplies 3.3V to the accessory board mounted in the J4 socket on the bottom pin.

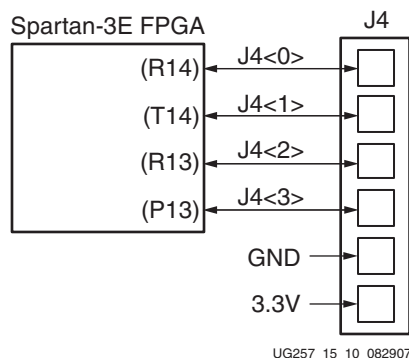


Figure 15-10: FPGA Connections to the J4 Accessory Header

## UCF Location Constraints

[Figure 15-11](#) provides the User Constraint File (UCF) constraints for accessory headers, including the I/O pin assignment and the I/O standard used. These header connections are shared with the FX2 connector, as shown in [Figure 15-7](#), page 122.

```
# ==== 6-pin header J1 ====
# These four connections are shared with the FX2 connector
#NET "J1<0>" LOC = "B4" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 6 ;
#NET "J1<1>" LOC = "A4" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 6 ;
#NET "J1<2>" LOC = "D5" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 6 ;
#NET "J1<3>" LOC = "C5" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 6 ;

# ==== 6-pin header J2 ====
# These four connections are shared with the FX2 connector
#NET "J2<0>" LOC = "A6" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 6 ;
#NET "J2<1>" LOC = "B6" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 6 ;
#NET "J2<2>" LOC = "E7" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 6 ;
#NET "J2<3>" LOC = "F7" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 6 ;

# ==== 6-pin header J4 ====
# These four connections are shared with the FX2 connector
#NET "J4<0>" LOC = "D7" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 6 ;
#NET "J4<1>" LOC = "C7" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 6 ;
#NET "J4<2>" LOC = "F8" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 6 ;
#NET "J4<3>" LOC = "E8" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 6 ;
```

UG257\_15\_11\_062106

Figure 15-11: UCF Location Constraints for Accessory Headers

## Connectorless Debugging Port Landing Pads (J6)

Landing pads for a connectorless debugging port are provided as header J6, shown in [Figure 15-1, page 115](#). There is no physical connector on the board. Instead a connectorless probe, such as those available from Agilent, provides an interface to a logic analyzer. This debugging port is intended primarily for the Xilinx ChipScope Pro software with the Agilent's FPGA Dynamic Probe. It can, however, be used with either the Agilent or Tektronix probes, without the ChipScope software, using FPGA Editor's probe command. Refer to "[Related Resources,](#)" [page 126](#) for more information on the ChipScope Pro tool, probes, and connectors.

[Table 15-3](#) provides the connector pinout. Only 18 FPGA pins attach to the connector; the remaining connector pads are unconnected. All 18 FPGA pins are shared with the FX2 connector (J3) and the 6-pin accessory port connectors (J1, J2, and J4). See [Table 15-1, page 117](#) for more information on how these pins are shared.

**Table 15-3: Connectorless Debugging Port Landing Pads (J6)**

Signal Name	FPGA Pin	Connectorless Landing Pads		FPGA Pin	Signal Name
FX2_IO1	B4	A1	B1	GND	GND
FX2_IO2	A4	A2	B2	D5	FX2_IO3
GND	GND	A3	B3	C5	FX2_IO4
FX2_IO5	A6	A4	B4	GND	GND
FX2_IO6	B6	A5	B5	E7	FX2_IO7
GND	GND	A6	B6	F7	FX2_IO8
FX2_IO9	D7	A7	B7	GND	GND
FX2_IO10	C7	A8	B8	F8	FX2_IO11
GND	GND	A9	B9	E8	FX2_IO12
FX2_IO13	F9	A10	B10	GND	GND
FX2_IO14	E9	A11	B11	D11	FX2_IO15
GND	GND	A12	B12	C11	FX2_IO16
FX2_IO17	F11	A13	B13	GND	GND
FX2_IO18	E11	A14	B14		
		A15	B15		
		A16	B16		
		A17	B17		
		A18	B18		
		A19	B19		
		A20	B20		
		A21	B21		
		A22	B22		
		A23	B23		
		A24	B24		
		A25	B25		
		A26	B26		
		A27	B27		

## Related Resources

- Hirose connectors  
<http://www.hirose-connectors.com/>
- FX2 Series Connector Data Sheet  
[http://www.hirose.co.jp/cataloge\\_hp/e57220088.pdf](http://www.hirose.co.jp/cataloge_hp/e57220088.pdf)
- Digilent, Inc. Peripheral Modules  
<http://www.digilentinc.com/Products/Catalog.cfm?Nav1=Products&Nav2=Peripheral&Cat=Peripheral>
- Xilinx ChipScope Pro Tool  
[http://www.xilinx.com/ise/optional\\_prod/cspro.htm](http://www.xilinx.com/ise/optional_prod/cspro.htm)
- Agilent B4655A FPGA Dynamic Probe for Logic Analyzer  
<http://www.home.agilent.com/USeng/nav/-536898189.536883660/pd.html?cmpid=92641>
- Agilent 5404A/6A Pro Series Soft Touch Connector  
[http://www.home.agilent.com/cgi-bin/pub/agilent/Product/cp\\_Product.jsp?NAV\\_ID=-536898227.0.00](http://www.home.agilent.com/cgi-bin/pub/agilent/Product/cp_Product.jsp?NAV_ID=-536898227.0.00)
- Tektronix P69xx Probe Modules with D-Max Technology  
[http://www.tek.com/products/accessories/logic\\_analyzers/p6800\\_p6900.html](http://www.tek.com/products/accessories/logic_analyzers/p6800_p6900.html)

## XC2C64A CoolRunner-II CPLD

---

The MicroBlaze Development Kit board includes a Xilinx XC2C64A CoolRunner-II CPLD. The CPLD is user programmable and available for customer applications. Portions of the CPLD are reserved to coordinate behavior between the various FPGA configuration memories, namely the Xilinx Platform Flash PROM and the Intel StrataFlash PROM. Consequently, the CPLD must provide the following functions in addition to the user application.

- When the FPGA is in the Master Serial configuration mode (FPGA\_M<2:0>=000), generate an active-Low enable signal for the XCF04S Platform Flash PROM. The Platform Flash PROM is disabled in all other configuration modes. The CPLD helps reduce the number of jumpers on the board and simplifies the interaction of all the possible FPGA configuration memory sources.
- When the FPGA is actively in the BPI-Up configuration mode (FPGA\_M<2:0>=010, DONE=0), set the upper five StrataFlash PROM address lines, A[24:20], to 00000 binary. When the FPGA is actively in the BPI-Down configuration mode (FPGA\_M<2:0>=011, DONE=0), set the upper five StrataFlash PROM address lines, A[24:20], to 11111 binary. Set the upper five address lines to ZZZZZ for all non-BPI configuration modes or whenever the FPGA's DONE pin is High. This behavior is identical to the way the FPGA's upper address lines function during BPI mode. So why add a CPLD to mimic this behavior? A future reference design demonstrates unique configuration capabilities. In a typical BPI-mode application, the CPLD is not required.

Other than the required CPLD functionality, there are between 13 to 21 user-I/O pins and 58 remaining macrocells available to the user application.

Jumper JP10 (WDT\_EN) defines the state on the CPLD's XC\_WDT\_EN signal. By default, this jumper is empty and the signal is pulled to a logic High.

The XC\_PROG\_B output from the CPLD, if used, must be configured as an open-drain out (*i.e.*, either actively drives Low or floats to Hi-Z, never drives High). This signal connects directly to the FPGA's PROG\_B programming pin.

The most-significant StrataFlash PROM address bit, SF\_A<24>, is the same as the FX2 connector signal called FX2\_IO<32>. The 16 Mbyte StrataFlash PROM only physically uses the lower 24 bits, SF\_A<23:0>. The extra address bit, SF\_A<24>, is provided for upward density migration for the StrataFlash PROM.



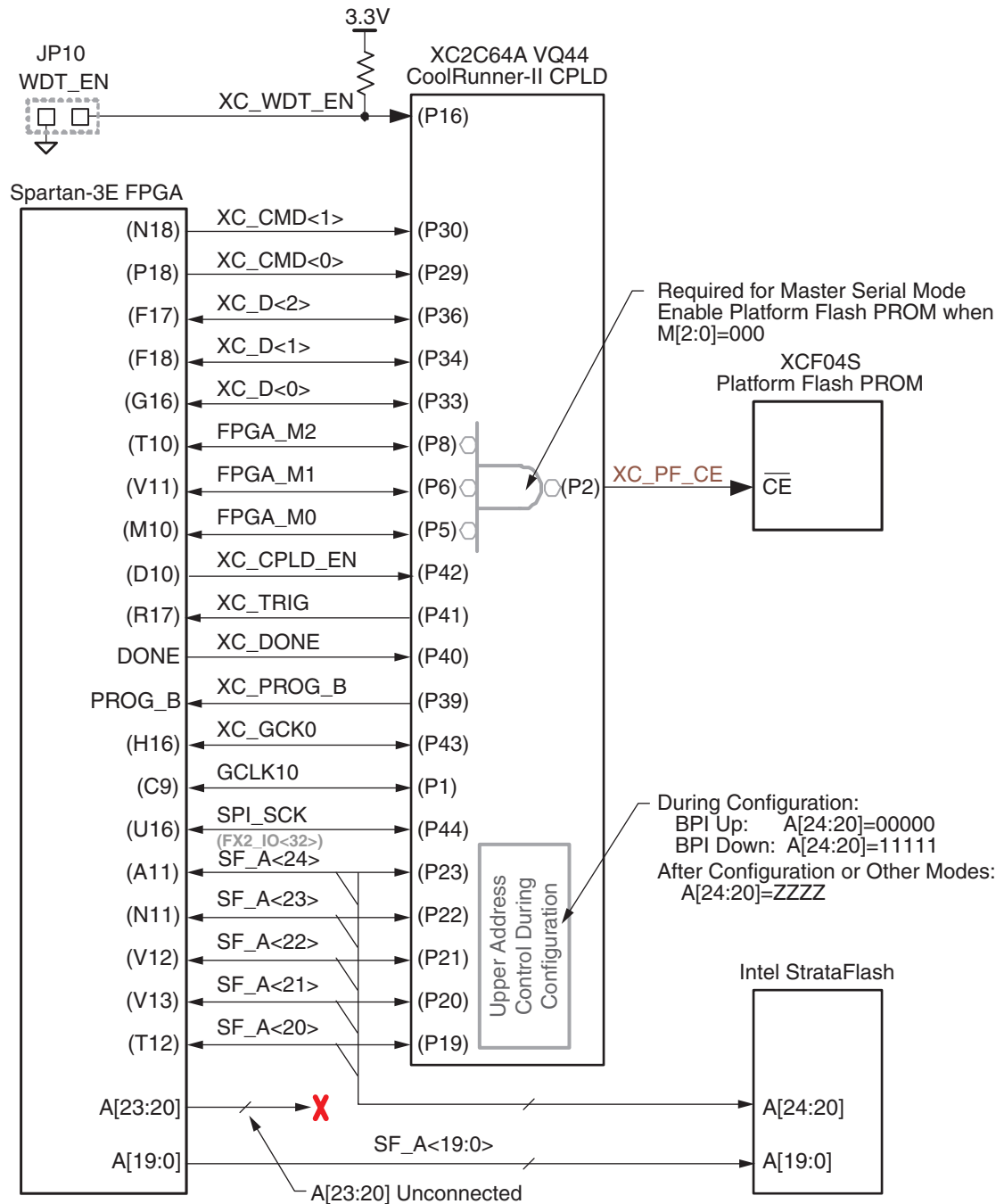


Figure 16-1: XC2C64A CoolRunner-II CPLD Controls Master Serial and BPI Configuration Modes

## UCF Location Constraints

There are two sets of constraints listed below—one for the Spartan-3E FPGA and one for the XC2C64A CoolRunner-II CPLD.

## FPGA Connections to CPLD

Figure 16-2 provides the UCF constraints for the FPGA connections to the CPLD , including the I/O pin assignment and the I/O standard used.

NET "XC_CMD<1>"	LOC = "N18"	IOSTANDARD = LVCMOS33	DRIVE = 4	SLEW = SLOW ;
NET "XC_CMD<0>"	LOC = "P18"	IOSTANDARD = LVCMOS33	DRIVE = 4	SLEW = SLOW ;
NET "XC_D<2>"	LOC = "F17"	IOSTANDARD = LVCMOS33	DRIVE = 4	SLEW = SLOW ;
NET "XC_D<1>"	LOC = "F18"	IOSTANDARD = LVCMOS33	DRIVE = 4	SLEW = SLOW ;
NET "XC_D<0>"	LOC = "G16"	IOSTANDARD = LVCMOS33	DRIVE = 4	SLEW = SLOW ;
NET "FPGA_M2"	LOC = "T10"	IOSTANDARD = LVCMOS33	DRIVE = 4	SLEW = SLOW ;
NET "FPGA_M1"	LOC = "V11"	IOSTANDARD = LVCMOS33	DRIVE = 4	SLEW = SLOW ;
NET "FPGA_M0"	LOC = "M10"	IOSTANDARD = LVCMOS33	DRIVE = 4	SLEW = SLOW ;
NET "XC_CPLD_EN"	LOC = "B10"	IOSTANDARD = LVCMOS33	DRIVE = 4	SLEW = SLOW ;
NET "XC_TRIG"	LOC = "R17"	IOSTANDARD = LVCMOS33 ;		
NET "XC_GCK0"	LOC = "H16"	IOSTANDARD = LVCMOS33	DRIVE = 4	SLEW = SLOW ;
NET "GCLK10"	LOC = "C9"	IOSTANDARD = LVCMOS33	DRIVE = 4	SLEW = SLOW ;
NET "SPI_SCK"	LOC = "U16"	IOSTANDARD = LVCMOS33	DRIVE = 4	SLEW = SLOW ;
# SF_A<24> is the same as FX2_IO<32>				
NET "SF_A<24>"	LOC = "A11"	IOSTANDARD = LVCMOS33	DRIVE = 4	SLEW = SLOW ;
NET "SF_A<23>"	LOC = "N11"	IOSTANDARD = LVCMOS33	DRIVE = 4	SLEW = SLOW ;
NET "SF_A<22>"	LOC = "V12"	IOSTANDARD = LVCMOS33	DRIVE = 4	SLEW = SLOW ;
NET "SF_A<21>"	LOC = "V13"	IOSTANDARD = LVCMOS33	DRIVE = 4	SLEW = SLOW ;
NET "SF_A<20>"	LOC = "T12"	IOSTANDARD = LVCMOS33	DRIVE = 4	SLEW = SLOW ;

UG257\_16\_02\_060806

Figure 16-2: UCF Location Constraints for FPGA Connections to CPLD

## CPLD

Figure 16-3 provides the UCF constraints for the CPLD , including the I/O pin assignment and the I/O standard used

NET "XC_WDT_EN"	LOC = "P16"	IOSTANDARD = LVCMOS33 ;		
NET "XC_CMD<1>"	LOC = "P30"	IOSTANDARD = LVCMOS33	SLEW = SLOW ;	
NET "XC_CMD<0>"	LOC = "P29"	IOSTANDARD = LVCMOS33	SLEW = SLOW ;	
NET "XC_D<2>"	LOC = "P36"	IOSTANDARD = LVCMOS33	SLEW = SLOW ;	
NET "XC_D<1>"	LOC = "P34"	IOSTANDARD = LVCMOS33	SLEW = SLOW ;	
NET "XC_D<0>"	LOC = "P33"	IOSTANDARD = LVCMOS33	SLEW = SLOW ;	
NET "FPGA_M2"	LOC = "P8"	IOSTANDARD = LVCMOS33	SLEW = SLOW ;	
NET "FPGA_M1"	LOC = "P6"	IOSTANDARD = LVCMOS33	SLEW = SLOW ;	
NET "FPGA_M0"	LOC = "P5"	IOSTANDARD = LVCMOS33	SLEW = SLOW ;	
NET "XC_CPLD_EN"	LOC = "P42"	IOSTANDARD = LVCMOS33	SLEW = SLOW ;	
NET "XC_TRIG"	LOC = "P41"	IOSTANDARD = LVCMOS33	SLEW = SLOW ;	
NET "XC_DONE"	LOC = "P40"	IOSTANDARD = LVCMOS33	SLEW = SLOW ;	
NET "XC_PROG_B"	LOC = "P39"	IOSTANDARD = LVCMOS33	SLEW = SLOW ;	
NET "XC_GCK0"	LOC = "P43"	IOSTANDARD = LVCMOS33	SLEW = SLOW ;	
NET "GCLK10"	LOC = "P1"	IOSTANDARD = LVCMOS33	SLEW = SLOW ;	
NET "SPI_SCK"	LOC = "P44"	IOSTANDARD = LVCMOS33	SLEW = SLOW ;	
# SF_A<24> is the same as FX2_IO<32>				
NET "SF_A<24>"	LOC = "P23"	IOSTANDARD = LVCMOS33	SLEW = SLOW ;	
NET "SF_A<23>"	LOC = "P22"	IOSTANDARD = LVCMOS33	SLEW = SLOW ;	
NET "SF_A<22>"	LOC = "P21"	IOSTANDARD = LVCMOS33	SLEW = SLOW ;	
NET "SF_A<21>"	LOC = "P20"	IOSTANDARD = LVCMOS33	SLEW = SLOW ;	
NET "SF_A<20>"	LOC = "P19"	IOSTANDARD = LVCMOS33	SLEW = SLOW ;	

UG257\_16\_03\_060806

Figure 16-3: UCF Location Constraints for the XC2C64A CPLD

## Related Resources

- CoolRunner-II CPLD Family Data Sheet  
<http://direct.xilinx.com/bvdocs/publications/ds090.pdf>
- XC2C64A CoolRunner-II CPLD Data Sheet  
<http://direct.xilinx.com/bvdocs/publications/ds311.pdf>
- Default XC2C64A CPLD Design for Spartan-3E Starter Kit Board  
<http://www.xilinx.com/s3estarter>

## DS2432 1-Wire SHA-1 EEPROM

The MicroBlaze Development Kit board includes a Maxim DS2432 serial EEPROM with an integrated SHA-1 engine. As shown in [Figure 17-1](#), the DS2432 EEPROM uses the Maxim 1-Wire interface, which uses a single wire for power and serial communication.

The DS2432 EEPROM offers one of many possible means to copy and protect the FPGA configuration bitstream, thereby making cloning difficult. Xilinx application note XAPP780, listed under [“Related Resources”](#) provides one possible implementation method.

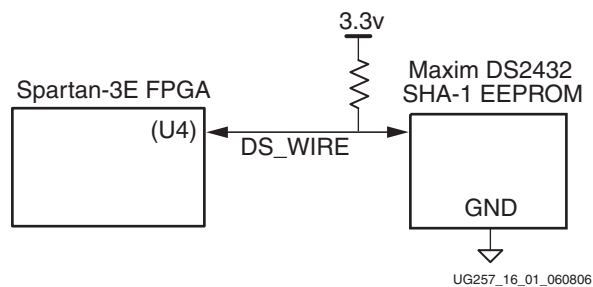


Figure 17-1: SHA-1 EEPROM

### UCF Location Constraints

[Figure 17-2](#) provides the UCF constraints for the FPGA connections to the DS2432 SHA-1 EEPROM, including the I/O pin assignment and the I/O standard used.

```
NET "DS_WIRE" LOC = "U4" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;
```

UG257\_17\_02\_061606

Figure 17-2: UCF Location Constraints for DS2432 SHA-1 EEPROM

### Related Resources

- Maxim DS2432 1-Wire EEPROM with SHA-1 Engine  
[http://www.maxim-ic.com/quick\\_view2.cfm/qv\\_pk/2914](http://www.maxim-ic.com/quick_view2.cfm/qv_pk/2914)
- XAPP780: FPGA IFF Copy Protection Using Dallas Semiconductor/Maxim DS2432 Secure EEPROMs  
<http://www.xilinx.com/bvdocs/appnotes/xapp780.pdf>



## Schematics

---

This appendix provides the following circuit board schematics:

- “FX2 Expansion Header, 6-pin Headers, and Connectorless Probe Header”
- “RS-232 Ports, VGA Port, and PS/2 Port”
- “Ethernet PHY, Magnetics, and RJ-11 Connector”
- “Voltage Regulators”
- “FPGA Configurations Settings, Platform Flash PROM, SPI Serial Flash, JTAG Connections”
- “FPGA I/O Banks 0 and 1, Oscillators”
- “FPGA I/O Banks 2 and 3”
- “Power Supply Decoupling”
- “XC2C64A CoolRunner-II CPLD”
- “Linear Technology ADC and DAC ”
- “Intel StrataFlash Parallel NOR Flash Memory and Micron DDR SDRAM ”
- “Buttons, Switches, Rotary Encoder, and Character LCD ”
- “DDR SDRAM Series Termination and FX2 Connector Differential Termination”

## FX2 Expansion Header, 6-pin Headers, and Connectorless Probe Header

Headers J1, J2, and J4 are six-pin connectors compatible with the Digilent Accessory board format.

Headers J3A and J3B are the connections to the FX2 expansion connector located along the right edge of the board.

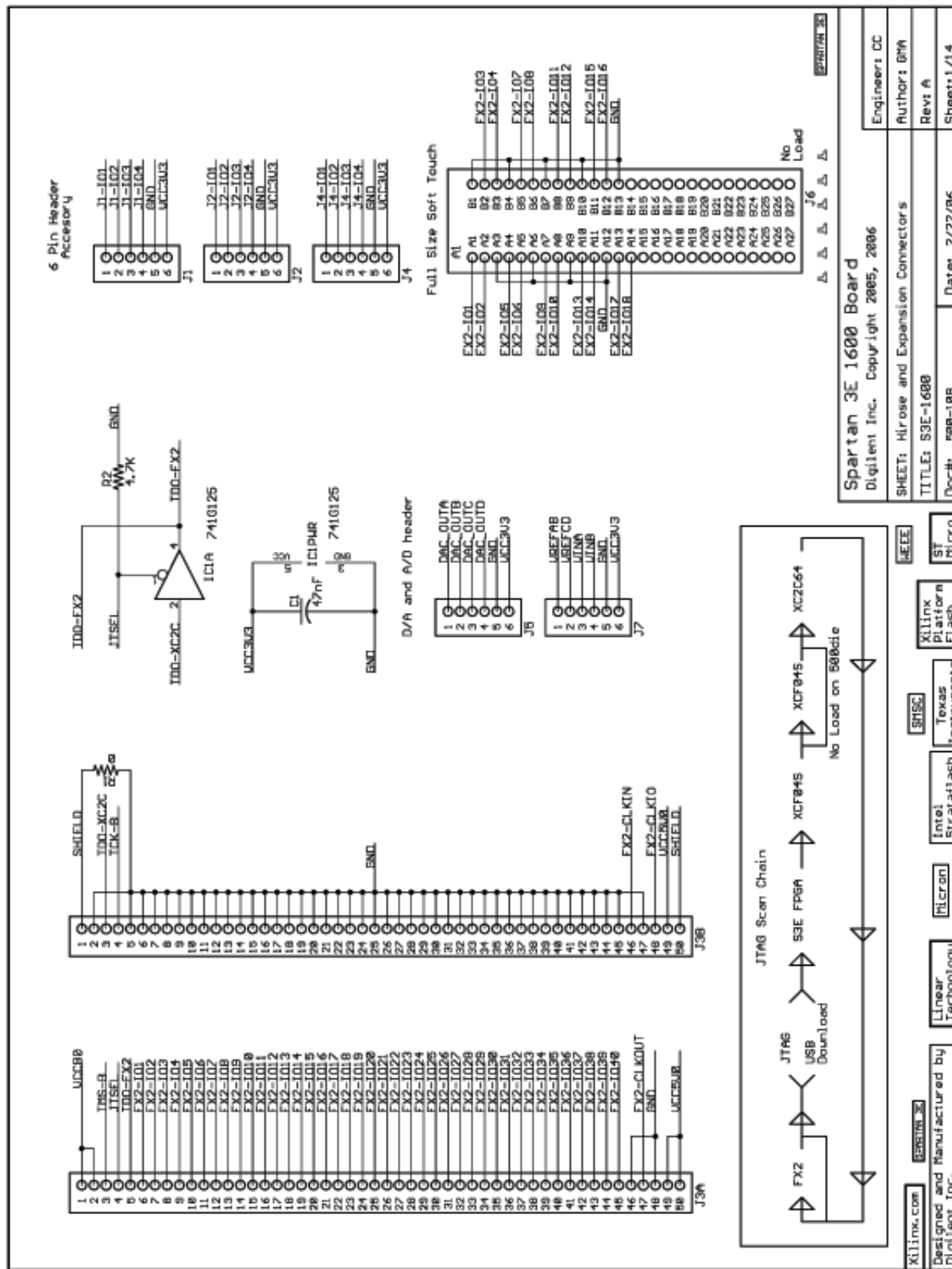
Header J5 provides the four analog outputs from the Digital-to-Analog Converter (DAC).

Header J6 is the landing pad for an Agilent or Tektronix connectorless probe.

Header J7 provides the two analog inputs to the programmable pre-amplifier (AMP) and two-channel Analog-to-Digital Converter (ADC).

The diagram in the lower left corner shows the JTAG chain.

See [Chapter 15, "Expansion Connectors,"](#) for additional information.



**Spartan 3E 1600 Board**  
 Digilent Inc. Copyright 2005, 2006

SHEET: Hirose and Expansion Connectors  
 TITLE: S3E-1600  
 Doc#: 500-108

Engineers CC  
 Author: DMH  
 Rev: A  
 Date: 2/22/06  
 Sheet: 1/14

Figure 18-1: Schematic Sheet 1



## RS-232 Ports, VGA Port, and PS/2 Port

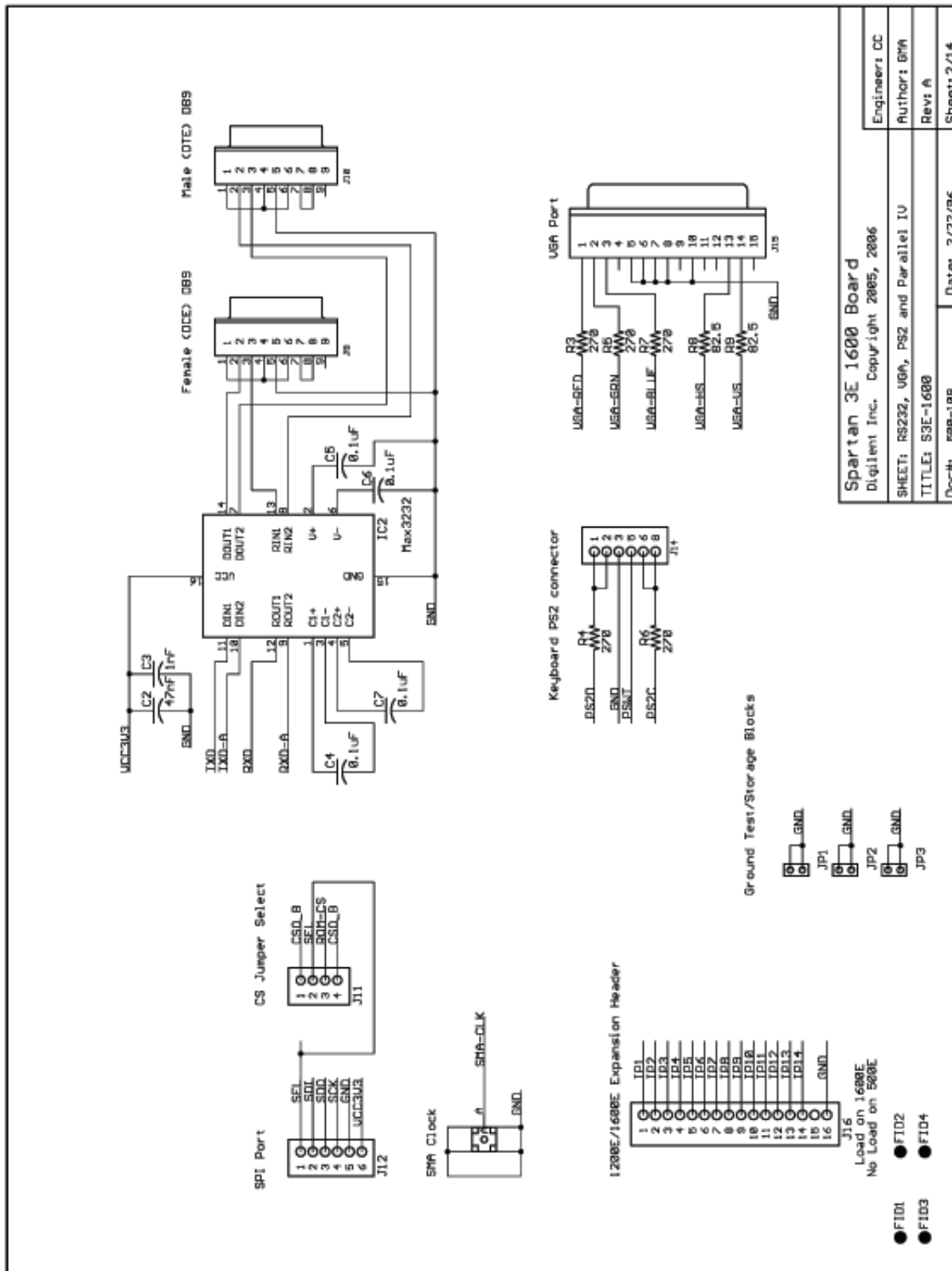
IC2 is the Maxim LVTTTL to RS-232 level converter. One of the serial channels connects to a female DB9 DCE connector (J9) and the other connects to a male DB9 DTE connector (J10). See [Chapter 7, “RS-232 Serial Ports,”](#) for additional information.

Connector J14 is a PS/2-style mouse/keyboard connector, powered from 5 volts. See [Chapter 8, “PS/2 Mouse/Keyboard Port,”](#) for additional information.

Connector J15 is a VGA connector, suitable for driving most VGA-compatible monitors and flat-screen displays. See [Chapter 6, “VGA Display Port,”](#) for additional information.

Header J12 provides programming support for the SPI serial Flash. Jumper J11 controls how the SPI serial Flash is enabled in the application. See [Chapter 12, “SPI Serial Flash,”](#) for additional information.

The SMA connector allows an external clock source to drive one of the FPGA’s global clock inputs. Alternatively, the FPGA can provide a high-performance clock to another board via the SMA connector. See [Chapter 3, “Clock Sources,”](#) for additional information.



UG257\_A02\_060606

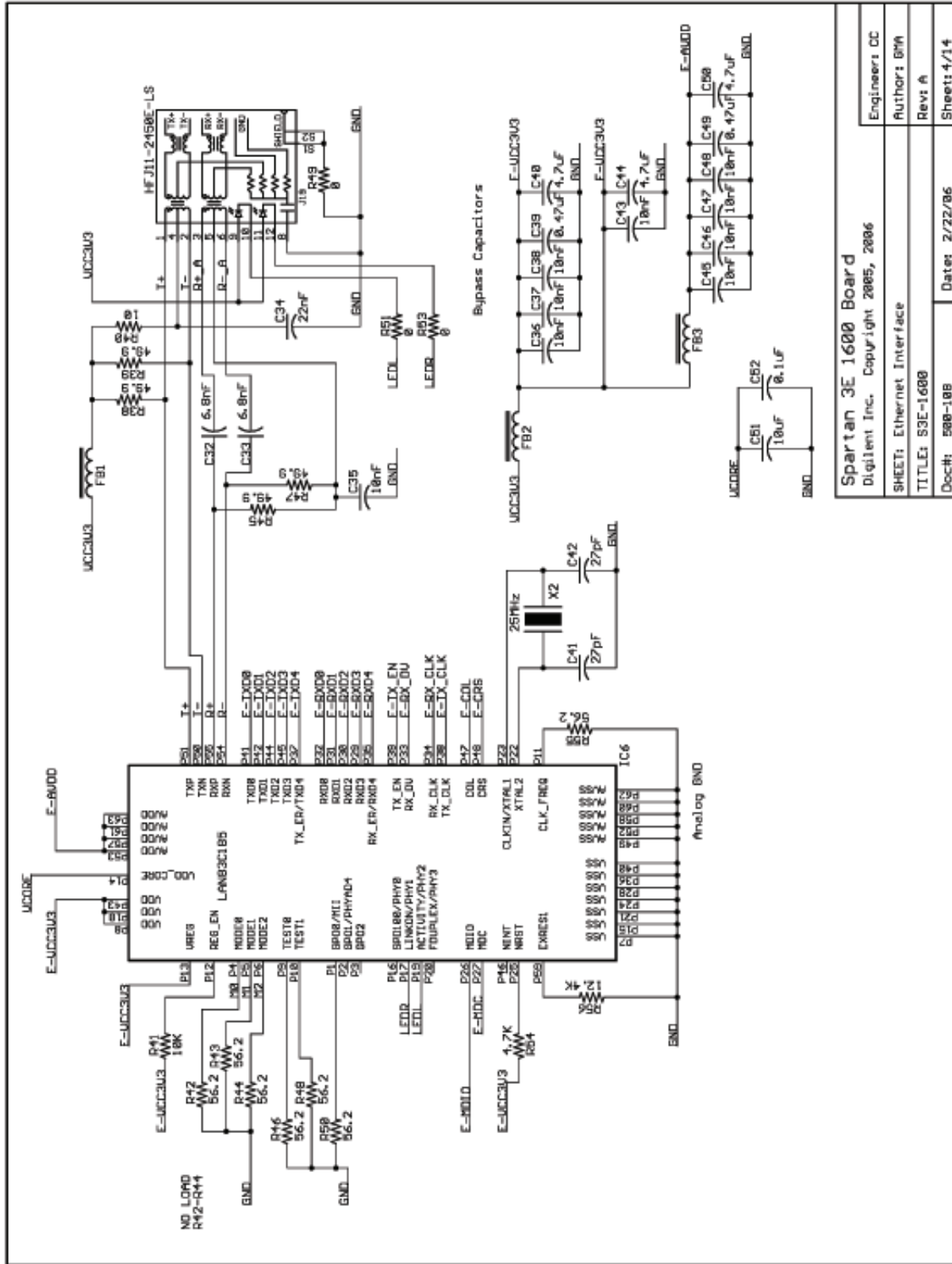
Figure 18-2: Schematic Sheet 2

## Ethernet PHY, Magnetics, and RJ-11 Connector

IC6 is an SMSC 10/100 Ethernet PHY, with its associated 25 MHz oscillator. The PHY requires an Ethernet MAC implemented within the FPGA.

J19 is the RJ-11 Ethernet connector associated with the 10/100 Ethernet PHY.

See [Chapter 14, "10/100 Ethernet Physical Layer Interface,"](#) for additional information.



Spartan 3E 1600 Board		Engineers CC
Diligent Inc. Copyright 2005, 2006		Author: B/M
SHEET: Ethernet Interface		Rev: A
TITLE: S3E-1600		Doc#: 508-108
Date: 2/22/06		Sheets: 4/14

Figure 18-3: Schematic Sheet 4

UG257\_A03\_060606

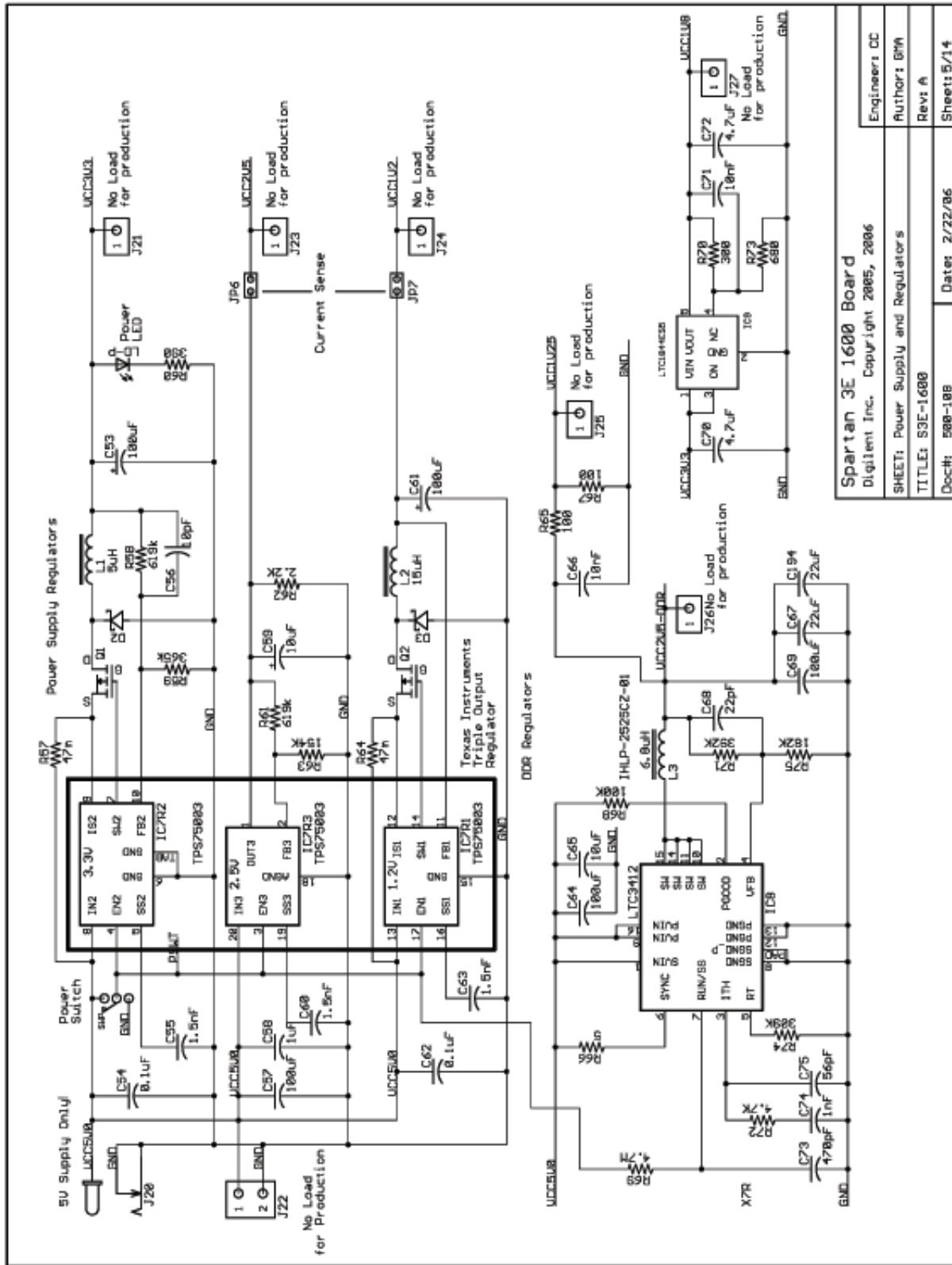
## Voltage Regulators

IC7 is a [Texas Instruments TPS75003](#) triple-output regulator. The regulator provides 1.2V to the FPGA's VCCINT supply input, 2.5V to the FPGA's VCCAUX supply input, and 3.3V to other components on the board and to the FPGA's VCCO supply inputs on I/O Banks 0, 1, and 2.

Jumpers JP6 and JP7 provide a means to measure current across the FPGA's VCCAUX and VCCINT supplies respectively.

IC8 is a Linear Technology LT3412 regulator, providing 2.5V to the on-board DDR SDRAM. Resistors R65 and R67 create a voltage divider to create the termination voltage required for the DDR SDRAM interface.

IC9 is a 1.8V supply to the Embedded USB download/debug circuit and to the CPLD's VCCINT supply input.



UG257\_A04\_060606

Figure 18-4: Schematic Sheet 5

Spartan 3E 1600 Board		Engineers CC
Diligent Inc. Copyright 2005, 2006		Author: BM
SHEET: Power Supply and Regulators		Rev: A
TITLE: S3E-1600		Date: 2/22/05
Doc#: 500-108		Sheet: 5/14

## FPGA Configurations Settings, Platform Flash PROM, SPI Serial Flash, JTAG Connections

IC10MISC represents the various FPGA configuration connections.

IC11 is a 4 Mbit XCF04S Platform Flash PROM. Landing pads for a second XCF04S PROM is shown as IC13, although the second PROM is not mounted on the XC3S500E version of the board. Resistor R100 jumpers over the JTAG chain, bypassing the second XCF04S PROM.

Jumper header J30 selects the FPGA's configuration mode. See [Table 4-1, page 25](#) for additional information.

Header J28 is an alternate JTAG header.

IC12 is a Maxim/Dallas Semiconductor DS2432 SHA-1 EEPROM. See [Chapter 17, "DS2432 1-Wire SHA-1 EEPROM,"](#) for more information.

IC14 and IC15 are alternate landing pads for the STMicro SPI serial Flash. IC14 accepts the 16-pin SOIC package option, while IC15 accepts either the 8-pin SOIC or MLP package option. See [Figure 12-19, page 103](#) for additional information.

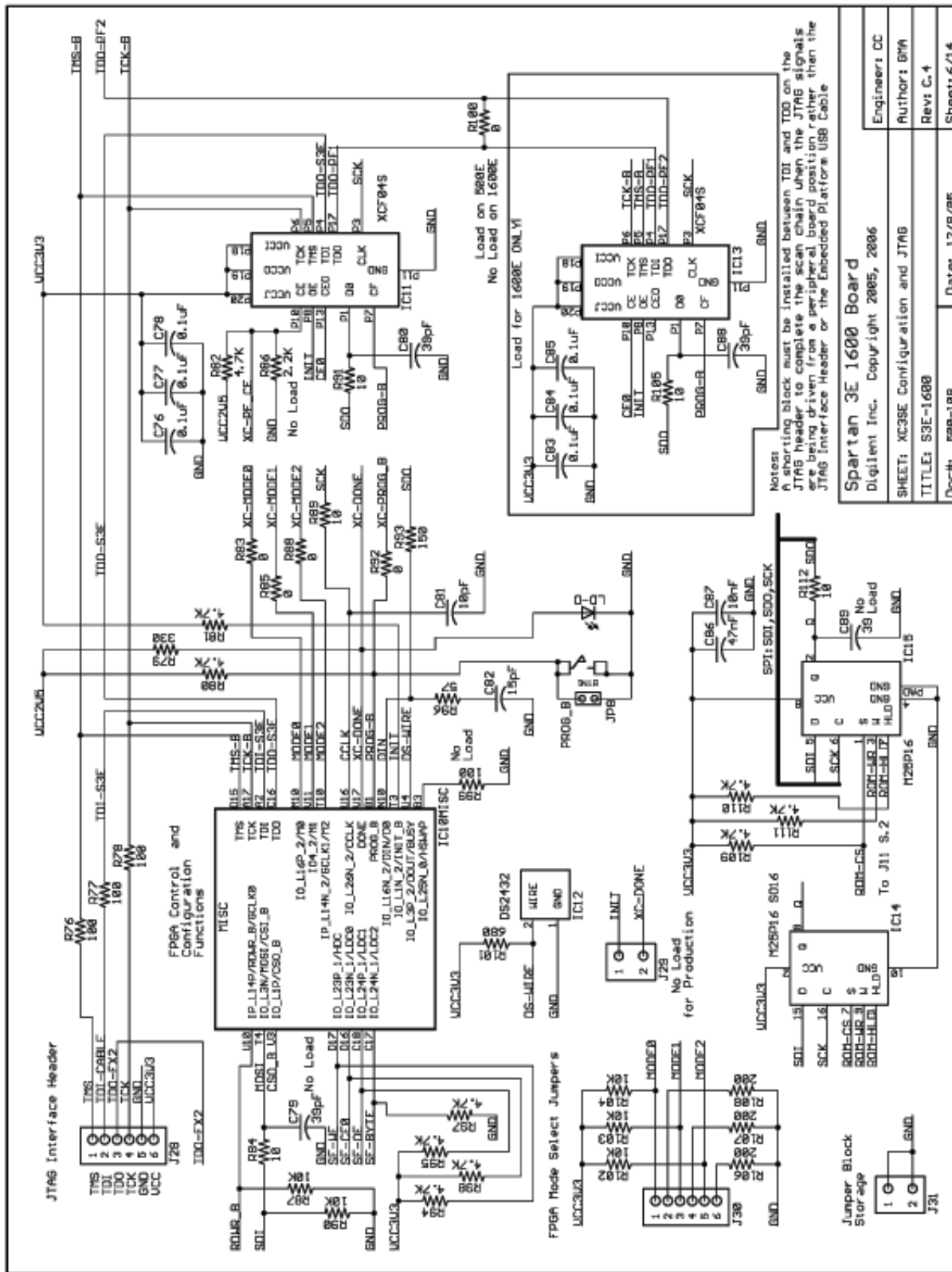


Figure 18-5: Schematic Sheet 6



## FPGA I/O Banks 0 and 1, Oscillators

IC10B0 represents the connections to I/O Bank 0 on the FPGA. The VCCO input to Bank 0 is 3.3V by default, but can be set to 2.5V using jumper JP9.

IC10B1 represents the connections to I/O Bank 1 on the FPGA.

IC17 is the 50 MHz clock oscillator. [Chapter 3, "Clock Sources,"](#) for additional information.

IC16 is an 8-pin DIP socket to insert an alternate clock oscillator with a different frequency.

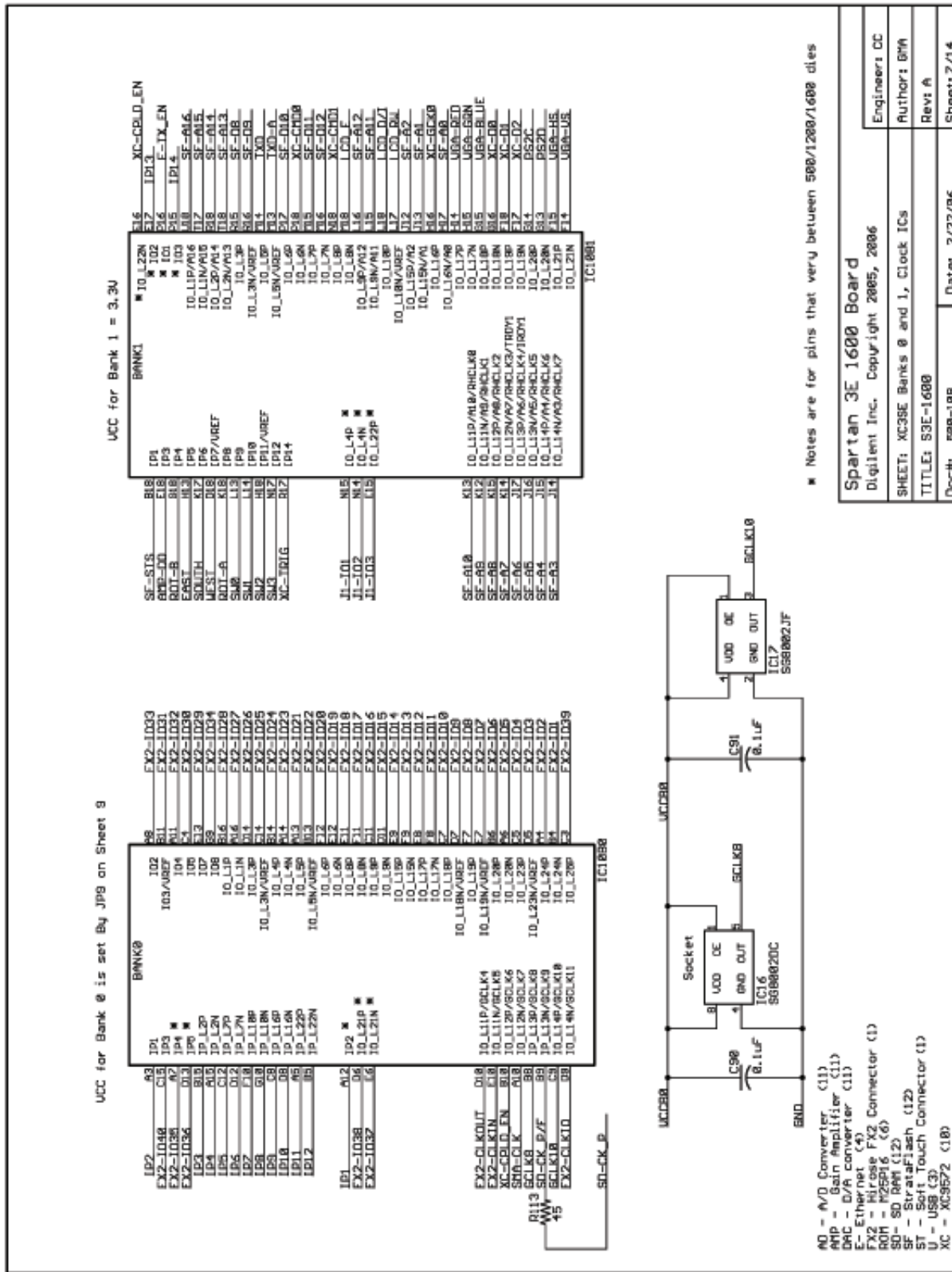


Figure 18-6: Schematic Sheet 7

## FPGA I/O Banks 2 and 3

IC10B2 represents the connections to I/O Bank 2 on the FPGA. Some of the I/O Bank 2 connections are used for FPGA configuration and are listed as IC10MISC.

IC10B3 represents the connections to I/O Bank 3 on the FPGA. Bank 3 is dedicated to the DDR SDRAM interface and is consequently powered by 2.5V. See [Chapter 13, “DDR SDRAM,”](#) for additional information.

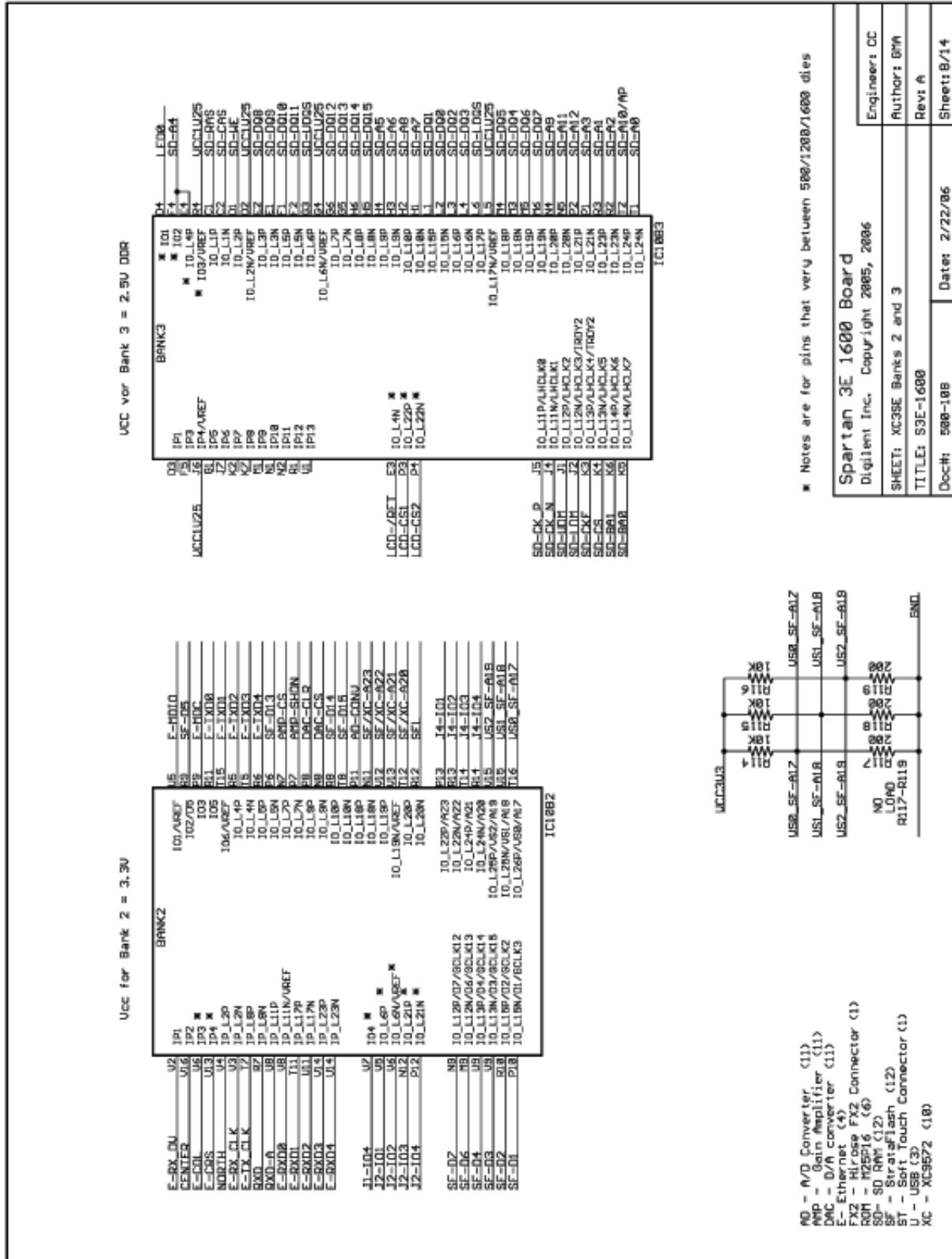


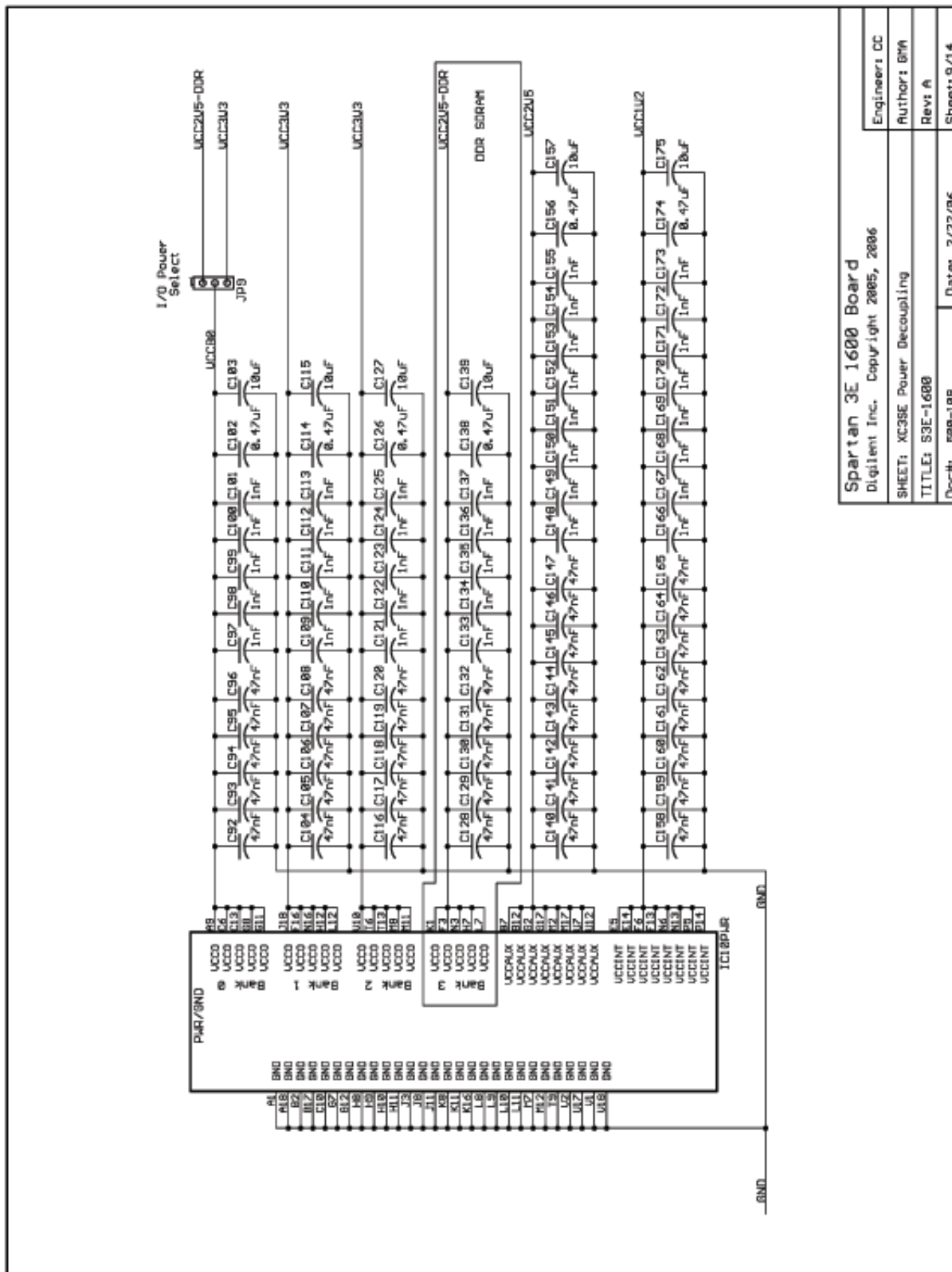
Figure 18-7: Schematic Sheet 8

UG257\_A07\_060606

## Power Supply Decoupling

IC10PWR represents the various voltage supply inputs to the FPGA and shows the power decoupling network.

Jumper JP9 defines the voltage applied to VCCO on I/O Bank 0. The default setting is 3.3V. See [“Voltage Control,”](#) page 20 and [“Voltage Supplies to the Connector,”](#) page 116 for additional details.



Spartan 3E 1600 Board	
Digilent Inc. Copyright 2005, 2006	
Engineers CC	Author: BVA
SHEET: XC3SE Power Decoupling	Rev: A
TITLE: 3E-1600	Date: 2/22/06
Doc#: 598-108	Sheets: 9/14

Figure 18-8: Schematic Sheet 9

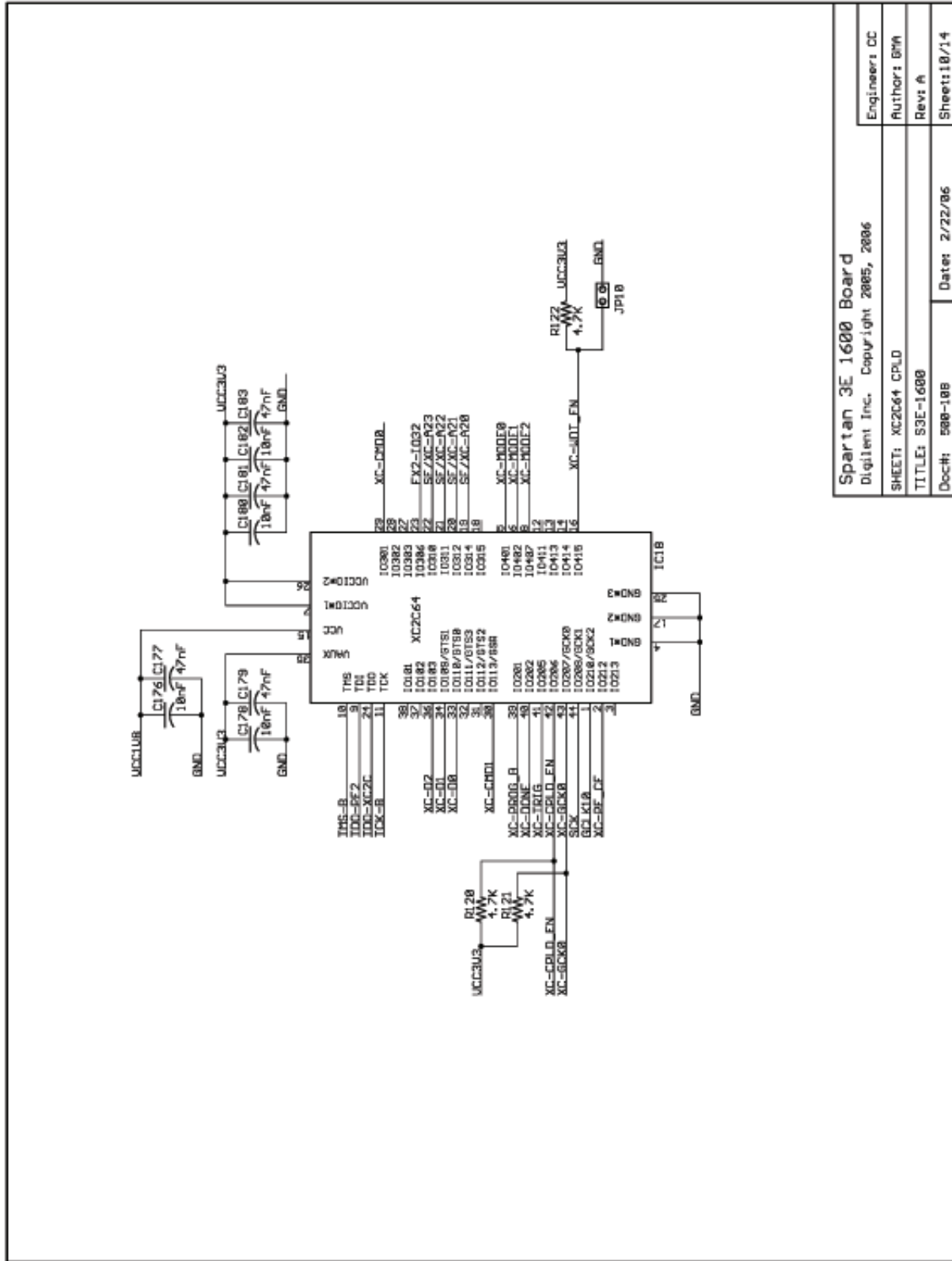
UG257\_A08\_060606

## XC2C64A CoolRunner-II CPLD

IC18 is a Xilinx XC2C64A CoolRunner-II CPLD. The CPLD primarily provides additional flexibility when configuring the FPGA from parallel NOR Flash and during MultiBoot configurations.

When the CPLD is loaded with the appropriate design, JP10 enables a watchdog timer in the CPLD used during fail-safe MultiBoot configurations.

See [Chapter 16, "XC2C64A CoolRunner-II CPLD,"](#) for more information.



Spartan 3E 1600 Board		Engineer: CC
Diligent Inc. Copyright 2005, 2006		Author: BVA
SHEET: XC2C64 CPLD		Rev: A
TITLE: S3E-1600		Date: 2/22/06
Doc#: 598-108		Sheet: 10/14

Figure 18-9: Schematic Sheet 10

UG257\_A09\_060606



## Linear Technology ADC and DAC

IC19 is a Linear Technology LTC1407A-1 two-channel ADC. IC20 is a Linear Technology LTC6912 programmable pre-amplifier (AMP) to condition the analog inputs to the ADC. See [Chapter 10, "Analog Capture Circuit,"](#) for additional information.

IC21 is a Linear Technology LTC2624 four-channel DAC. See [Chapter 9, "Digital to Analog Converter \(DAC\),"](#) for additional information.

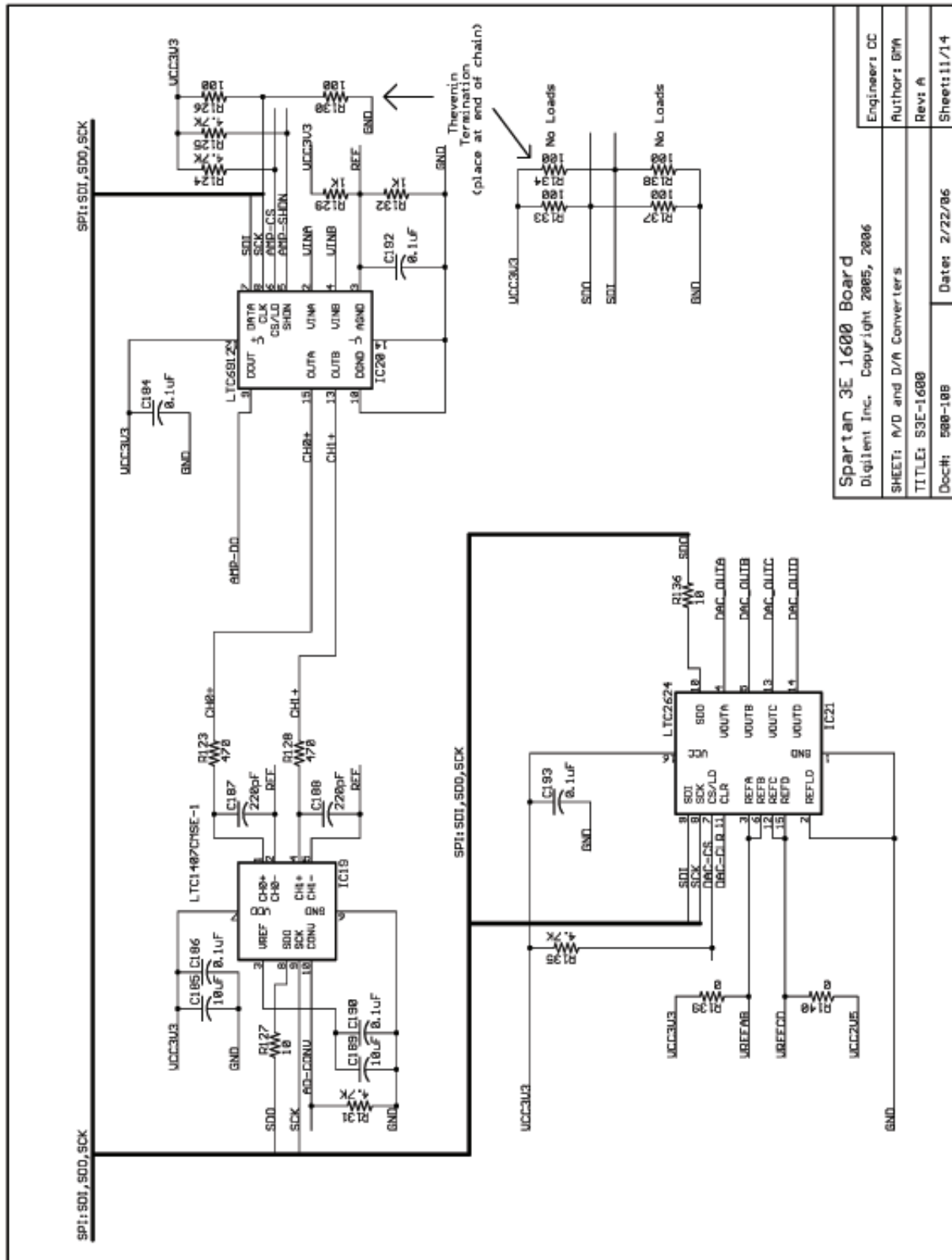


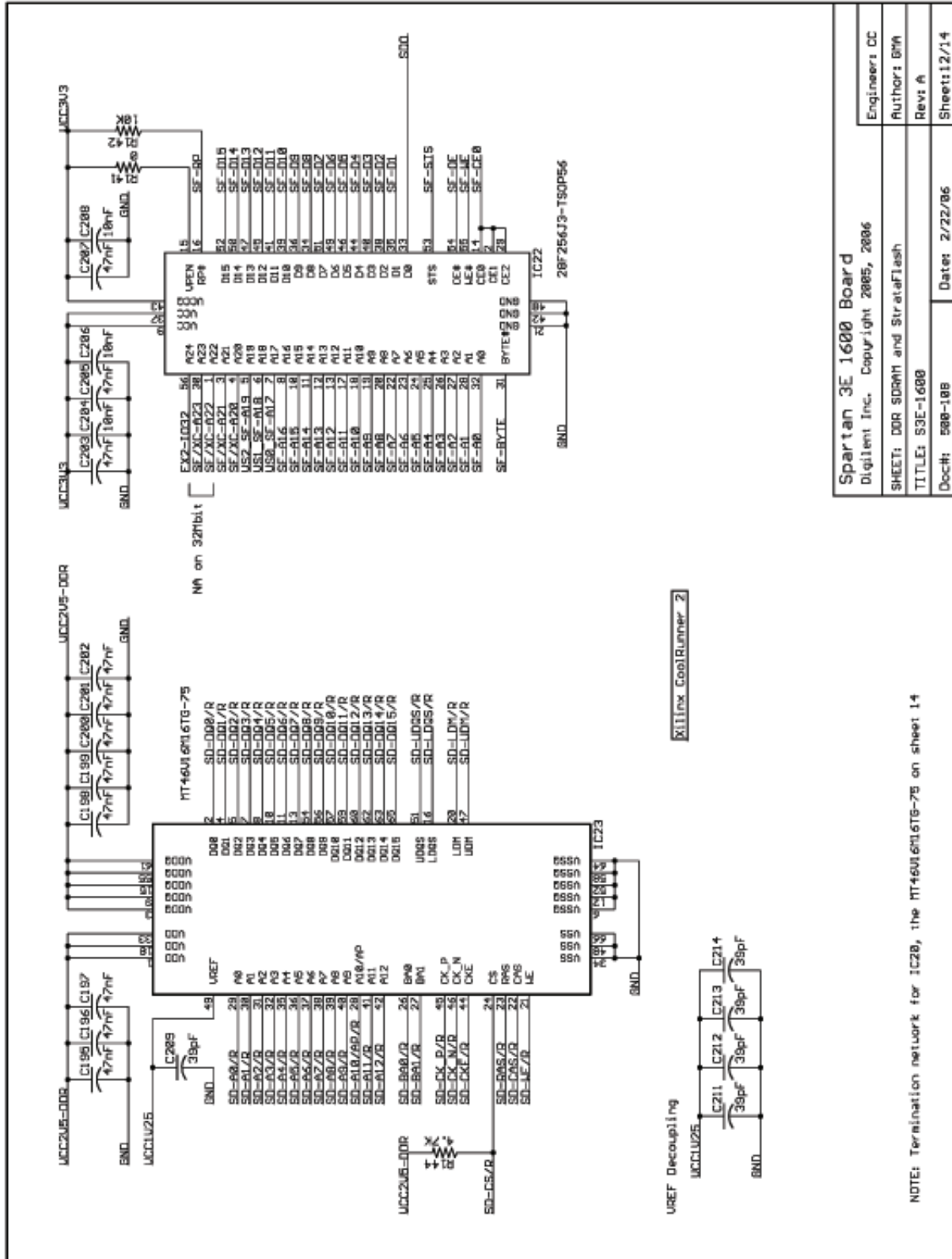
Figure 18-10: Schematic Sheet 11

<b>Spartan 3E 1600 Board</b>	
Digilent Inc. Copyright 2005, 2006	
SHEET: A/D and D/A Converters	Engineer: CC
TITLE: S3E-1600	Author: DM
Doc#: 598-10B	Rev: A
Date: 2/22/06	Sheet: 11/14

## Intel StrataFlash Parallel NOR Flash Memory and Micron DDR SDRAM

IC22 is a 128 Mbit (16 Mbyte) Intel StrataFlash parallel NOR Flash PROM. See [Chapter 11, "Intel StrataFlash Parallel NOR Flash PROM,"](#) for additional information.

IC23 is a 512 Mbit (64 Mbyte) Micron DDR SDRAM. See [Chapter 13, "DDR SDRAM,"](#) for additional information.



Spartan 3E 1600 Board	
Digilent Inc. Copyright 2005, 2006	
Engineers CC	Author: GWA
TITLE: S3E-1600	
Doc#: 598-108	Date: 2/22/06
Sheet: 12/14	

Figure 18-11: Schematic Sheet 12

UG257\_A11\_060606

## Buttons, Switches, Rotary Encoder, and Character LCD

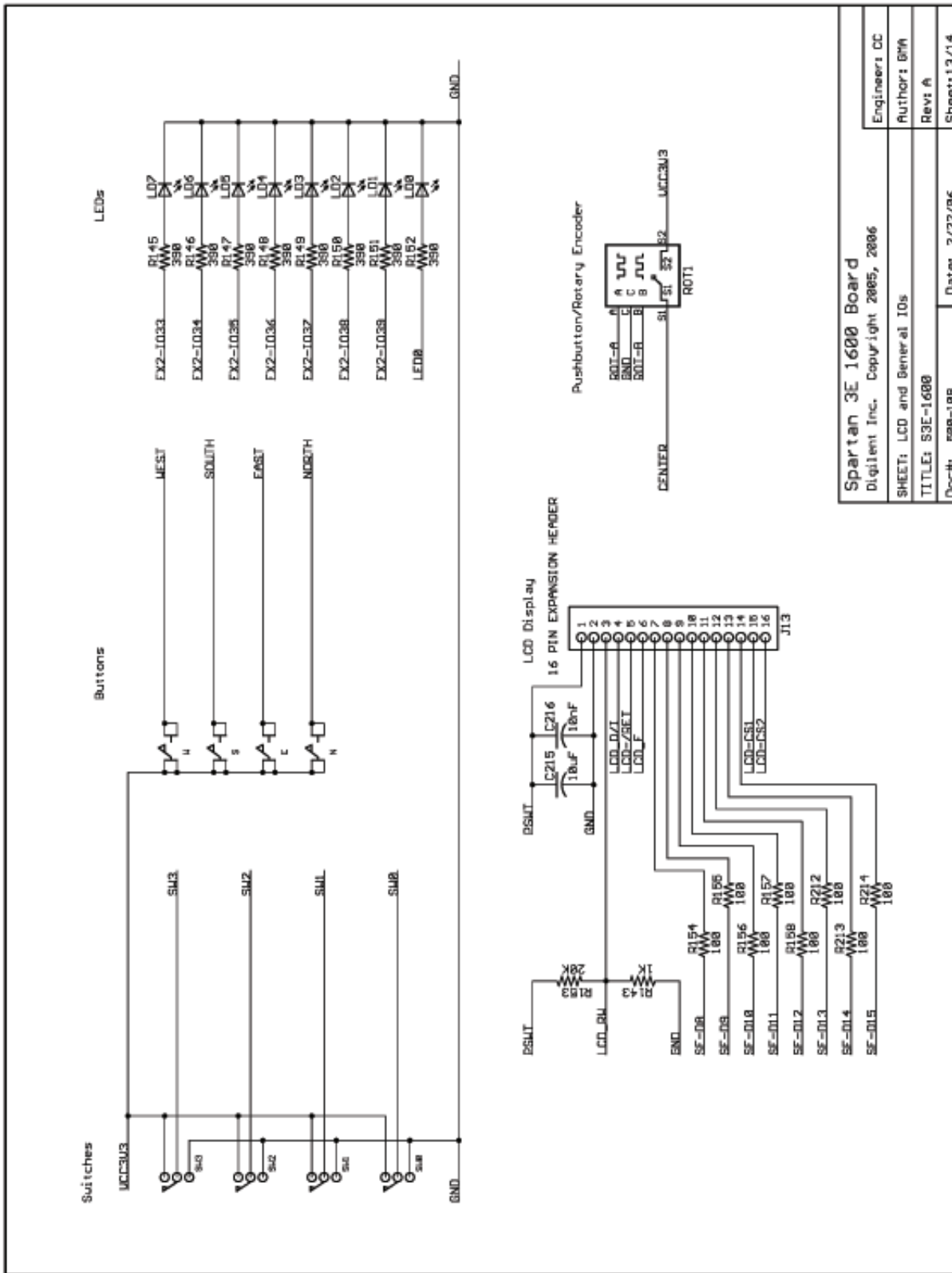
SW0, SW1, SW2, and SW3 are slide switches.

Push-button switches W, E, S, and N are located around the ROT1 push-button switch/rotary encoder.

LD0 through LD7 are discrete LEDs.

See [Chapter 2, “Switches, Buttons, and Knob,”](#) for additional information.

DISP1 is a 2x16 character LCD screen. See [Chapter 5, “Character LCD Screen,”](#) for additional information.



Engineers CC	
Author: DM	
Rev: A	
Sheet: 13/14	
<b>Spartan 3E 1600 Board</b>	
Digilent Inc. Copyright 2005, 2006	
SHEET: LCD and General I/Os	
TITLE: S3E-1600	
Doc#: 598-10B	Date: 2/22/06

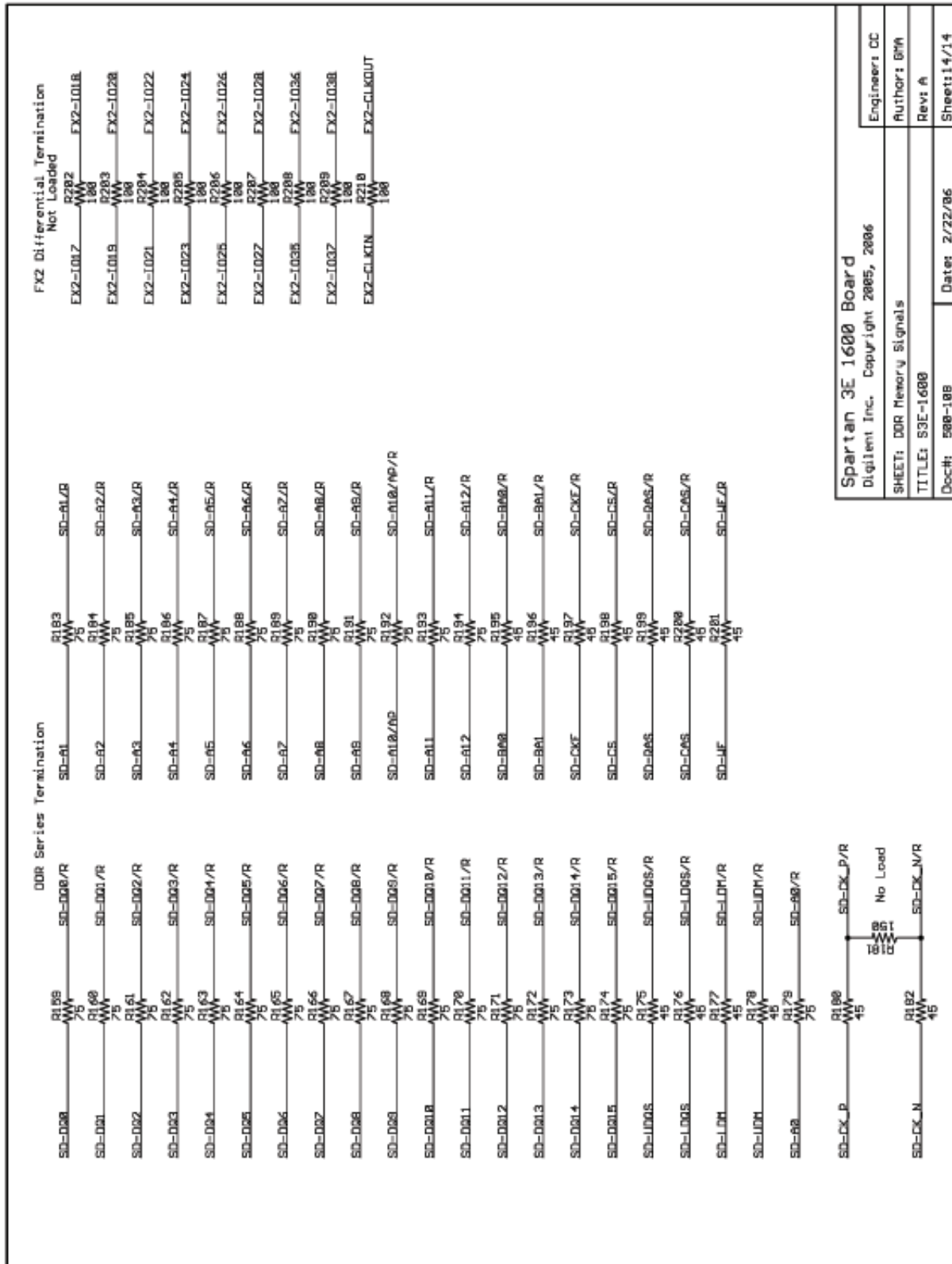
Figure 18-12: Schematic Sheet 13

UG257\_A12\_060606

## DDR SDRAM Series Termination and FX2 Connector Differential Termination

Resistors R160 through R201 represent the series termination resistors for the DDR SDRAM. See [Chapter 13, “DDR SDRAM,”](#) for additional information.

Resistors R202 through R210 are not loaded on the board. These landing pads provide optional connections for 100 $\Omega$  differential termination resistors. See [“Using Differential Inputs,”](#) [page 120](#) for additional information.



UG257\_A13\_060606

Figure 18-13: Schematic Sheet 14





## Example User Constraints File (UCF)

---

```
#####
### SPARTAN-3E MicroBlaze Development KIT BOARD CONSTRAINTS FILE
#####
# ==== FPGA Configuration Mode, INIT_B Pins (FPGA) ====
NET "FPGA_M0" LOC = "M10" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;
NET "FPGA_M1" LOC = "V11" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;
NET "FPGA_M2" LOC = "T10" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;
NET "FPGA_INIT_B" LOC = "T3" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 4 ;
NET "FPGA_RDWR_B" LOC = "U10" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 4 ;
NET "FPGA_HSWAP" LOC = "B3" | IOSTANDARD = LVCMOS33 ;
#
# ==== Clock inputs (CLK) ====
NET "CLK_50MHZ" LOC = "C9" | IOSTANDARD = LVCMOS33 ; # GCLK10
# Define clock period for 50 MHz oscillator (40%/60% duty-cycle)
NET "CLK_50MHZ" PERIOD = 20.0ns HIGH 40%;
#
NET "CLK_AUX_66MHZ" LOC = "B8" | IOSTANDARD = LVCMOS33 ; # GCLK8 Populated with 66MHz Osc
# Define clock period for 66 MHz oscillator (50%/50% duty-cycle)
NET "CLK_AUX_66MHZ" PERIOD = 14.9ns HIGH 50%;
#
NET "CLK_SMA" LOC = "A10" | IOSTANDARD = LVCMOS33 ;
#
# ==== RS-232 Serial Ports (RS232) ====
NET "RS232_DCE_RXD" LOC = "R7" | IOSTANDARD = LVTTTL ;
NET "RS232_DCE_TXD" LOC = "M14" | IOSTANDARD = LVTTTL | DRIVE = 8 | SLEW = SLOW ;
NET "RS232_DTE_RXD" LOC = "U8" | IOSTANDARD = LVTTTL ;
NET "RS232_DTE_TXD" LOC = "M13" | IOSTANDARD = LVTTTL | DRIVE = 8 | SLEW = SLOW ;
#
# ==== Rotary Pushbutton Switch (ROT) ====
NET "ROT_A" LOC = "K18" | IOSTANDARD = LVTTTL | PULLUP ;
NET "ROT_B" LOC = "G18" | IOSTANDARD = LVTTTL | PULLUP ;
NET "ROT_CENTER" LOC = "V16" | IOSTANDARD = LVTTTL | PULLDOWN ;
#
# ==== Pushbuttons (BTN) ====
NET "BTN_EAST" LOC = "H13" | IOSTANDARD = LVTTTL | PULLDOWN ;
NET "BTN_NORTH" LOC = "V4" | IOSTANDARD = LVTTTL | PULLDOWN ;
NET "BTN_SOUTH" LOC = "K17" | IOSTANDARD = LVTTTL | PULLDOWN ;
NET "BTN_WEST" LOC = "D18" | IOSTANDARD = LVTTTL | PULLDOWN ;
#
# ==== Slide Switches (SW) ====
NET "SW<0>" LOC = "L13" | IOSTANDARD = LVTTTL | PULLUP ;
NET "SW<1>" LOC = "L14" | IOSTANDARD = LVTTTL | PULLUP ;
NET "SW<2>" LOC = "H18" | IOSTANDARD = LVTTTL | PULLUP ;
NET "SW<3>" LOC = "N17" | IOSTANDARD = LVTTTL | PULLUP ;
#
# ==== Discrete LEDs (LED) ====
```

```

# These are shared connections with the FX connector
NET "LED<0>" LOC = "D4" | IOSTANDARD = SSTL2_I ;
NET "LED<1>" LOC = "C3" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ; # FX2-IO39
NET "LED<2>" LOC = "D6" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ; # FX2-IO38
NET "LED<3>" LOC = "E6" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ; # FX2-IO37
NET "LED<4>" LOC = "D13" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ; # FX2-IO36
NET "LED<5>" LOC = "A7" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ; # FX2-IO35
NET "LED<6>" LOC = "G9" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ; # FX2-IO34
NET "LED<7>" LOC = "A8" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ; # FX2-IO33
#
# ===== Character LCD (LCD) =====
NET "LCD_E" LOC = "M18" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "LCD_DI" LOC = "L18" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "LCD_RW" LOC = "L17" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "LCD_RET" LOC = "E3" | IOSTANDARD = SSTL2_I ;
NET "LCD_CS1" LOC = "P3" | IOSTANDARD = SSTL2_I ;
NET "LCD_CS2" LOC = "P4" | IOSTANDARD = SSTL2_I ;
# LCD data connections are shared with StrataFlash connections SF_D<15:8>
#NET "SF_D<8>" LOC = "R15" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
#NET "SF_D<9>" LOC = "R16" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
#NET "SF_D<10>" LOC = "P17" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
#NET "SF_D<11>" LOC = "M15" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
#NET "SF_D<12>" LOC = "M16" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
#NET "SF_D<13>" LOC = "P6" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
#NET "SF_D<14>" LOC = "R8" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
#NET "SF_D<15>" LOC = "T8" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
#
# ===== PS/2 Mouse/Keyboard Port (PS2) =====
NET "PS2_CLK" LOC = "G14" | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = SLOW ;
NET "PS2_DATA" LOC = "G13" | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = SLOW ;
#
# ===== Analog-to-Digital Converter (ADC) =====
# some connections shared with SPI Flash, DAC, ADC, and AMP
NET "AD_CONV" LOC = "P11" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 6 ;
# ===== Programmable Gain Amplifier (AMP) =====
# some connections shared with SPI Flash, DAC, ADC, and AMP
NET "AMP_CS" LOC = "N7" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 6 ;
NET "AMP_DOUT" LOC = "E18" | IOSTANDARD = LVCMOS33 ;
NET "AMP_SHDN" LOC = "P7" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 6 ;
#
# ===== Digital-to-Analog Converter (DAC) =====
# some connections shared with SPI Flash, DAC, ADC, and AMP
NET "DAC_CLR" LOC = "P8" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;
NET "DAC_CS" LOC = "N8" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;
#
# ===== 1-Wire Secure EEPROM (DS) =====
NET "DS_WIRE" LOC = "U4" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 8 ; # SD0
#
# ===== Ethernet PHY (E) =====
NET "E_COL" LOC = "U6" | IOSTANDARD = LVCMOS33 ;
NET "E_CRS" LOC = "U13" | IOSTANDARD = LVCMOS33 ;
NET "E_MDC" LOC = "P9" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;
NET "E_MDIO" LOC = "U5" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;
NET "E_RX_CLK" LOC = "V3" | IOSTANDARD = LVCMOS33 ;
NET "E_RX_DV" LOC = "V2" | IOSTANDARD = LVCMOS33 ;
NET "E_RXD<0>" LOC = "V8" | IOSTANDARD = LVCMOS33 ;
NET "E_RXD<1>" LOC = "T11" | IOSTANDARD = LVCMOS33 ;
NET "E_RXD<2>" LOC = "U11" | IOSTANDARD = LVCMOS33 ;
NET "E_RXD<3>" LOC = "V14" | IOSTANDARD = LVCMOS33 ;

```

```

NET "E_RXD<4>" LOC = "U14" | IOSTANDARD = LVCMOS33 ;
NET "E_TX_CLK" LOC = "T7" | IOSTANDARD = LVCMOS33 ;
NET "E_TX_EN" LOC = "P15" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;
NET "E_TXD<0>" LOC = "R11" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;
NET "E_TXD<1>" LOC = "T15" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;
NET "E_TXD<2>" LOC = "R5" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;
NET "E_TXD<3>" LOC = "T5" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;
NET "E_TXD<4>" LOC = "R6" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;
#
# ===== DDR SDRAM (SD) ===== (I/O Bank 3, VCCO=2.5V)
NET "SD_A<0>" LOC = "T1" | IOSTANDARD = SSTL2_I ;
NET "SD_A<1>" LOC = "R3" | IOSTANDARD = SSTL2_I ;
NET "SD_A<2>" LOC = "R2" | IOSTANDARD = SSTL2_I ;
NET "SD_A<3>" LOC = "P1" | IOSTANDARD = SSTL2_I ;
NET "SD_A<4>" LOC = "E4" | IOSTANDARD = SSTL2_I ;
NET "SD_A<5>" LOC = "H4" | IOSTANDARD = SSTL2_I ;
NET "SD_A<6>" LOC = "H3" | IOSTANDARD = SSTL2_I ;
NET "SD_A<7>" LOC = "H1" | IOSTANDARD = SSTL2_I ;
NET "SD_A<8>" LOC = "H2" | IOSTANDARD = SSTL2_I ;
NET "SD_A<9>" LOC = "N4" | IOSTANDARD = SSTL2_I ;
NET "SD_A<10>" LOC = "T2" | IOSTANDARD = SSTL2_I ;
NET "SD_A<11>" LOC = "N5" | IOSTANDARD = SSTL2_I ;
NET "SD_A<12>" LOC = "P2" | IOSTANDARD = SSTL2_I ;
NET "SD_BA<0>" LOC = "K5" | IOSTANDARD = SSTL2_I ;
NET "SD_BA<1>" LOC = "K6" | IOSTANDARD = SSTL2_I ;
NET "SD_CAS" LOC = "C2" | IOSTANDARD = SSTL2_I ;
NET "SD_CK_N" LOC = "J4" | IOSTANDARD = SSTL2_I ;
NET "SD_CK_P" LOC = "J5" | IOSTANDARD = SSTL2_I ;
NET "SD_CKE" LOC = "K3" | IOSTANDARD = SSTL2_I ;
NET "SD_CS" LOC = "K4" | IOSTANDARD = SSTL2_I ;
NET "SD_DQ<0>" LOC = "L2" | IOSTANDARD = SSTL2_I ;
NET "SD_DQ<1>" LOC = "L1" | IOSTANDARD = SSTL2_I ;
NET "SD_DQ<2>" LOC = "L3" | IOSTANDARD = SSTL2_I ;
NET "SD_DQ<3>" LOC = "L4" | IOSTANDARD = SSTL2_I ;
NET "SD_DQ<4>" LOC = "M3" | IOSTANDARD = SSTL2_I ;
NET "SD_DQ<5>" LOC = "M4" | IOSTANDARD = SSTL2_I ;
NET "SD_DQ<6>" LOC = "M5" | IOSTANDARD = SSTL2_I ;
NET "SD_DQ<7>" LOC = "M6" | IOSTANDARD = SSTL2_I ;
NET "SD_DQ<8>" LOC = "E2" | IOSTANDARD = SSTL2_I ;
NET "SD_DQ<9>" LOC = "E1" | IOSTANDARD = SSTL2_I ;
NET "SD_DQ<10>" LOC = "F1" | IOSTANDARD = SSTL2_I ;
NET "SD_DQ<11>" LOC = "F2" | IOSTANDARD = SSTL2_I ;
NET "SD_DQ<12>" LOC = "G6" | IOSTANDARD = SSTL2_I ;
NET "SD_DQ<13>" LOC = "G5" | IOSTANDARD = SSTL2_I ;
NET "SD_DQ<14>" LOC = "H6" | IOSTANDARD = SSTL2_I ;
NET "SD_DQ<15>" LOC = "H5" | IOSTANDARD = SSTL2_I ;
NET "SD_LDM" LOC = "J2" | IOSTANDARD = SSTL2_I ;
NET "SD_LDQS" LOC = "L6" | IOSTANDARD = SSTL2_I ;
NET "SD_RAS" LOC = "C1" | IOSTANDARD = SSTL2_I ;
NET "SD_UDM" LOC = "J1" | IOSTANDARD = SSTL2_I ;
NET "SD_UDQS" LOC = "G3" | IOSTANDARD = SSTL2_I ;
NET "SD_WE" LOC = "D1" | IOSTANDARD = SSTL2_I ;
# Path to allow connection to top DCM connection
NET "SD_CK_FB" LOC = "B9" | IOSTANDARD = LVCMOS33 ;
# Prohibit VREF pins
CONFIG PROHIBIT = D2;
CONFIG PROHIBIT = G4;
CONFIG PROHIBIT = J6;
CONFIG PROHIBIT = L5;

```

```

CONFIG PROHIBIT = R4;
# ==== Intel StrataFlash Parallel NOR Flash (SF) ====
NET "SF_A<0>" LOC = "H17" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_A<1>" LOC = "J13" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_A<2>" LOC = "J12" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_A<3>" LOC = "J14" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_A<4>" LOC = "J15" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_A<5>" LOC = "J16" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_A<6>" LOC = "J17" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_A<7>" LOC = "K14" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_A<8>" LOC = "K15" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_A<9>" LOC = "K12" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_A<10>" LOC = "K13" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_A<11>" LOC = "L15" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_A<12>" LOC = "L16" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_A<13>" LOC = "T18" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_A<14>" LOC = "R18" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_A<15>" LOC = "T17" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_A<16>" LOC = "U18" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_A<17>" LOC = "T16" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_A<18>" LOC = "U15" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_A<19>" LOC = "V15" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_A<20>" LOC = "T12" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_A<21>" LOC = "V13" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_A<22>" LOC = "V12" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_A<23>" LOC = "N11" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_A<24>" LOC = "A11" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_BYTE" LOC = "C17" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_CE0" LOC = "D16" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_D<1>" LOC = "P10" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_D<2>" LOC = "R10" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_D<3>" LOC = "V9" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_D<4>" LOC = "U9" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_D<5>" LOC = "R9" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_D<6>" LOC = "M9" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_D<7>" LOC = "N9" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_D<8>" LOC = "R15" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_D<9>" LOC = "R16" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_D<10>" LOC = "P17" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_D<11>" LOC = "M15" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_D<12>" LOC = "M16" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_D<13>" LOC = "P6" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_D<14>" LOC = "R8" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_D<15>" LOC = "T8" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_OE" LOC = "C18" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_STS" LOC = "B18" | IOSTANDARD = LVCMOS33 ;
NET "SF_WE" LOC = "D17" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
# ==== STMicro SPI serial Flash (SPI) ====
# some connections shared with SPI Flash, DAC, ADC, and AMP
NET "SPI_MISO" LOC = "N10" | IOSTANDARD = LVCMOS33 ;
NET "SPI_MOSI" LOC = "T4" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 6 ;
NET "SPI_SCK" LOC = "U16" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 6 ;
NET "SPI_SS_B" LOC = "U3" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 6 ;
NET "SPI_ALT_CS_JP11" LOC = "R12" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 6 ;
# ==== VGA Port (VGA) ====
NET "VGA_BLUE" LOC = "G15" | IOSTANDARD = LVTTTL | DRIVE = 8 | SLEW = FAST ;
NET "VGA_GREEN" LOC = "H15" | IOSTANDARD = LVTTTL | DRIVE = 8 | SLEW = FAST ;
NET "VGA_HSYNC" LOC = "F15" | IOSTANDARD = LVTTTL | DRIVE = 8 | SLEW = FAST ;
NET "VGA_RED" LOC = "H14" | IOSTANDARD = LVTTTL | DRIVE = 8 | SLEW = FAST ;

```

```

NET "VGA_VSYNC" LOC = "F14" | IOSTANDARD = LVTTTL | DRIVE = 8 | SLEW = FAST ;
#
# ===== FX2 Connector (FX2) =====
NET "FX2_CLKIN" LOC = "E10" | IOSTANDARD = LVCMOS33 ;
NET "FX2_CLKIO" LOC = "D9" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_CLKOUT" LOC = "D10" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_IO<1>" LOC = "B4" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_IO<2>" LOC = "A4" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_IO<3>" LOC = "D5" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_IO<4>" LOC = "C5" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_IO<5>" LOC = "A6" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_IO<6>" LOC = "B6" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_IO<7>" LOC = "E7" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_IO<8>" LOC = "F7" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_IO<9>" LOC = "D7" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_IO<10>" LOC = "C7" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_IO<11>" LOC = "F8" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_IO<12>" LOC = "E8" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
#NET "FX2_IO<13>" LOC = "F9" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
#NET "FX2_IO<14>" LOC = "E9" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
#NET "FX2_IO<15>" LOC = "D11" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
#NET "FX2_IO<16>" LOC = "C11" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
#NET "FX2_IO<17>" LOC = "F11" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
#NET "FX2_IO<18>" LOC = "E11" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
#NET "FX2_IO<19>" LOC = "E12" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
#NET "FX2_IO<20>" LOC = "F12" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_IO<21>" LOC = "A13" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_IO<22>" LOC = "B13" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_IO<23>" LOC = "A14" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_IO<24>" LOC = "B14" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_IO<25>" LOC = "C14" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_IO<26>" LOC = "D14" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_IO<27>" LOC = "A16" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_IO<28>" LOC = "B16" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_IO<29>" LOC = "E13" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_IO<30>" LOC = "C4" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_IO<31>" LOC = "B11" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "FX2_IO<32>" LOC = "A11" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
# The discrete LEDs are shared with the following 8 FX2 connections
# NET "FX2_IO<33>" LOC = "A8" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ; # LED7
# NET "FX2_IO<34>" LOC = "G9" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ; # LED6
# NET "FX2_IP<35>" LOC = "A7" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ; # LED5
# NET "FX2_IP<36>" LOC = "D13" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ; # LED4
# NET "FX2_IP<37>" LOC = "E6" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ; # LED3
# NET "FX2_IP<38>" LOC = "D6" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ; # LED2
# NET "FX2_IP<39>" LOC = "C3" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ; # LED1
NET "FX2_IP<40>" LOC = "C15" | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
# ===== 6-pin header J1 =====
# These are independent of the FX2 connector
#NET "J1<0>" LOC = "V7" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 6 ;
#NET "J1<1>" LOC = "E15" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 6 ;
#NET "J1<2>" LOC = "N14" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 6 ;
#NET "J1<3>" LOC = "N15" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 6 ;
# ===== 6-pin header J2 =====
# These are independent of the FX2 connector
#NET "J2<0>" LOC = "P12" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 6 ;
#NET "J2<1>" LOC = "N12" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 6 ;
#NET "J2<2>" LOC = "V6" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 6 ;
#NET "J2<3>" LOC = "V5" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 6 ;

```

```

# ==== 6-pin header J4 ====
# These are independent of the FX2 connector
#NET "J4<0>" LOC = "R14" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 6 ;
#NET "J4<1>" LOC = "T14" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 6 ;
#NET "J4<2>" LOC = "R13" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 6 ;
#NET "J4<3>" LOC = "P13" | IOSTANDARD = LVTTTL | SLEW = SLOW | DRIVE = 6 ;
#
# ==== J15 Input Only Connector ====
NET "INPUT_IO<1>" LOC = "A12" | PULLDOWN | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "INPUT_IO<2>" LOC = "A3" | PULLDOWN | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "INPUT_IO<3>" LOC = "B15" | PULLDOWN | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "INPUT_IO<4>" LOC = "A15" | PULLDOWN | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "INPUT_IO<5>" LOC = "C13" | PULLDOWN | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "INPUT_IO<6>" LOC = "D12" | PULLDOWN | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "INPUT_IO<7>" LOC = "F10" | PULLDOWN | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "INPUT_IO<8>" LOC = "G10" | PULLDOWN | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "INPUT_IO<9>" LOC = "C8" | PULLDOWN | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "INPUT_IO<10>" LOC = "D8" | PULLDOWN | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "INPUT_IO<11>" LOC = "A5" | PULLDOWN | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
NET "INPUT_IO<12>" LOC = "B5" | PULLDOWN | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;
#NET "INPUT_IO<13>" LOC = "E17" | PULLDOWN | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE =
8 ;
#NET "INPUT_IO<14>" LOC = "P15" | PULLDOWN | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE =
8 ;

#
# ==== Xilinx CPLD (XC) ====
NET "XC_CMD<0>" LOC = "P18" | IOSTANDARD = LVTTTL | DRIVE = 4 | SLEW = SLOW ;
NET "XC_CMD<1>" LOC = "N18" | IOSTANDARD = LVTTTL | DRIVE = 4 | SLEW = SLOW ;
NET "XC_CPLD_EN" LOC = "B10" | IOSTANDARD = LVTTTL ;
NET "XC_D<0>" LOC = "G16" | IOSTANDARD = LVTTTL | DRIVE = 4 | SLEW = SLOW ;
NET "XC_D<1>" LOC = "F18" | IOSTANDARD = LVTTTL | DRIVE = 4 | SLEW = SLOW ;
NET "XC_D<2>" LOC = "F17" | IOSTANDARD = LVTTTL | DRIVE = 4 | SLEW = SLOW ;
NET "XC_TRIG" LOC = "R17" | IOSTANDARD = LVCMOS33 ;
NET "XC_GCK0" LOC = "H16" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;

```

# XBee™/XBee-PRO™ OEM RF Modules

---

XBee/XBee-PRO OEM RF Modules  
RF Module Operation  
RF Module Configuration  
Appendices



## Product Manual v1.06

For OEM RF Module Part Numbers: XB24-...-001, XB24-...-002  
XBP24-...-001, XBP24-...-002

ZigBee™/IEEE® 802.15.4 OEM RF Modules by MaxStream, Inc.



**MaxStream®**

355 South 520 West, Suite 180  
Lindon, UT 84042  
Phone: (801) 765-9885  
Fax: (801) 765-9895

rf-xperts@maxstream.net  
www.MaxStream.net (live chat support)

M100232  
2005.10.28



**© 2005 MaxStream, Inc. All rights reserved**

No part of the contents of this manual may be transmitted or reproduced in any form or by any means without the written permission of MaxStream, Inc.

XBee™ and XBee-PRO™ are trademarks of MaxStream, Inc.

ZigBee™ is a registered trademark of the ZigBee Alliance.

**Technical Support:**

Phone: (801) 765-9885

Live Chat: [www.maxstream.net](http://www.maxstream.net)

E-mail: [rf-xperts@maxstream.net](mailto:rf-xperts@maxstream.net)

# Contents

<b>1. XBee/XBee-PRO OEM RF Modules</b>	<b>4</b>	<b>Appendix A: Agency Certifications</b>	<b>23</b>
<b>1.1. Key Features</b>	<b>4</b>	<b>FCC Certification</b>	<b>23</b>
1.1.1. Worldwide Acceptance	4	OEM Labeling Requirements	23
<b>1.2. Specifications</b>	<b>5</b>	FCC Notices	23
<b>1.3. Mechanical Drawings</b>	<b>5</b>	FCC-Approved Antennas (2.4 GHz)	24
<b>1.4. Pin Signals</b>	<b>6</b>	<b>European Certification (pending)</b>	<b>25</b>
<b>1.5. Electrical Characteristics</b>	<b>6</b>	OEM Labeling Requirements	25
<b>2. RF Module Operation</b>	<b>7</b>	Restrictions	25
<b>2.1. Serial Communications</b>	<b>7</b>	Declarations of Conformity	25
2.1.1. UART Data Flow	7	<b>Appendix B: Development Guide</b>	<b>26</b>
2.1.2. Flow Control	8	<b>Development Kit Contents</b>	<b>26</b>
<b>2.2. Modes of Operation</b>	<b>9</b>	Interfacing Options	26
2.2.1. Idle Mode	9	<b>RS-232 Interface Board</b>	<b>27</b>
2.2.2. Transmit & Receive Modes	9	Physical Interface	27
2.2.3. Sleep Mode	11	RS-232 Pin Signals	28
2.2.4. Command Mode	13	Wiring Diagrams	29
<b>3. RF Module Configuration</b>	<b>14</b>	Adapters	30
<b>3.1. Programming the RF Module</b>	<b>14</b>	<b>USB Interface Board</b>	<b>31</b>
3.1.1. Programming Examples	14	Physical Interface	31
3.1.2. Command Reference Tables	15	USB Pin Signals	31
<b>3.2. Command Descriptions</b>	<b>16</b>	<b>Appendix C: Additional Information</b>	<b>32</b>
		<b>1-Year Warranty</b>	<b>32</b>
		<b>Ordering Information</b>	<b>32</b>
		<b>Contact MaxStream</b>	<b>33</b>

# 1. XBee/XBee-PRO OEM RF Modules

XBee and XBee-PRO Modules were engineered to meet ZigBee/IEEE 802.15.4 standards and support the unique needs of low-cost, low-power wireless sensor networks. The modules require minimal power and provide reliable delivery of critical data between devices.

The modules operate within the ISM 2.4 GHz frequency band and are pin-for-pin compatible with each other.



## 1.1. Key Features

### High Performance, Low Cost

XBee

- Indoor/Urban: up to 100' (30 m)
- Outdoor line-of-sight: up to 300' (100 m)
- Transmit Power: 1 mW (0 dBm)
- Receiver Sensitivity: -92 dBm

XBee-PRO

- Indoor/Urban: up to 300' (100 m)
- Outdoor line-of-sight: up to 1 mile (1500 m)
- Transmit Power: 100 mW (20 dBm) EIRP
- Receiver Sensitivity: -100 dBm

RF Data Rate: 250,000 bps

### Advanced Networking & Security

Retries and Acknowledgements

DSSS (Direct Sequence Spread Spectrum)

Each direct sequence channels has over 65,000 unique network addresses available

Point-to-point, point-to-multipoint and peer-to-peer topologies supported

128-bit Encryption (downloadable firmware version coming soon)

Self-routing/Self-healing mesh networking (downloadable firmware version coming soon)

### Low Power

XBee

- TX Current: 45 mA (@3.3 V)
- RX Current: 50 mA (@3.3 V)
- Power-down Current: < 10  $\mu$ A

XBee-PRO

- TX Current: 270 mA (@3.3 V)
- RX Current: 55 mA (@3.3 V)
- Power-down Current: < 10  $\mu$ A

### Easy-to-Use

No configuration necessary for out-of box RF communications

Free X-CTU Software (Testing and configuration software)

AT Command Mode for simple configuration of module parameters

Small form factor

Network compatible with other ZigBee/802.15.4 devices

**Free & Unlimited Technical Support**

### 1.1.1. Worldwide Acceptance

**FCC Approval** (USA) Refer to Appendix A [p23] for FCC Requirements.

Systems that include XBee/XBee-PRO Modules inherit MaxStream's Certifications.

**ISM** (Industrial, Scientific & Medical) 2.4 GHz frequency band

Manufactured under **ISO 9001:2000** registered standards

XBee/XBee-PRO RF Modules are optimized for use in **US, Canada, Australia, Israel and Europe** (contact MaxStream for complete list of approvals).



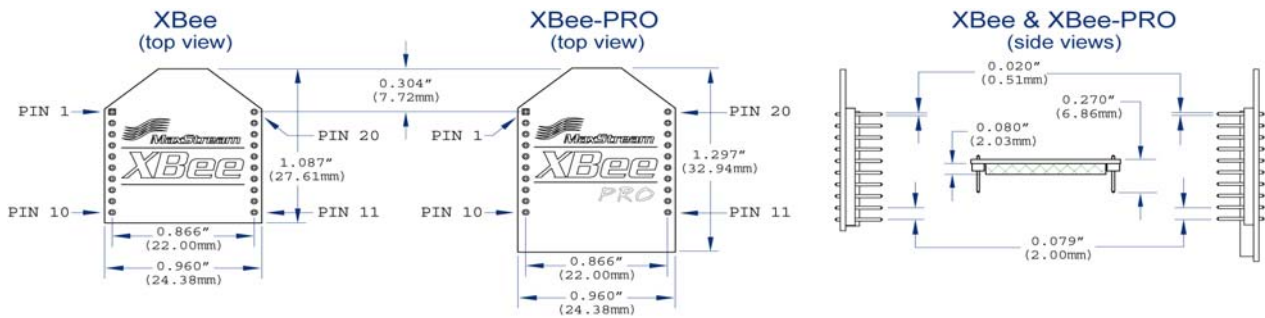
## 1.2. Specifications

Table 1-01. Specifications of the XBee/XBee-PRO OEM RF Modules

Specification	XBee	XBee-PRO
<b>Performance</b>		
Indoor/Urban Range	up to 100 ft. (30 m)	Up to 300' (100 m)
Outdoor RF line-of-sight Range	up to 300 ft. (100 m)	Up to 1 mile (1500 m)
Transmit Power Output	1mW (0 dBm)	60 mW (18 dBm) conducted, 100 mW (20 dBm) EIRP
RF Data Rate	250,000 bps	250,000 bps
Interface Data Rate (software selectable)	1200 - 115200 bps (non-standard baud rates also supported)	1200 - 115200 bps (non-standard baud rates also supported)
Receiver Sensitivity	-92 dBm (1% packet error rate)	-100 dBm (1% packet error rate)
<b>Power Requirements</b>		
Supply Voltage	2.8 – 3.4 V	2.8 – 3.4 V
Transmit Current (typical)	45 mA (@ 3.3 V)	270 mA (@ 3.3 V)
Receive Current (typical)	50 mA (@ 3.3 V)	55 mA (@ 3.3 V)
Power-down Current	< 10 µA	< 10 µA
<b>General</b>		
Operating Frequency	ISM 2.4 GHz	ISM 2.4 GHz
Dimensions	0.960" x 1.087" (2.438cm x 2.761cm)	0.960" x 1.297" (2.438cm x 3.294cm)
Operating Temperature	-40 to 85° C (industrial)	-40 to 85° C (industrial)
Antenna Options	U.FL Connector, Chip Antenna or Whip Antenna	U.FL Connector, Chip Antenna or Whip Antenna
<b>Networking &amp; Security</b>		
Supported Network Topologies	Point-to-Point, Point-to-Multipoint, Peer-to-Peer and Mesh (coming soon)	Point-to-Point, Point-to-Multipoint, Peer-to-Peer and Mesh (coming soon)
Number of Channels (software selectable)	16 Direct Sequence Channels	13 Direct Sequence Channels
Filtration Options	PAN ID, Channel and Source/Destination Addresses	PAN ID, Channel and Source/Destination Addresses
<b>Agency Approvals</b>		
FCC Part 15.247	OUR-XBEE	pending
Industry Canada (IC)	pending	pending
Europe	pending	pending

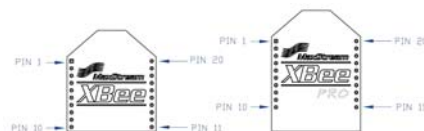
## 1.3. Mechanical Drawings

Figure 1-01. Mechanical drawings of the XBee/XBee-PRO OEM RF Modules (antenna options not shown)  
 XBee and XBee-PRO RF Modules are pin-for-pin compatible.



## 1.4. Pin Signals

**Figure 1-02. XBee/XBee-PRO RF Module Pin Number**  
(top sides shown - shields on bottom)



**Table 1-02. Pin Assignments for the XBee and XBee-PRO Modules**  
(Low-asserted signals are distinguished with a horizontal line above signal name.)

Pin #	Name	Direction	Description
1	VCC	-	Power supply
2	DOUT	Output	UART Data Out
3	DIN / <u>CONFIG</u>	Input	UART Data In
4	CD* / <u>DOUT_EN</u> * / DO8*	Output	Carrier Detect, TX_enable or Digital Output 8
5	<u>RESET</u>	Input	Module Reset
6	PWM0 / RSSI	Output	PWM Output 0 or RX Signal Strength Indicator
7	[reserved]	-	Do not connect
8	[reserved]	-	Do not connect
9	<u>DTR</u> / <u>SLEEP_RQ</u> / DI8	Input	Pin Sleep Control Line or Digital Input 8
10	GND	-	Ground
11	RF_TX* / AD4* / DIO4*	Either	Transmission Indicator, Analog Input 4 or Digital I/O 4
12	<u>CTS</u> * / DIO7*	Either	Clear-to-Send Flow Control or Digital I/O 7
13	ON / <u>SLEEP</u>	Output	Module Status Indicator
14	VREF*	Input	Voltage Reference for A/D Inputs
15	Associate / AD5* / DIO5*	Either	Associated Indicator, Analog Input 5 or Digital I/O 5
16	<u>RTS</u> * / AD6* / DIO6*	Either	Request-to-Send Flow Control, Analog Input 6 or Digital I/O 6
17	COORD_SEL* / AD3* / DIO3*	Either	Analog Input 3, Digital I/O 3 or Coordinator Select
18	AD2* / DIO2*	Either	Analog Input 2 or Digital I/O 2
19	AD1* / DIO1*	Either	Analog Input 1 or Digital I/O 1
20	AD0* / DIO0*	Either	Analog Input 0 or Digital I/O 0

\* Functions not supported at the time of this release.

### Design Notes:

- Minimum connections are: VCC, GND, DOUT and DIN
- Signal Direction is specified with respect to the module
- Module includes a 50k pull-up resistor attached to RESET
- Unused pins should be left disconnected.

## 1.5. Electrical Characteristics

**Table 1-03. DC Characteristics of the XBee & XBee-PRO (VCC = 2.8 - 3.4 VDC)**

Symbol	Parameter	Condition	Min	Typical	Max	Units
V <sub>IL</sub>	Input Low Voltage	All Digital Inputs	-	-	0.35 * VCC	V
V <sub>IH</sub>	Input High Voltage	All Digital Inputs	0.7 * VCC	-	-	V
V <sub>OL</sub>	Output Low Voltage	I <sub>OL</sub> = 2 mA, VCC >= 2.7 V	-	-	0.5	V
V <sub>OH</sub>	Output High Voltage	I <sub>OH</sub> = -2 mA, VCC >= 2.7 V	VCC - 0.5	-	-	V
I <sub>IN</sub>	Input Leakage Current	V <sub>IN</sub> = VCC or GND, all inputs, per pin	-	0.025	1	uA
I <sub>OZ</sub>	High Impedance Leakage Current	V <sub>IN</sub> = VCC or GND, all I/O High-Z, per pin	-	0.025	1	uA
TX	Transmit Current	VCC = 3.3 V	-	45 (XBee) 270 (PRO)	-	mA
RX	Receive Current	VCC = 3.3 V	-	50 (XBee) 55 (PRO)	-	mA
PWR-DWN	Power-down Current	SM parameter = 1	-	< 10	-	uA

## 2. RF Module Operation

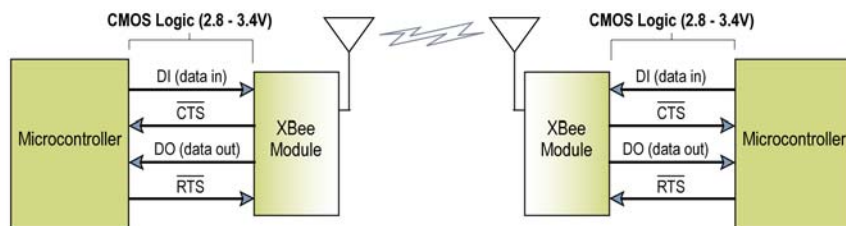
### 2.1. Serial Communications

The XBee/XBee-PRO OEM RF Modules interface to a host device through a logic-level asynchronous serial port. Through its serial port, the module can communicate with any logic and voltage compatible UART; or through a level translator to any serial device (For example: RS-232/485/422 or USB interface board).

#### 2.1.1. UART Data Flow

Devices that have a UART interface can connect directly to the pins of the RF module as shown in the figure below.

**Figure 2-01. Figure 2-01.System Data Flow Diagram in a UART-interfaced environment**  
(Low-asserted signals distinguished with horizontal line over signal name.)

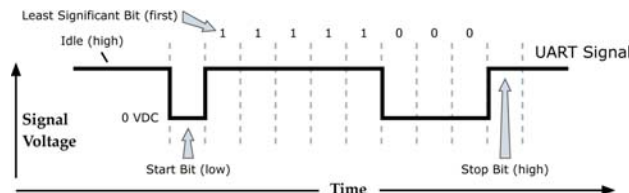


#### Serial Data

Data enters the module UART through the DI pin (pin 3) as an asynchronous serial signal. The signal should idle high when no data is being transmitted.

Each data byte consists of a start bit (low), 8 data bits (least significant bit first) and a stop bit (high). The following figure illustrates the serial bit pattern of data passing through the module.

**Figure 2-02. UART data packet 0x1F (decimal number "31") as transmitted through the RF module**  
Example Data Format is 8-N-1 (bits - parity - # of stop bits)



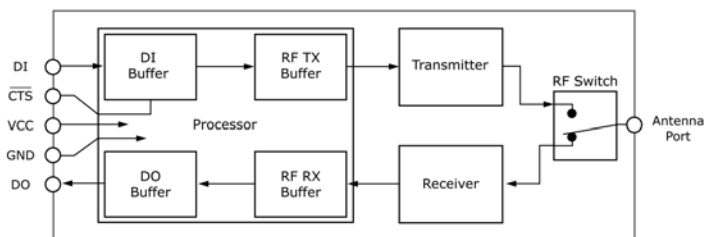
The module UART performs tasks, such as timing and parity checking, that are needed for data communications. Serial communications depend on the two UARTs to be configured with compatible settings (baud rate, parity, start bits, stop bits, data bits)

Both the module and host (PC) settings can be viewed and adjusted using MaxStream's proprietary X-CTU Software. Use the "PC Settings" tab to configure host settings. Use the "Terminal" or "RF Module Configuration" tab to configure the module settings.

**NOTE:** Failure to enter AT Command Mode is most commonly due to baud rate mismatch. Ensure the 'Baud' setting on the "PC Settings" tab matches the interface data rate of the RF module (by default, BD parameter = 3 (which is associated to 9600 bps)).

## 2.1.2. Flow Control

Figure 2-03. Internal Data Flow Diagram



### DI (Data In) Buffer

When serial data enters the RF module through the DI pin (pin 3), the data is stored in the DI Buffer until it can be processed.

**Hardware Flow Control ( $\overline{\text{CTS}}$ ).** When the DI buffer is 17 bytes away from being full; by default, the module de-asserts  $\overline{\text{CTS}}$  (high) to signal to the host device to stop sending data [refer to D7 (DIO7 Configuration) parameter].  $\overline{\text{CTS}}$  is re-asserted after the DI Buffer has 34 bytes of memory available.

#### How to eliminate the need for flow control:

1. Send messages that are smaller than the DI buffer size.
2. Interface at a lower baud rate [BD (Interface Data Rate) parameter] than the throughput data rate.

#### Case in which the DI Buffer may become full and possibly overflow:

If the module is receiving a continuous stream of RF data, any serial data that arrives on the DI pin is placed in the DI Buffer. The data in the DI buffer will be transmitted over-the-air when the module is no longer receiving RF data in the network.

NOTE:  $\overline{\text{CTS}}$  hardware flow control is not supported in this release (v1.06). Contact MaxStream support to download firmware that supports this function.

### DO (Data Out) Buffer

When RF data is received, the data enters the DO buffer and is sent out the serial port to a host device. Once the DO Buffer reaches capacity, any additional incoming RF data is lost.

**Hardware Flow Control ( $\overline{\text{RTS}}$ ).** If  $\overline{\text{RTS}}$  is enabled for flow control (D6 (DIO6 Configuration) Parameter = 1), data will not be sent out the DO Buffer as long as  $\overline{\text{RTS}}$  (pin 16) is de-asserted.

#### Two cases in which the DO Buffer may become full and possibly overflow:

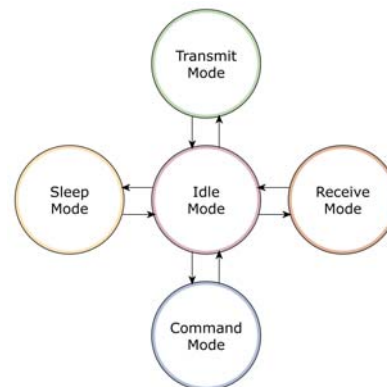
1. If the RF data rate is set higher than the interface data rate of the module, the module will receive data from the transmitting module faster than it can send the data to the host.
2. If the host does not allow the module to transmit data out from the DO buffer because of being held off by hardware or software flow control.

NOTE:  $\overline{\text{RTS}}$  hardware flow control is not supported in this release (v1.06). Contact MaxStream support to download firmware that supports this function.

## 2.2. Modes of Operation

XBee/XBee-PRO RF Modules operate in five modes.

**Figure 2-04. XBee/XBee-PRO RF Module Modes of Operation**



### 2.2.1. Idle Mode

When not receiving or transmitting data, the RF module is in Idle Mode. The RF module shifts into the other modes of operation under the following conditions:

- Transmit Mode: Serial data is received in the DI Buffer
- Receive Mode: Valid RF data is received through the antenna
- Sleep Mode: Sleep Mode condition is met
- Command Mode: Command Mode Sequence is issued

### 2.2.2. Transmit & Receive Modes

#### Addressing

When communication occurs between two networked devices, each data packet contains a <Source Address> and a <Destination Address> field. The XBee/XBee-PRO RF Module conforms to the 802.15.4 specification and supports both short 16-bit addresses and long 64-bit addresses. A unique 64-bit IEEE source address is assigned at the factory and can be read with the SL (Serial Number Low) and SH (Serial Number High) parameters. Short addressing must be configured manually. An RF module will use its unique 64-bit address as its Source Address if its MY value is "0xFFFF" or "0xFFFE".

To send a packet to a specific RF module using 64-bit addressing, set the Destination Address (DL + DH) to match the Source Address (SL + SH) of the intended destination RF module. To send a packet to a specific RF module using 16-bit addressing, set the DL (Destination Address Low) parameter to the MY (Source Address) parameter and set the DH (Destination Address High) parameter to "0".

#### Unicast Mode

Unicast Mode enables acknowledged communications. While in this mode, receiving modules send an ACK (acknowledgement) of RF packet reception to the transmitter. If the transmitting module does not receive the ACK, the transmitter will re-send the packet up to three times until the ACK is received.

Unicast Mode is the only mode that supports retries.

**Short 16-bit addresses.** The module can be configured to use short 16-bit addresses as the Source Address by setting (MY < 0xFFFE). Setting the DH parameter (DH = 0) will configure the Destination Address to be a short 16-bit address (if DL < 0xFFFE). For two modules to communicate using short addressing, the Destination Address of the transmitter module must match the MY parameter of the receiver.



The following table shows a sample network configuration that would enable Unicast Mode communications using 16-bit short addresses.

**Table 2-01. Sample Unicast Configuration (using 16-bit addressing)**

Parameter	RF Module 1	RF Module 2
MY (Source Address)	0x01	0x02
DH (Destination Address High)	0	0
DL (Destination Address Low)	0x02	0x01

**Long 64-bit addresses.** The RF module’s serial number (SL parameter concatenated to the SH parameter) can be used as a 64-bit source address when the MY (16-bit Source Address) parameter is disabled. When the MY parameter is disabled (set MY = 0xFFFF or 0xFFFE), the module’s source address is set to the 64-bit IEEE address stored in the SH and SL parameters.

When an End Device associates to a Coordinator, its MY parameter is set to 0xFFFE to enable 64-bit addressing. The 64-bit address of the module is stored as SH and SL parameters. To send a packet to a specific module, the Destination Address (DL + DH) on one module must match the Source Address (SL + SH) of the other.

**Broadcast Mode**

Any RF module will accept a packet that contains a broadcast address. When configured to operate in Broadcast Mode, receiving modules do not send ACKs (Acknowledgements) and transmitting RF modules do not automatically re-send packets as is the case in Unicast Mode.

To send a broadcast packet to all modules regardless of 16-bit or 64-bit addressing, set destination addresses of all the modules as shown below.

Sample Configuration (All modules in the network):

- DL (Destination Low Address) = 0x0000FFFF
- DH (Destination High Address) = 0x00000000

NOTE: When programming the module, parameters are entered in hexadecimal notation (without the “0x” prefix). Leading zeros may be omitted.

### 2.2.3. Sleep Mode

Sleep Modes enable the RF module to enter states of low-power consumption when not in use. In order to enter Sleep Mode, one of the following conditions must be met (in addition to the module having a non-zero SM parameter value):

- Sleep\_RQ (pin 9) is asserted.
- The module is idle (no data transmission or reception) for the amount of time defined by the ST (Time before Sleep) parameter. [NOTE: ST is only active when SM = 4-5.]

Table 2-02. Sleep Mode Configurations

Sleep Mode Setting	Transition into Sleep Mode	Transition out of Sleep Mode (wake)	Characteristics	Related Commands	Power Consumption
Pin Hibernate (SM = 1)	Assert (high) Sleep_RQ (pin 9)	De-assert (low) Sleep_RQ	Pin/Host-controlled / NonBeacon systems only / Lowest Power	(SM)	< 10 $\mu$ A (@3.0 VCC)
Pin Doze (SM = 2)	Assert (high) Sleep_RQ (pin 9)	De-assert (low) Sleep_RQ	Pin/Host-controlled / NonBeacon systems only / Fastest Wake-up	(SM)	< 50 $\mu$ A
Cyclic Sleep (SM = 4 - 5)	Automatic transition to Sleep Mode as defined by the SM (Sleep Mode) and ST (Time before Sleep) parameters.	Transition occurs after the cyclic sleep time interval elapses. The time interval is defined by the SP (Cyclic Sleep Period) parameter.	RF Module wakes in pre-determined time intervals to detect if RF data is present / When SM = 5, NonBeacon systems only	(SM), SP, ST	< 50 $\mu$ A when sleeping

The SM command is central to setting Sleep Mode configurations. By default, Sleep Modes are disabled (SM = 0) and the module remains in Idle/Receive Mode. When in this state, the module is constantly ready to respond to serial or RF activity.

#### Pin/Host-controlled Sleep Modes

##### Pin Hibernate (SM = 1)

- Pin/Host-controlled
- Typical power-down current: < 10  $\mu$ A (@3.0 VCC)
- Wake-up time: 13.2 msec

Pin Hibernate Mode minimizes quiescent power (power consumed when in a state of rest or inactivity). This mode is voltage level-activated; when Sleep\_RQ is asserted, the module will finish any transmit, receive or association activities, enter Idle Mode and then enter a state of sleep. The module will not respond to either serial or RF activity while in pin sleep.

To wake a sleeping module operating in Pin Hibernate Mode, de-assert Sleep\_RQ (pin 9). The module will wake when Sleep\_RQ is de-asserted and is ready to transmit or receive when the  $\overline{\text{CTS}}$  line is low.

##### Pin Doze (SM = 2)

- Pin/Host-controlled
- Typical power-down current: < 50  $\mu$ A
- Wake-up time: 2 msec

Pin Doze Mode functions as does Pin Hibernate Mode; however, Pin Doze features faster wake-up time and higher power consumption.

## Cyclic Sleep Modes

---

### Cyclic Sleep Remote (SM = 4)

- Typical Power-down Current: < 50  $\mu$ A (when asleep)
- Wake-up time: 2 msec

The Cyclic Sleep Modes allow modules to periodically check for RF data. When the SM parameter is set to '4', the module is configured to sleep, then wakes once a cycle to check for data from a module configured as a Cyclic Sleep Coordinator (SM = 6). The Cyclic Sleep Remote sends a poll request to the coordinator at a specific interval set by the SP (Cyclic Sleep Period) parameter. The coordinator will transmit any queued data addressed to that specific remote upon receiving the poll request. If no data is queued for the remote, the coordinator will not transmit and the remote will return to sleep for another cycle. If queued data is transmitted back to the remote, it will stay awake to allow for back and forth communication until the ST (Time before Sleep) timer expires.

Also note that  $\overline{\text{CTS}}$  will go low each time the remote wakes, allowing for communication initiated by the remote host if desired.

### Cyclic Sleep Remote with Pin Wake-up (SM = 5)

Use this mode to wake a sleeping remote module through either the RF interface or by the de-assertion of Sleep\_RQ for event-driven communications. The cyclic sleep mode works as described above (Cyclic Sleep Remote) with the addition of a pin-controlled wake-up at the remote module. The module will wake quickly when a low is detected and set  $\overline{\text{CTS}}$  low as soon as it is ready to transmit or receive. Any activity will reset the ST (Time before Sleep) timer so the module will go back to sleep only after Sleep\_RQ is asserted and there is no activity for the duration of the timer.

## 2.2.4. Command Mode

To modify or read RF Module parameters, the module must first enter into Command Mode - a state in which incoming characters are interpreted as commands. Two command modes are supported: AT Command Mode and AT+ Command Mode.

A robust set of AT Commands is available for programming and customizing the module.

### AT Command Mode

#### To Enter AT Command Mode:

Send the 3-character command sequence “+++” and observe guard times before and after the command characters. [Refer to the “Default AT Command Mode Sequence” below.]

Default AT Command Mode Sequence (for transition to Command Mode):

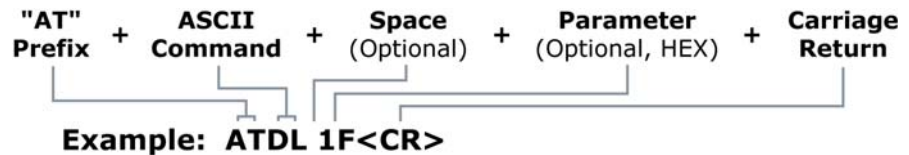
- No characters sent for one second [GT (Guard Times) parameter = 0x3E8]
- Input three plus characters (“+++”) within one second [CC (Command Sequence Character) parameter = 0x2B.]
- No characters sent for one second [GT (Guard Times) parameter = 0x3E8]

All of the parameter values in the sequence can be modified to reflect user preferences.

#### To Send AT Commands:

Send AT commands and parameters using the syntax shown below.

Figure 2-05. Syntax for sending AT Commands



To read a parameter value stored in the RF module’s register, leave the parameter field blank.

The preceding example would change the RF module Destination Address (Low) to “0x1F”. To store the new value to non-volatile (long term) memory, subsequently send the WR (Write) command.

For modified parameter values to persist in the module’s registry, changes must be saved to non-volatile memory using the WR (Write) Command. Otherwise, parameters are restored to previously saved values after the module is powered off and then on again (or re-booted).

**System Response.** When a command is sent to the RF module, the module will parse and execute the command. Upon successful execution of a command, the module returns an “OK” message. If execution of a command results in an error, the module returns an “ERROR” message.

#### To Exit AT Command Mode:

1. Send ATCN (Exit Command Mode) Command.  
[OR]
2. If no valid AT Commands are received within the time specified by CT (Command Mode Timeout) Command, the RF module automatically returns to Idle Mode.

For an example of programming the RF module using AT Commands and descriptions of each configurable parameter, refer to the “RF Module Configuration” chapter [p14].

## 3. RF Module Configuration

### 3.1. Programming the RF Module

Refer to the “Command Mode” section [p13] for more information about entering Command Mode, sending AT commands and exiting Command Mode.

#### 3.1.1. Programming Examples

##### Setup

The programming examples in this section require the installation of MaxStream's X-CTU Software and a serial connection to a PC. (MaxStream stocks RS-232 and USB boards to facilitate interfacing to a PC.)

1. Install MaxStream's X-CTU Software to a PC by double-clicking the "setup\_X-CTU.exe" file. (The file is located on the MaxStream CD and under the 'Software' section of the following web page: [www.maxstream.net/helpdesk/download.php](http://www.maxstream.net/helpdesk/download.php))
2. Mount the RF module to an interface board, then connect the module assembly to a PC.
3. Launch the X-CTU Software and select the 'PC Settings' tab. Verify the baud and parity settings of the Com Port match those of the RF module.

NOTE: Failure to enter AT Command Mode is most commonly due to baud rate mismatch. Ensure the 'Baud' setting on the 'PC Settings' tab matches the interface data rate of the RF module (by default, BD parameter = 3 (which corresponds to 9600 bps)).

##### Sample Configuration: Modify RF Module Destination Address

Example: Utilize the 'Terminal' tab of the X-CTU Software to change the RF module's DL (Destination Address Low) parameter and save the new address to non-volatile memory.

After establishing a serial connection between the RF module and a PC [refer to the 'Setup' section above], select the 'Terminal' tab of the X-CTU Software and enter the following command lines ('CR' stands for carriage return):

Method 1 (One line per command)

Send AT Command	System Response
+++	OK <CR> (Enter into Command Mode)
ATDL <Enter>	{current value} <CR> (Read Destination Address Low)
ATDL1AOD <Enter>	OK <CR> (Modify Destination Address Low)
ATWR <Enter>	OK <CR> (Write to non-volatile memory)
ATCN <Enter>	OK <CR> (Exit Command Mode)

Method 2 (Multiple commands on one line)

Send AT Command	System Response
+++	OK <CR> (Enter into Command Mode)
ATDL <Enter>	{current value} <CR> (Read Destination Address Low)
ATDL1AOD,WR,CN <Enter>	OK <CR> (Execute commands)

##### Sample Configuration: Restore RF Module Defaults

Example: Utilize the 'Modem Configuration' tab of the X-CTU Software to restore default parameter values of the RF module.

After establishing a connection between the RF module and a PC [refer to the 'Setup' section above], select the 'Modem Configuration' tab of the X-CTU Software.

1. Select the 'Read' button.
2. Select the 'Restore' button.

### 3.1.2. Command Reference Tables

**Table 3-01. XBee/XBee-PRO Commands** (RF modules expect numerical values in hexadecimal. Hexadecimal values are designated by the “0x” prefix. Decimal equivalents are designated by the “d” suffix.)

AT Command	Command Category	Name and Description	Parameter Range	Default
BD	Serial Interfacing	<b>Interface Data Rate.</b> Set/Read the serial interface data rate for communications between the RF module serial port and host.	0 - 7 (custom rates also supported)	3
CC	AT Command Mode Options	<b>Command Sequence Character.</b> Set/Read the ASCII character value to be used between Guard Times of the AT Command Mode Sequence (GT+CC+GT). The AT Command Mode Sequence enters the RF module to AT Command Mode.	0 - 0xFF	0x2B (+' ASCII)
CH	Networking & Security	<b>Channel.</b> Set/Read the channel number used for transmitting and receiving between RF modules. Uses 802.15.4 protocol channel numbers.	0x0B - 0x1A (XBee) 0x0C - 0x18 (XBee-PRO)	0x0C (12d)
CN	AT Command Mode Options	<b>Exit Command Mode.</b> Explicitly exit AT Command Mode.	-	-
CT	AT Command Mode Options	<b>Command Mode Timeout.</b> Set/Read the period of inactivity (no valid commands received) after which the RF module automatically exits AT Command Mode and returns to Idle Mode.	2 - 0xFFFF [x 100 ms]	0x64 (100d)
DB	Diagnostics	<b>Received Signal Strength.</b> Read signal level [in dB] of last good packet received (RSSI). Absolute value is reported. (For example: 0x58 = -88 dBm) Reported value is accurate between -40 dBm and RX sensitivity.	0 - 0x64 [read-only]	-
DH	Networking & Security	<b>Destination Address High.</b> Set/Read the upper 32 bits of the 64-bit destination address. When combined with DL, it defines the destination address used for transmission. To transmit using a 16-bit address, set DH parameter to zero and DL less than 0xFFFF. 0x0000000000000000 is the broadcast address for the PAN.	0 - 0xFFFFFFFF	0
DL	Networking & Security	<b>Destination Address Low.</b> Set/Read the lower 32 bits of the 64-bit destination address. When combined with DH, DL defines the destination address used for transmission. To transmit using a 16-bit address, set DH parameter to zero and DL less than 0xFFFF. 0x0000000000000000 is the broadcast address for the PAN.	0 - 0xFFFFFFFF	0
GT	AT Command Mode Options	<b>Guard Times.</b> Set required period of silence before and after the Command Sequence Characters of the AT Command Mode Sequence (GT+ CC + GT). The period of silence is used to prevent inadvertent entrance into AT Command Mode.	0x02 - 0xFFFF [x 1 ms]	0x3E8 (1000d)
ID	Networking & Security	<b>PAN ID.</b> Set/Read the PAN (Personal Area Network) ID. 0xFFFF indicates a message for all PANs.	0xFFFF	0x3332 (13106d)
MY	Networking & Security	<b>16-bit Source Address.</b> Set/Read the RF module 16-bit source address. Set MY = 0xFFFF to disable reception of packets with 16-bit addresses. 64-bit source address (serial number) and broadcast address (0x0000000000000000) is always enabled.	0 - 0xFFFF	0
P0	Diagnostics	<b>PWM0 Configurations.</b> Select/Read function for PWM0.	0 - 1	1
PL	RF Interfacing	<b>Power Level.</b> Select/Read power level at which the RF module transmits.	0 - 4	4
RE	(Special)	<b>Restore Defaults.</b> Restore RF module parameters to factory defaults. Follow with WR command to save values to non-volatile memory.	-	-
RN	Networking & Security	<b>Random Delay Slots.</b> Set/Read the minimum value of the back-off exponent in the CSMA-CA algorithm that is used for collision avoidance. If RN = 0, collision avoidance is disabled during the first iteration of the algorithm (802.15.4 - macMinBE).	0 - 3	0
RO	Serial Interfacing	<b>Packetization Timeout.</b> Set/Read number of character times of inter-character delay required before transmission. Set to zero to transmit characters as they arrive instead of buffering them into one RF packet.	0 - 0xFF [x character times]	3
RP	Diagnostics	<b>RSSI PWM Timer.</b> Enable a PWM (pulse width modulation) output (on pin 3 of the RF modules) which shows RX signal strength.	0 - 0xFF [x 100 ms]	0x28 (40d)
SH	Diagnostics	<b>Serial Number High.</b> Read high 32 bits of the RF module's unique IEEE 64-bit address. 64-bit source address is always enabled.	0 - 0xFFFFFFFF [read-only]	Factory-set
SL	Diagnostics	<b>Serial Number Low.</b> Read low 32 bits of the RF module's unique IEEE 64-bit address. 64-bit source address is always enabled.	0 - 0xFFFFFFFF [read-only]	Factory-set
SM	Sleep (Low Power)	<b>Sleep Mode.</b> Set/Read Sleep Mode configurations.	0 - 5	0
SP	Sleep (Low Power)	<b>Cyclic Sleep Period.</b> Set/Read sleep period for cyclic sleeping remotes. Maximum sleep period is 268 seconds (0x68B0).	0x01 - 0x68B0 [x 10 ms]	0x64 (100d)
ST	Sleep (Low Power)	<b>Time before Sleep.</b> Set/Read time period of inactivity (no serial or RF data is sent or received) before activating Sleep Mode. The ST parameter is only valid with Cyclic Sleep settings (SM = 4 - 6). Set ST on Cyclic Sleep Coordinator to match Cyclic Sleep Remotes.	0x01 - 0xFFFF [x 1 ms]	0x1388 (5000d)
VR	Diagnostics	<b>Firmware Version.</b> Read firmware version of the RF module.	0 - 0xFFFF [read-only]	Factory-set
WR	(Special)	<b>Write.</b> Write parameter values to RF module's non-volatile memory so that modifications persist through subsequent power-up or reset.	-	-

## 3.2. Command Descriptions

Command descriptions in this section are listed alphabetically. Command categories are designated within "< >" symbols that follow each command title. XBee-PRO RF modules expect parameter values in hexadecimal (designated by the "0x" prefix).

### BD (Interface Data Rate) Command

<Serial Interfacing> The BD command is used to set and read the serial interface data rate (baud rate) used between the RF module and host. This parameter determines the rate at which serial data is sent to the RF module from the host. Modified interface data rates do not take effect until the CN (Exit AT Command Mode) command is issued and the system returns the 'OK' response.

When parameters 0-7 are sent to the RF module, the respective interface data rates are used (as shown in the table on the right).

The RF data rate is not affected by the BD parameter. If the interface data rate is set higher than the RF data rate, a flow control configuration may need to be implemented.

AT Command: ATBD

Parameter Range: 0 – 7 (standard rates)

Parameter	Configuration (bps)
0	1200
1	2400
2	4800
3	9600
4	19200
5	38400
6	57600
7	115200

Default Parameter Value: 3

#### Non-standard Interface Data Rates:

When parameter values outside the range of standard baud rates are sent, the closest interface data rate represented by the number is stored in the BD register. For example, a rate of 19200 bps can be set by sending the following command line "ATBD4B00". NOTE: When using MaxStream's X-CTU Software, non-standard interface data rates can only be set and read using the X-CTU 'Terminal' tab. Non-standard rates are not accessible through the 'Modem Configuration' tab.

When the BD command is sent with a non-standard interface data rate, the UART will adjust to accommodate the requested interface rate. In most cases, the clock resolution will cause the stored BD parameter to vary from the parameter that was sent (refer to the table below). Reading the BD command (send "ATBD" command without an associated parameter value) will return the value that was actually stored to the BD register.

Table 3-02. Parameters Sent Versus Parameters Stored

BD Parameter Sent (HEX)	Interface Data Rate (bps)	BD Parameter Stored (HEX)
0	1200	0
4	19,200	4
7	115,200	7
12C	300	12B
1C200	115,200	1B207

### CC (Command Sequence Character) Command

<AT Command Mode Options> The CC command is used to set and read the ASCII character used between guard times of the AT Command Mode Sequence (GT + CC + GT). This sequence enters the RF module into AT Command Mode so that data entering the modem from the host is recognized as commands instead of payload.

Refer to the Command Mode section [p13] for more information regarding the AT Command Mode Sequence.

AT Command: ATCC

Parameter Range: 0 – 0xFF

Default Parameter Value: 0x2B (ASCII "+")

Related Commands: GT (Guard Times)

**CH (Channel) Command**

<Networking {Addressing}> The CH command is used to set and read the channel on which RF connections are made between RF modules. The channel is one of three filtration layers available to the RF module. The other layers are the PAN ID (ID command) and destination addresses (DL & DH commands).

In order for RF modules to communicate with each other, the RF modules must share the same channel number. Different channels can be used to prevent RF modules in one network from listening to transmissions of another.

*The RF module uses channel numbers of the 802.15.4 standard.*

$$\text{Center Frequency} = 2.405 + (\text{CH} - 11d) * 5 \text{ MHz} \quad (d = \text{decimal})$$

Refer to the "Addressing" section [p9] for more information.

AT Command: ATCH

Parameter Range: 0x0B – 0x1A (XBee)  
0x0C – 0x18 (XBee-PRO)

Default Parameter Value: 0x0C (12 decimal)

Related Commands: ID (PAN ID), DL (Destination Address Low, DH (Destination Address High)

**CN (Exit AT Command Mode) Command**

<AT Command Mode Options> The CN command is used to explicitly exit the RF module from AT Command Mode.

AT Command: ATCN

**CT (Command Mode Timeout) Command**

<AT Command Mode Options> The CT command is used to set and read the amount of inactive time that elapses before the RF module automatically exits from AT Command Mode and returns to Idle Mode.

Use the CN (Exit AT Command Mode) command to exit AT Command Mode manually.

AT Command: ATCT

Parameter Range: 2 – 0xFFFF  
[x 100 milliseconds]

Default Parameter Value: 0x64 (100 decimal, which equals 10 decimal seconds)

Number of bytes returned: 2

Related Command: CN (Exit AT Command Mode)

**DB (Received Signal Strength) Command**

<Diagnostics> DB parameter is used to read the received signal strength (in dBm) of the last RF packet received. Reported values are accurate between -40 dBm and the RF module's receiver sensitivity.

Absolute values are reported. For example: 0x58 = -88 dBm (decimal). If no packets have been received (since last reset, power cycle or sleep event), "0" will be reported.

AT Command: ATDB

Parameter Range: 0 – 0x64 [read-only]

**DH (Destination Address High) Command**

<Networking {Addressing}> The DH command is used to set and read the upper 32 bits of the RF module's 64-bit destination address. When combined with the DL (Destination Address Low) parameter, it defines the destination address used for transmission.

An RF module will only communicate with other RF modules having the same channel (CH parameter), PAN ID (ID parameter) and destination address (DH + DL parameters).

To transmit using a 16-bit address, set the DH parameter to zero and the DL parameter less than 0xFFFF. 0x000000000000FFFF (DL concatenated to DH) is the broadcast address for the PAN.

Refer to the "Addressing" section [p9] for more information.

AT Command: ATDH

Parameter Range: 0 – 0xFFFFFFFF

Default Parameter Value: 0

Related Commands: DL (Destination Address Low), CH (Channel), ID (PAN VID), MY (Source Address)



**DL (Destination Address Low) Command**

<Networking {Addressing}> The DL command is used to set and read the lower 32 bits of the RF module's 64-bit destination address. When combined with the DH (Destination Address High) parameter, it defines the destination address used for transmission.

An RF module will only communicate with other RF modules having the same channel (CH parameter), PAN ID (ID parameter) and destination address (DH + DL parameters).

To transmit using a 16-bit address, set the DH parameter to zero and the DL parameter less than 0xFFFF. 0x00000000000000FFFF (DL concatenated to DH) is the broadcast address for the PAN.

Refer to the "Addressing" section [p9] for more information.

AT Command: ATDL

Parameter Range: 0 - 0xFFFFFFFF

Default Parameter Value: 0

Related Commands: DH (Destination Address High), CH (Channel), ID (PAN VID), MY (Source Address)

**GT (Guard Times) Command**

<AT Command Mode Options> GT Command is used to set the DI (data in from host) time-of-silence that surrounds the AT command sequence character (CC Command) of the AT Command Mode sequence (GT + CC + GT).

The DI time-of-silence is used to prevent inadvertent entrance into AT Command Mode.

Refer to the Command Mode section [p13] for more information regarding the AT Command Mode Sequence.

AT Command: ATGT

Parameter Range: 2 - 0xFFFF  
[x 1 millisecond]

Default Parameter Value: 0x3E8  
(1000 decimal)

Related Command: CC (Command Sequence Character)

**ID (Pan ID) Command**

<Networking {Addressing}> The ID command is used to set and read the PAN (Personal Area Network) ID of the RF module. Only RF modules with matching PAN IDs can communicate with each other. RF modems with non-matching PAN IDs will not receive unintended data transmission.

Setting the ID parameter to 0xFFFF indicates a global message for all PANs.

Refer to the "Addressing" section [p9] for more information.

AT Command: ATID

Parameter Range: 0 - 0xFFFF

Default Parameter Value: 0x3332  
(13106 decimal)

**MY (16-bit Source Address) Command**

<Networking {Addressing}> The MY command is used to set and read the 16-bit source address of the RF module.

By setting MY to 0xFFFF, the reception of RF packets having a 16-bit address is disabled. The 64-bit address is the module serial number and is always enabled.

Refer to the "Addressing" section [p9] for more information.

AT Command: ATMY

Parameter Range: 0 - 0xFFFF

Default Parameter Value: 0

Related Commands: DH (Destination Address High), DL (Destination Address Low), CH (Channel), ID (PAN ID)

**P0 (PWM0 Configuration) Command**

<Diagnostics> The P0 command is used to select and read the function for PWM0 (Pulse Width Modulation output 0 - pin 6).

Note: The second character in the command is a zero ("0"), not the letter "O".

AT Command: ATPO

Parameter Range: 0 – 1

Parameter	Configuration
0	Disabled
1	RSSI PWM0 enabled

Default Parameter Value: 1

**PL (Power Level) Command**

<RF Interfacing> The PL command is used to select and read the power level at which the RF module transmits conducted power.

AT Command: ATPL

Parameter Range: 0 – 4

Parameter	XBee	XBee-Pro
0	-10 dBm	10 dBm
1	-6 dBm	12 dBm
2	-4 dBm	14 dBm
3	-2 dBm	16 dBm
4	0 dBm	18 dBm

Default Parameter Value: 4

**RE (Restore Defaults) Command**

<(Special)> The RE command is used to restore all configurable parameters to their factory default settings. The RE command does not write restored values to non-volatile (persistent) memory. Issue the WR (Write) command subsequent to issuing the RE command to save restored parameter values to non-volatile memory.

AT Command: ATRE

**RN (Random Delay Slots) Command**

<Networking & Security> The RN command is used to set and read the minimum value of the back-off exponent in the CSMA-CA algorithm. The CSMA-CA algorithm was engineered for collision avoidance (random delays are inserted to prevent data loss caused by data collisions).

AT Command: ATRN

Parameter Range: 0 – 3 [exponent]

Default Parameter Value: 0

If RN = 0, collision avoidance is disabled during the first iteration of the algorithm (802.15.4 - macMinBE).

CSMA-CA stands for "Carrier Sense Multiple Access - Collision Avoidance". Unlike CSMA-CD (reacts to network transmissions after collisions have been detected), CSMA-CA acts to prevent data collisions before they occur. As soon as a modem receives a packet that is to be transmitted, it checks if the channel is clear (no other modem is transmitting). If the channel is clear, the packet is sent over-the-air. If the channel is not clear, the RF module waits for a randomly selected period of time, then checks again to see if the channel is clear. After a time, the process ends and the data is lost.

**RO (Packetization Timeout) Command**

<Serial Interfacing> RO command is used to set and read the number of character times of inter-character delay required before transmission.

RF transmission commences when data is detected in the DI (data in from host) buffer and RO character times of silence are detected on the UART receive lines (after receiving at least 1 byte).

RF transmission will also commence after 100 bytes (maximum packet size) are received in the DI buffer.

Set the RO parameter to '0' to transmit characters as they arrive instead of buffering them into one RF packet.

AT Command: ATRO

Parameter Range: 0 – 0xFF  
[x character times]

Default Parameter Value: 3

**RP (RSSI PWM Timer) Command**

<Diagnostics> The RP command is used to enable PWM (Pulse Width Modulation) output on the RF module. The output is calibrated to show the level a received RF signal is above the sensitivity level of the RF module. The PWM pulses vary from zero to 95 percent. Zero to twenty-nine percent means the received RF signal is at or below the published sensitivity level of the RF module. The following table shows levels above sensitivity and PWM values.

The total period of the PWM output is 8.32 ms. Because there are 40 steps in the PWM output, the minimum step size is 0.208 ms.

**Table 3-03. PWM Percentages**

dB above Sensitivity	PWM percentage* (high period / total period)
10	46.0%
20	63.0%
30	80.1%

\* PWM% = (295 + (17.5 \* dBm above sensitivity)) / 10.24

A non-zero value defines the time that the PWM output will be active with the RSSI value of the last received RF packet. After the set time when no RF packets are received, the PWM output will be set low (0 percent PWM) until another RF packet is received. The PWM output will also be set low at power-up until the first RF packet is received. A parameter value of 0xFF permanently enables the PWM output and it will always reflect the value of the last received RF packet.

AT Command: ATRP

Parameter Range: 0 – 0xFF  
[x 100 milliseconds]

Default Parameter Value: 0x28 (40 decimal)

**SH (Serial Number High) Command**

<Diagnostics> The SH command is used to read the high 32 bits of the RF module's unique IEEE 64-bit address.

The RF module serial number is set at the factory and is read-only.

AT Command: ATSH

Parameter Range: 0 – 0xFFFFFFFF [read-only]

Related Commands: SL (Serial Number Low), MY (Source Address)

**SL (Serial Number Low) Command**

<Diagnostics> The SL command is used to read the low 32 bits of the RF module's unique IEEE 64-bit address.

The RF module serial number is set at the factory and is read-only.

AT Command: ATSL

Parameter Range: 0 – 0xFFFFFFFF [read-only]

Related Commands: SH (Serial Number High), MY (Source Address)

### SM (Sleep Mode) Command

<Sleep Mode (Low Power)> The SM command is used to set and read Sleep Mode settings. By default, Sleep Modes are disabled (SM = 0) and the RF module remains in Idle/Receive Mode. When in this state, the RF module is constantly ready to respond to either serial or RF activity. SM command options vary according to the networking system type. By default, the module is configured to operate in a NonBeacon system.

AT Command: ATSM

Parameter Range: 0 – 5

Parameter	Configuration
0	Disabled
1	Pin Hibernate
2	Pin Doze
3	(reserved)
4	Cyclic Sleep Remote
5	Cyclic Sleep Remote (with Pin Wake-up)

Default Parameter Value: 0

Related Commands: SP (Cyclic Sleep Period), ST (Time before Sleep)

### SP (Cyclic Sleep Period) Command

<Sleep Mode (Low Power)> The SP command is used to set and read the duration of time in which a remote RF module sleeps. After the cyclic sleep period is over, the RF module wakes and checks for data. If data is not present, the RF module goes back to sleep. The maximum sleep period is 268 seconds (SP = 0x68B0).

AT Command: ATSP

Parameter Range: 1 – 0x68B0  
[x 10 milliseconds]

Default Parameter Value: 0x64 (100d)

Related Commands: SM (Sleep Mode), ST (Time before Sleep)

The SP parameter is only valid if the RF module is configured to operate in Cyclic Sleep (SM = 4-6).

### ST (Time before Sleep) Command

<Sleep Mode (Low Power)> The ST command is used to set and read the period of time that the RF module remains inactive (no transmitting or receiving) before entering into Sleep Mode.

AT Command: ATST

Parameter Range: 1 – 0xFFFF  
[x 1 millisecond]

Default Parameter Value: 0x1388  
(5000 decimal)

Related Commands: SM (Sleep Mode), SP (Cyclic Sleep Period)

For example, if the ST parameter is set to its default value of 0x1388 (5000 decimal), the RF module will enter into Sleep mode after 5 seconds of inactivity. This command can only be used if Cyclic Sleep settings have been selected using SM (Sleep Mode) Command (SM = 4-6).

NOTE: The GT parameter value must always be less than the ST value. (If GT > ST, the configuration will render the module unable to enter into command mode.) If the ST parameter is modified, also modify the GT parameter accordingly.

### **VR (Firmware Version) Command**

---

<Diagnostics> The VR command is used to read which firmware version is stored in the RF module.

---

AT Command: ATVR

---

Parameter Range: 0 – 0xFFFF [read only]

---

### **WR (Write) Command**

---

<(Special)> The WR command is used to write configurable parameters to the RF module's non-volatile memory (Parameter values remain in RF module's memory until overwritten by subsequent use of the WR Command).

---

AT Command: ATWR

---

If changes are made without writing them to non-volatile memory, the RF module reverts back to previously saved parameters the next time the RF module is powered-on.

---

**NOTE:** Once the WR command is sent to the RF module, no additional characters should be sent until after the "OK/r" response is received.

---

# Appendix A: Agency Certifications

## FCC Certification

The XBee/XBee-PRO RF Module complies with Part 15 of the FCC rules and regulations. Compliance with the labeling requirements, FCC notices and antenna usage guidelines is required.

To fulfill FCC Certification requirements, the OEM must comply with the following regulations:

1. The system integrator must ensure that the text on the external label provided with this device is placed on the outside of the final product [Figure A-01].
2. The XBee/XBee-PRO RF Module may be used only with approved antennas that have been tested with this modem.

## OEM Labeling Requirements



**WARNING:** The Original Equipment Manufacturer (OEM) must ensure that FCC labeling requirements are met. This includes a clearly visible label on the outside of the final product enclosure that displays the contents shown in the figure below.

**Figure A-01. Required FCC Label for OEM products containing the XBee/XBee-PRO RF Module**

Contains FCC ID: OUR-XBEE\*

The enclosed device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions: (1) this device may not cause harmful interference and (2) this device must accept any interference received, including interference that may cause undesired operation.

\* The FCC ID for the XBee is "OUR-XBEE". The FCC certification for the XBee-PRO is pending.

## FCC Notices

**IMPORTANT:** The XBee/XBee-PRO OEM RF Module has been certified by the FCC for use with other products without any further certification (as per FCC section 2.1091). Modifications not expressly approved by MaxStream could void the user's authority to operate the equipment.

**IMPORTANT:** OEMs must test final product to comply with unintentional radiators (FCC section 15.107 & 15.109) before declaring compliance of their final product to Part 15 of the FCC Rules.

**IMPORTANT:** The RF module has been certified for remote and base radio applications. If the module will be used for portable applications, the device must undergo SAR testing.

This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation.

If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures: Re-orient or relocate the receiving antenna, Increase the separation between the equipment and receiver, Connect equipment and receiver to outlets on different circuits, or Consult the dealer or an experienced radio/TV technician for help.

## FCC-Approved Antennas (2.4 GHz)

The XBee/XBee-Pro OEM RF Module can be installed utilizing antennas and cables constructed with standard connectors (Type-N, SMA, TNC, etc.) if the installation is performed professionally and according to FCC guidelines. For installations not performed by a professional, non-standard connectors (RPSMA, RPTNC, etc.) must be used.

The modules are pre-FCC approved for use in fixed base station and mobile applications [refer to table below]. As long as the antenna is mounted at least 20 cm (8 in) from nearby persons, the application is considered a mobile application. Antennas not listed in the table must be tested to comply with FCC Section 15.203 (unique antenna connectors) and Section 15.247 (emissions).

**Table A-01. Antennas approved for use with the XBee/XBee-PRO OEM RF Modules (all 2.4 GHz)**

Part Number	Type (Description)	Gain	Application	Min. Separation
A24-HABMM-PSI	Dipole (Half-wave bulkhead mount articulated MMCX w/ pigtail)	2.1 dBi	Fixed/Mobile*	20 cm
A24-HBMM-PSI	Dipole (Half-wave bulkhead mount MMCX w/ pigtail)	2.1 dBi	Fixed/Mobile*	20 cm
A24-HABSM	Dipole (Articulated RPSMA)	2.1 dBi	Fixed/Mobile*	20 cm
A24-QBMM-PSI	Monopole (Quarter-wave bulkhead mount MMCX w/pigtail)	1.9 dBi	Fixed/Mobile*	20 cm
A24-QABMM-PSI	Monopole (Quarter-wave bulkhead mount articulated MMCX w/pigtail)	1.9 dBi	Fixed/Mobile*	20 cm
A24-QI	Monopole (Integrated whip)	1.9 dBi	Fixed/Mobile*	20 cm
A24-C1	Surface Mount	-1.5 dBi	Fixed/Mobile*	20 cm
A24-Y4NF	Yagi (4-element)	6.0 dBi	Fixed*	2 m
A24-Y6NF	Yagi (6-element)	8.8 dBi	Fixed*	2 m
A24-Y7NF	Yagi (7-element)	9.0 dBi	Fixed*	2 m
A24-Y9NF	Yagi (9-element)	10.0 dBi	Fixed*	2 m
A24-Y10NF	Yagi (10-element)	11.0 dBi	Fixed*	2 m
A24-Y12NF	Yagi (12-element)	12.0 dBi	Fixed*	2 m
A24-Y13NF	Yagi (13-element)	12.0 dBi	Fixed*	2 m
A24-Y15NF	Yagi (15-element)	12.5 dBi	Fixed*	2 m
A24-Y16NF	Yagi (16-element)	13.5 dBi	Fixed*	2 m
A24-Y16RM	Yagi (16-element, RPSMA connector)	13.5 dBi	Fixed*	2 m
A24-Y18NF	Yagi (18-element)	15.0 dBi	Fixed*	2 m
A24-F2NF	Omni-directional (Fiberglass base station)	2.1 dBi	Fixed/Mobile*	20 cm
A24-F3NF	Omni-directional (Fiberglass base station)	3.0 dBi	Fixed/Mobile*	20 cm
A24-F5NF	Omni-directional (Fiberglass base station)	5.0 dBi	Fixed/Mobile*	20 cm
A24-F8NF	Omni-directional (Fiberglass base station)	8.0 dBi	Fixed*	2 m
A24-F9NF	Omni-directional (Fiberglass base station)	9.5 dBi	Fixed*	2 m
A24-F10NF	Omni-directional (Fiberglass base station)	10.0 dBi	Fixed*	2 m
A24-F12NF	Omni-directional (Fiberglass base station)	12.0 dBi	Fixed*	2 m
A24-F15NF	Omni-directional (Fiberglass base station)	15.0 dBi	Fixed*	2 m
A24-W7NF	Omni-directional (Base station)	7.2 dBi	Fixed*	2 m
A24-M7NF	Omni-directional (Mag-mount base station)	7.2 dBi	Fixed*	2 m
A24-P8SF	Flat Panel	8.5 dBi	Fixed*	2 m
A24-P8NF	Flat Panel	8.5 dBi	Fixed*	2 m
A24-P13NF	Flat Panel	13.0 dBi	Fixed*	2 m
A24-P14NF	Flat Panel	14.0 dBi	Fixed*	2 m
A24-P15NF	Flat Panel	15.0 dBi	Fixed*	2 m
A24-P16NF	Flat Panel	16.0 dBi	Fixed*	2 m
A24-P19NF	Flat Panel	19.0 dBi	Fixed*	2 m

\* Antennas can be approved for portable applications if integrator gains approval through SAR testing. If the antenna will be mounted closer than 20 cm to nearby persons, then the application is considered "portable" and requires an additional test performed on the final product. This test is called the Specific Absorption Rate (SAR) testing and measures the emissions from the module and how they affect the person.

### RF Exposure



**WARNING:** To satisfy FCC RF exposure requirements for mobile transmitting devices, a separation distance of 20 cm or more should be maintained between the antenna of this device and persons during device operation. To ensure compliance, operations at closer than this distance is not recommended. The antenna used for this transmitter must not be co-located in conjunction with any other antenna or transmitter.

The preceding statement must be included as a CAUTION statement in manuals for OEM products to alert users on FCC RF Exposure compliance.

## European Certification (pending)

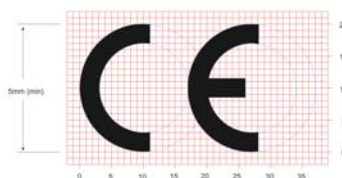
The XBee/XBee-PRO RF Module has been certified for use in several European countries. For a complete list, refer to [www.maxstream.net](http://www.maxstream.net).

If the XBee/XBee-PRO RF Modules are incorporated into a product, the manufacturer must ensure compliance of the final product to the European harmonized EMC and low-voltage/safety standards. A Declaration of Conformity must be issued for each of these standards and kept on file as described in Annex II of the R&TTE Directive. Furthermore, the manufacturer must maintain a copy of the XBee/XBee-PRO user manual documentation and ensure the final product does not exceed the specified power ratings, antenna specifications, and/or installation requirements as specified in the user manual. If any of these specifications are exceeded in the final product, a submission must be made to a notified body for compliance testing to all required standards.

### OEM Labeling Requirements

The 'CE' marking must be affixed to a visible location on the OEM product.

Figure A-02. CE Labeling Requirements



The CE mark shall consist of the initials "CE" taking the following form:

- If the CE marking is reduced or enlarged, the proportions given in the above graduated drawing must be respected.
- The CE marking must have a height of at least 5mm except where this is not possible on account of the nature of the apparatus.
- The CE marking must be affixed visibly, legibly, and indelibly.

### Restrictions

France - France imposes restrictions on the 2.4 GHz band. Go to [www.art-telecom.fr](http://www.art-telecom.fr) or contact MaxStream for more information.

Norway - Norway prohibits operation near Ny-Alesund in Svalbard. More information can be found at the Norway Posts and Telecommunications site ([www.npt.no](http://www.npt.no)).

### Declarations of Conformity

MaxStream has issued Declarations of Conformity for the XBee/XBee-PRO RF Modules concerning emissions, EMC and safety. Files are located in the 'documentation' folder of the MaxStream CD.

#### Important Note

MaxStream does not list the entire set of standards that must be met for each country. MaxStream customers assume full responsibility for learning and meeting the required guidelines for each country in their distribution market. For more information relating to European compliance of an OEM product incorporating the XBee/XBee-PRO RF Module, contact MaxStream, or refer to the following web sites:

CEPT ERC 70-03E - Technical Requirements, European restrictions and general requirements: Available at [www.ero.dk/](http://www.ero.dk/).

R&TTE Directive - Equipment requirements, placement on market: Available at [www.ero.dk/](http://www.ero.dk/).



# Appendix B: Development Guide

## Development Kit Contents

The XBee Development Kit includes the hardware and software needed to rapidly create long range wireless links between devices.

**Table B-01. Items Included in the Development Kit**

Item	Qty.	Description	Part #
XBee-PRO Module	2	(1) OEM RF Module w/ U.FL antenna connector (1) OEM RF Module w/ attached wire antenna	XB24-...UI-... XB24-...WI-...
XBee Module	3	(1) OEM RF Module w/ U.FL antenna connector (1) OEM RF Module w/ attached wire antenna (1) OEM RF Module w/ chip antenna	XB24-...UI-... XB24-...WI-... XB24-...CI-...
RS-232 Interface Board	1	Board for interfacing between modules and RS-232 devices (Converts signal levels, displays diagnostic info, & more)	XBIB-R
USB Interface Board	1	Board for interfacing between modules & USB devices (Converts signal levels, displays diagnostic info, & more)	XBIB-U
RS-232 Cable (6', straight-through)	1	Cable for connecting RS-232 interface board with DTE devices (devices that have a male serial DB-9 port - such as most PCs)	JD2D3-CDS-6F
USB Cable (6')	1	Cable for connecting USB interface board to USB devices	JU1U2-CSB-6F
Serial Loopback Adapter	1	[Red] Adapter for configuring the module assembly (module + RS-232 interface board) to function as a repeater for range testing	JD2D3-CDL-A
NULL Modem Adapter (male-to-male)	1	[Black] Adapter for connecting the module assembly (module + RS-232 interface board) to other DCE (female DB-9) devices	JD2D2-CDN-A
NULL Modem Adapter (female-to-female)	1	[Gray] Adapter for connecting serial devices. It allows users to bypass the radios to verify serial cabling is functioning properly.	JD3D3-CDN-A
9VDC Power Adapter	1	Adapter for powering the RS-232 interface board	JP5P2-9V11-6F
9V Battery Clip	1	Clip for remotely powering the RS-232 board w/ a 9V battery	JP2P3-C2C-4I
RPSMA Antenna	1	RPSMA half-wave dipole antenna (2.4 GHz, 2.1 dB)	A24-HASM-525
RF Cable Assembly	1	Adapter for connecting RPSMA antenna to U.FL connector	JF1R6-CR3-4I
CD	1	Documentation and Software	MD0010
Quick Start Guide	1	Step-by-step instruction on how to create wireless links & test range capabilities of the modules	MD0026

## Interfacing Options

The development kit includes an RS-232 and a USB interface board. Both boards provide a direct connection to many serial devices and therefore provide access to the RF module registries. Parameters stored in the registry allow OEMs and integrators to customize the modules to suite the needs of their data radio systems.

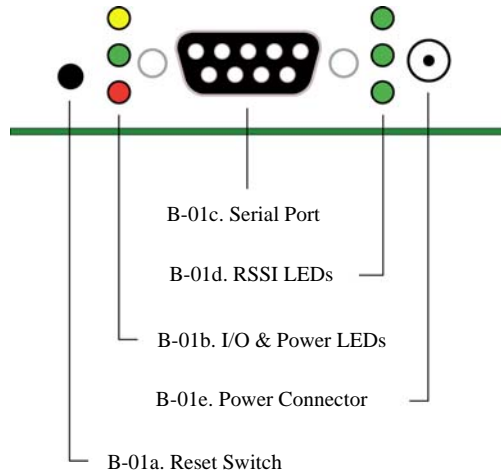
The following sections illustrate how to use the interface boards for development purposes. The MaxStream Interface board provides means for connecting the module to any node that has an available RS-232 or USB connector. Since the module requires signals to enter at TTL voltages, one of the main functions of the interface board is to convert signals between TTL levels and RS-232 and USB levels.

**Note:** In the following sections, an OEM RF Module mounted to an interface board will be referred to as a "Module Assembly".

## RS-232 Interface Board

### Physical Interface

Figure B-01. Front View



#### B-01a. Reset Switch

The Reset Switch is used to reset (re-boot) the RF module. This switch only applies when using the configuration tabs of MaxStream's X-CTU Software.

#### B-01b. I/O & Power LEDs

LEDs indicate RF module activity as follows:

- Yellow (top LED) = Serial Data Out (to host)
- Green (middle) = Serial Data In (from host)
- Red (bottom) = Power/TX Indicator (LED is on when module assembly is powered)



#### B-01c. Serial Port

Standard female DB-9 (RS-232) connector.

#### B-01d. RSSI LEDs

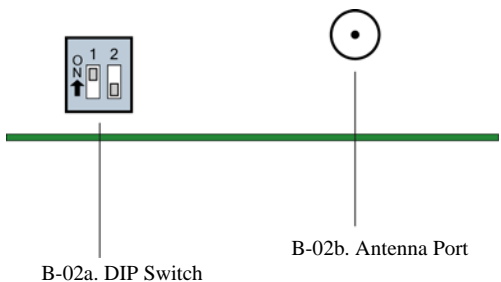
RSSI LEDs indicate the amount of fade margin present in an active wireless link. Fade margin is defined as the difference between the incoming signal strength and the modem's receiver sensitivity.

- 3 LEDs ON = Very Strong Signal (> 30 dB fade margin)
- 2 LEDs ON = Strong Signal (> 20 dB fade margin)
- 1 LED ON = Moderate Signal (> 10 dB fade margin)
- 0 LED ON = Weak Signal (< 10 dB fade margin)

#### B-01e. Power Connector

5-14 VDC power connector

Figure B-02. Back View



#### B-02a. DIP Switch

DIP Switch functions are not supported in this release. Future downloadable firmware versions will support DIP Switch configurations.

#### B-02b. Antenna Port

Port is a 50Ω RF signal connector for connecting to an external antenna. The connector type is RPSMA (Reverse Polarity SMA) female. The connector has threads on the outside of a barrel and a male center conductor.

## RS-232 Pin Signals

Figure B-03. Pins used on the female RS-232 (DB-9) Serial Connector

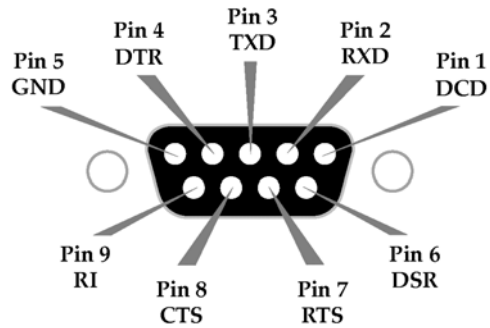


Table B-02. Pin Assignments and Implementations

DB-9 Pin	RS-232 Name	Description	Implementation*
1	DCD	Data-Carrier-Detect	Connected to DSR (pin6)
2	RXD	Received Data	Serial data exiting the module assembly (to host)
3	TXD	Transmitted Data	Serial data entering into the module assembly (from host)
4	DTR	Data-Terminal-Ready	Can enable Power-Down on the module assembly
5	GND	Ground Signal	Ground
6	DSR	Data-Set-Ready	Connected to DCD (pin1)
7	$\overline{\text{RTS}}$ / CMD	Request-to-Send / Command Mode	Provides $\overline{\text{RTS}}$ flow control or enables Command Mode
8	$\overline{\text{CTS}}$	Clear-to-Send	Provides $\overline{\text{CTS}}$ flow control
9	RI	Ring Indicator	Optional power input that is connected internally to the positive lead of the front power connector

\* Functions listed in the implementation column may not be available at the time of release.

## Wiring Diagrams

Figure B-04. DTE Device (RS-232, male DB-9 connector) wired to a DCE Module Assembly (female DB-9)

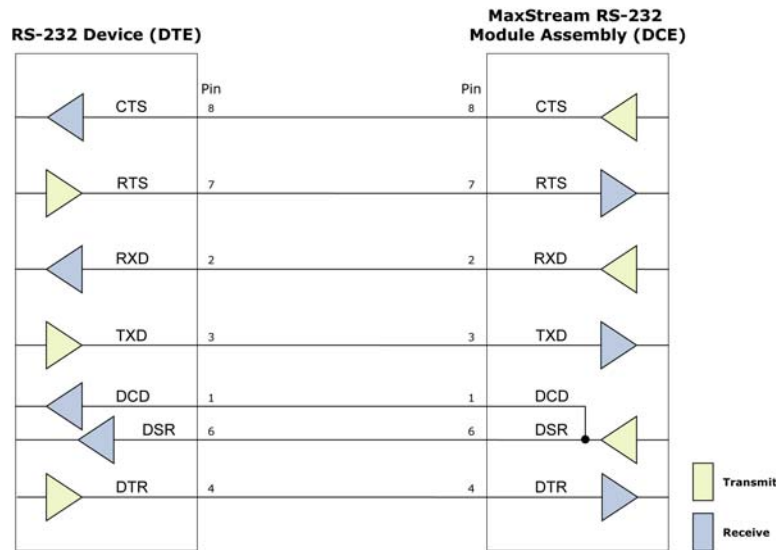
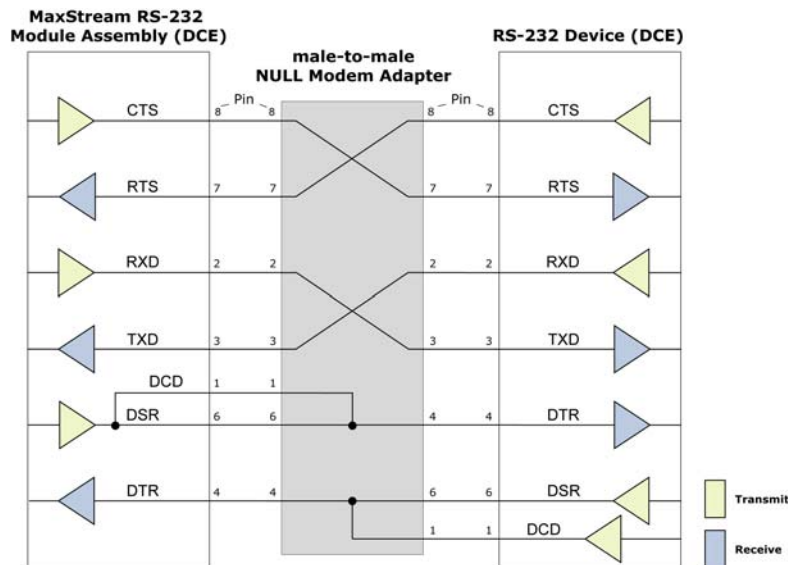
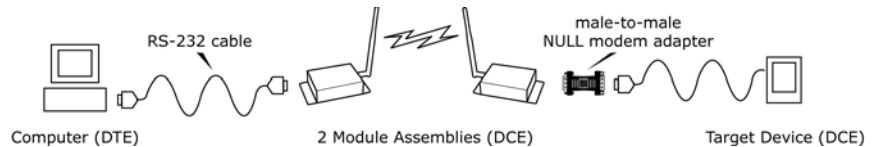


Figure B-05. DCE Module Assembly (female DB-9 connector) wired to a DCE Device (RS-232, male DB-9)



### Sample Wireless Connection: DTE <--> DCE <--> DCE <--> DCE

Figure B-06. Typical wireless link between DTE and DCE devices



## Adapters

The development kit includes several adapters that support the following functions:

- Performing Range Tests
- Testing Cables
- Connecting to other RS-232 DCE and DTE devices
- Connecting to terminal blocks or RJ-45 (for RS-485/422 devices)

### NULL Modem Adapter (male-to-male)

**Part Number: JD2D2-CDN-A (Black, DB-9 M-M)** The male-to-male NULL modem adapter is used to connect two DCE devices. A DCE device connects with a straight-through cable to the male serial port of a computer (DTE).

Figure B-07. Male NULL modem adapter and pinouts

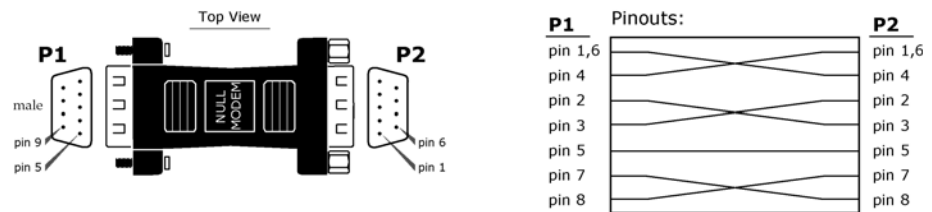
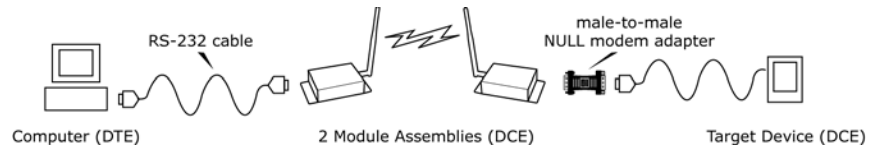


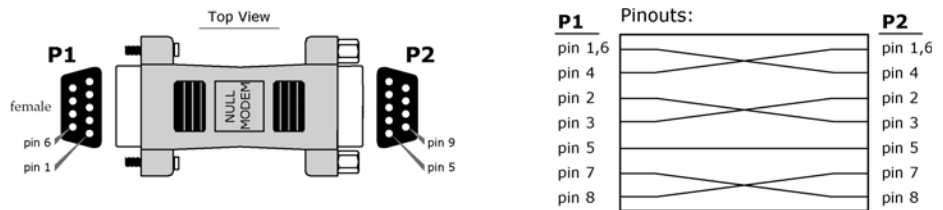
Figure B-08. Example of a MaxStream Radio Modem (DCE Device) connecting to another DCE device



### NULL Modem Adapter (female-to-female)

**Part Number: JD3D3-CDN-A (Gray, DB-9 F-F)** The female-to-female NULL modem adapter is used to verify serial cabling is functioning properly. To test cables, insert the female-to-female NULL modem adapter in place of a pair of module assemblies (RS-232 interface board + XTend Module) and test the connection without radio modules in the connection.

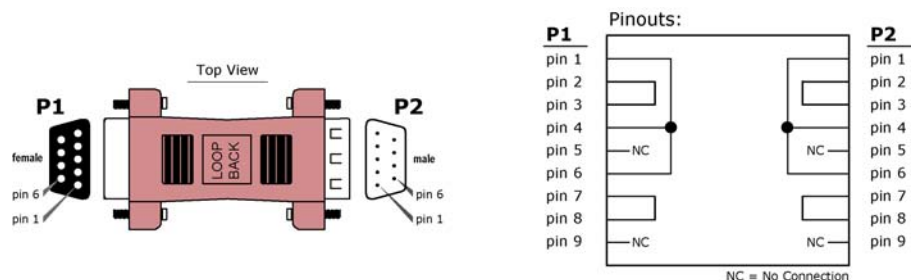
Figure B-09. Female NULL modem adapter and pinouts



### Serial Loopback Adapter

**Part Number: JD2D3-CDL-A (Red, DB-9 M-F)** The serial loopback adapter is used for range testing. During a range test, the serial loopback adapter configures the module to function as a repeater by looping serial data back into the radio for retransmission.

Figure B-10. Serial loopback adapter and pinouts

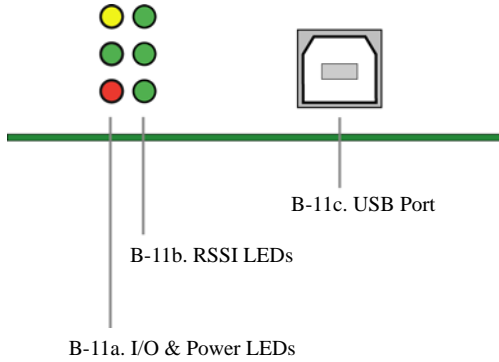


## USB Interface Board

### Physical Interface

#### B-11a. I/O & Power LEDs

Figure B-11. Front View



LEDs indicate RF module activity as follows:

- Yellow (top LED) = Serial Data Out (to host)
- Green (middle) = Serial Data In (from host)
- Red (bottom) = Power/TX Indicator (Red LED is illuminated when RF module is powered)



#### B-11b. RSSI LEDs

RSSI LEDs indicate the amount of fade margin present in an active wireless link. Fade margin is defined as the difference between the incoming signal strength and the module's receiver sensitivity.

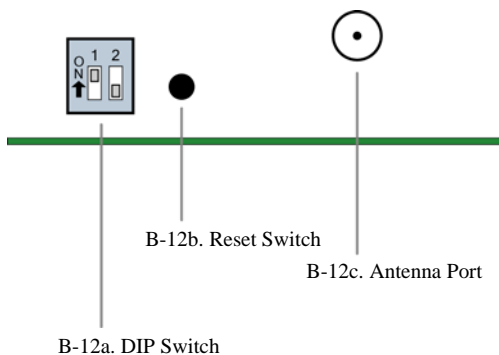
- 3 LEDs ON = Very Strong Signal (> 30 dB fade margin)
- 2 LEDs ON = Strong Signal (> 20 dB fade margin)
- 1 LED ON = Moderate Signal (> 10 dB fade margin)
- 0 LED ON = Weak Signal (< 10 dB fade margin)

#### B-11c. USB Port

Standard Type-B OEM connector is used to communicate with OEM host and power the RF module.

#### B-12a. DIP Switch

Figure B-12. Back View



DIP Switch functions are not supported in this release. Future downloadable firmware versions will support the DIP Switch configurations.

#### B-12b Reset Switch

The Reset Switch is used to reset (re-boot) the RF module.

#### B-12c. Antenna Port

Port is a 50Ω RF signal connector for connecting to an external antenna. The connector type is RPSMA (Reverse Polarity SMA) female. The connector has threads on the outside of a barrel and a male center conductor.

### USB Pin Signals

Table B-03. USB signals and their implantations on the XBee/XBee-PRO RF Module

Pin	Name	Description	Implementation
1	VBUS	Power	Power the RF module
2	D-	Transmitted & Received Data	Transmit data to and from the RF module
3	D+	Transmitted & Received Data	Transmit data to and from the RF module
4	GND	Ground Signal	Ground

# Appendix C: Additional Information

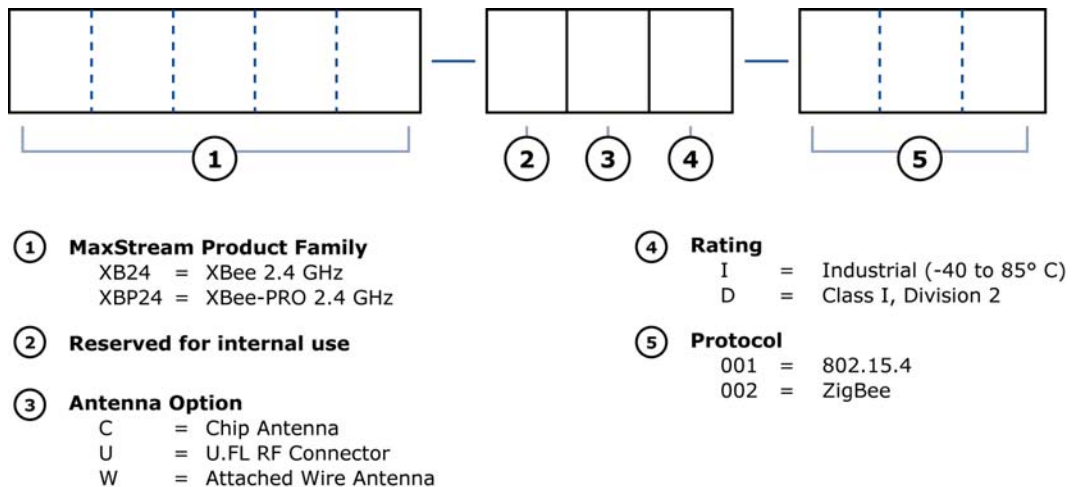
## 1-Year Warranty

XBee/XBee-PRO RF Modules from MaxStream, Inc. (the "Product") are warranted against defects in materials and workmanship under normal use, for a period of 1-year from the date of purchase. In the event of a product failure due to materials or workmanship, MaxStream will repair or replace the defective product. For warranty service, return the defective product to MaxStream, shipping prepaid, for prompt repair or replacement.

The foregoing sets forth the full extent of MaxStream's warranties regarding the Product. Repair or replacement at MaxStream's option is the exclusive remedy. THIS WARRANTY IS GIVEN IN LIEU OF ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, AND MAXSTREAM SPECIFICALLY DISCLAIMS ALL WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL MAXSTREAM, ITS SUPPLIERS OR LICENSORS BE LIABLE FOR DAMAGES IN EXCESS OF THE PURCHASE PRICE OF THE PRODUCT, FOR ANY LOSS OF USE, LOSS OF TIME, INCONVENIENCE, COMMERCIAL LOSS, LOST PROFITS OR SAVINGS, OR OTHER INCIDENTAL, SPECIAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PRODUCT, TO THE FULL EXTENT SUCH MAY BE DISCLAIMED BY LAW. SOME STATES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES. THEREFORE, THE FOREGOING EXCLUSIONS MAY NOT APPLY IN ALL CASES. This warranty provides specific legal rights. Other rights which vary from state to state may also apply.

## Ordering Information

Figure C-01. Divisions of the XBee/XBee-PRO RF Module Part Numbers



For example:

XBP24-AWI-001 = XBee-PRO OEM RF Module, 2.4 GHz, attached wire antenna, Industrial temperature rating, IEEE 802.15.4 standard

## Contact MaxStream

---

Free and unlimited technical support is included with every MaxStream Radio Modem sold.

For the best in wireless data solutions and support, please use the following resources:

Documentation: [www.maxstream.net/helpdesk/download.php](http://www.maxstream.net/helpdesk/download.php)

Technical Support: Phone. (866) 765-9885 toll-free U.S.A. & Canada  
(801) 765-9885 Worldwide

Live Chat. [www.maxstream.net](http://www.maxstream.net)

E-Mail. [rf-xperts@maxstream.net](mailto:rf-xperts@maxstream.net)

MaxStream office hours are 8:00 am - 5:00 pm [U.S. Mountain Standard Time]