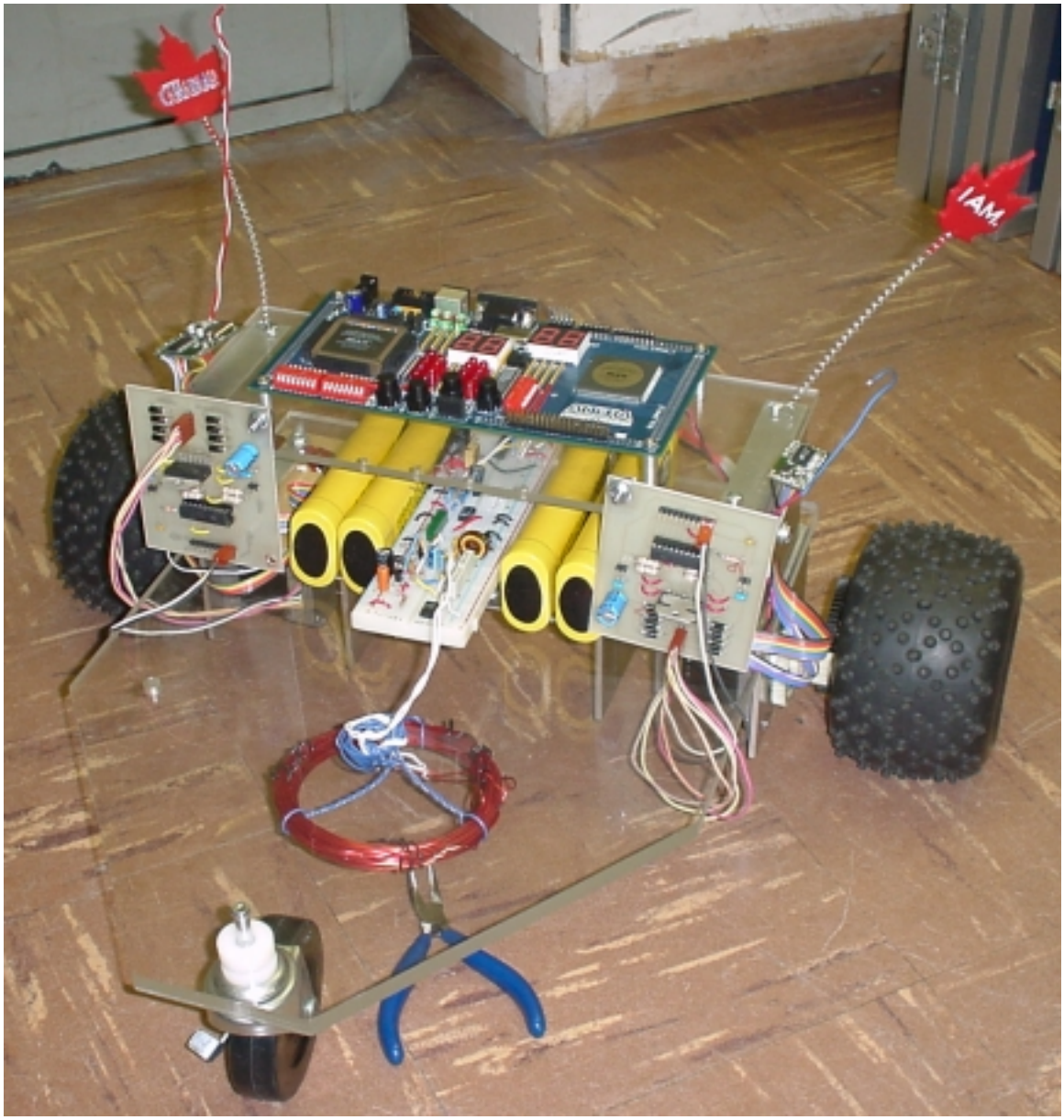
Metal Rover

Final Report

Michael Cumming
mcumming@ee.ualberta.ca

Andrew Leung
anleung@ee.ualberta.ca

November 27, 2000



Introduction

The idea of finding lost treasure left by our pillaging ancestors has crossed all of our minds at one time or another. Getting rich quick with as little effort as possible is something we all desire. Fortunately our ancestors were pretty skilled at accidentally discarding valuables or just burying things for later retreaval. Luckily for us, the treasure maybe still there while they are not.

A popular tool to hunt for lost treasure from the past is to use a simple metal detector. The electronics required are quite simple and effective. The main problem is this metal detector cannot tell scrap metal from gold very easily. Hence, many man-hours can be spent looking for treasure but only to have a few pieces of scrap metal to show for it. This project will help one achieve that dream of finding lost treasure but without all the hours needed to manually search for it.

Problem description

The problem we seek to solve is to overcome the tedious manual searching of treasure with a metal detector. The most effective way to find this treasure is to find a map, which our ancestors usually kept to use to return to the treasure. This project is not about that. The other possibility of finding the treasure is to be using a metal detector and finding it manually. Now, the most taxing part of using this method is that you have to manually search through every square inch yourself if you were to buy a metal detector unit at the local electronics store. Too much work for only a mere chance at unbelievable riches. The local lottery probably has better odds. The problem we have is: how do we search through every square inch of my backyard for treasure?

Solution

The implementation strategy we will take to solve this problem will be to implement a simple metal detector with an autonomous moving robot. The robot will scan an area for metal with the metal detector and will alert through different means when metal is found. This is quite a simple and effective solution to the problem at hand. A robot can be easily programmed and it will do all the manual work of searching with the metal detector and the user can sit back and have the treasure come to him, literally!

We will implement a simple robot with 2 motors, a metal detector, and a FPGA board to interface with all the components together and to be the brains of the machine. A wireless controller will control from a “comfortable” distance and may report its findings through a monitor after it is complete its search.

Achievements

Keypad

The Keypad, like all general contact switches was difficult to implement. A de-bounce system had to be created to produce clean inputs and a valid pulse that could be read by the transmitter system.

Rangefinders

The sharp GP2DO2 rangefinder input and output serial communications proved to be too complicated to implement. It had timing requirements that were too detailed and specific and tests with the devices could not produce valid results.

Clock Divider

The clock divider implemented in other projects used various means to create timing signals which were based on complicated processes using if-then statements and the main clock to produced longer pulses. Our project required millisecond clocking for the slower components and as such the general and simple method we used was the implement an lpm counter with 16 bits width with all the bits set as individual output ports. This produced a much more stable output.

Stepper Motor Drive Controller Interface

The stepper motor circuitry used SGS Thompson chips to provide the direct power control of the four stepper motor phases. The UP1 board is not capable of driving the motor with the necessary current or protect itself against the back emf surges of the motor phases. The controller accepts 4 main inputs that when grounded change the action of the motor. The controls are direction, reset, step, and full or half step. The step signal requires microsecond pulses to be sent to the controller at millisecond rates so the motors will drive smoothly and at no more then 300 steps per second. This is the limit of these and most stepper motors.

Power Supply

For the remote systems on the motor a battery system was required to provide power to the UP1 board, the radio transmitter and receiver, the motors and the metal detector. The power requirements for the systems was about 0.75 Amp while the motors were in operation.

Body Design

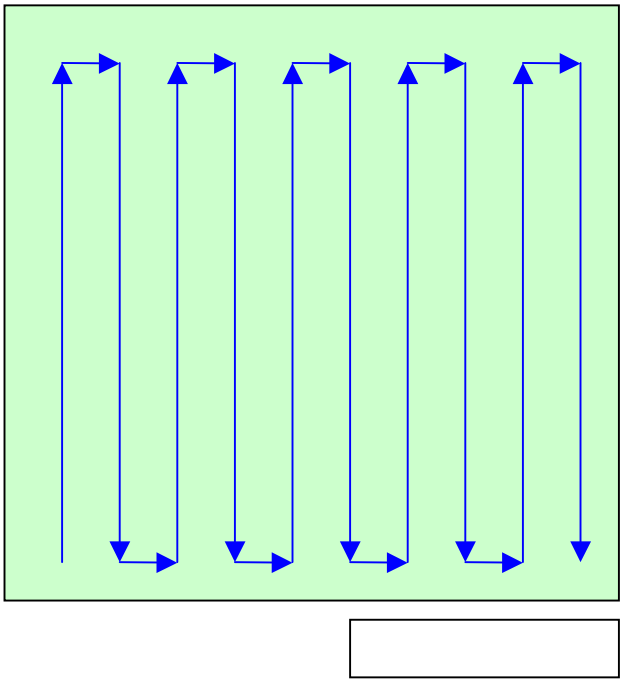
Though the concept and design of the body mechanics was our own, credit must go to the electrical machine shop for the fabrication of the parts. The system would not be able to be implemented without their help.

RF Transmitter and Receiver System

The Ramsey TXE-433 and RXD-433 were poorly documented and much experimentation was require to allow the system to work. The input and output required buffering to move the data correctly. The lpm FIFO buffer was difficult to implement and required a great deal of work to drive.

Rover Control System

The control system for the rover require a lot of thought and experimentation to make the system work with the complex timing constraints of the hardware components used in the project. The rover had to operate in either manual (remote control) or autonomously. The system required a large amount of states to give it adequate settling time for all signals. Control and conditioning of all signals was a very time consuming issue and proved the need for a well planned pre-design of the system into sub-state modules. The large number of states was necessary to allow the pattern of movement below under autonomous operation. The control system receives information as to the manual direction (initiated with the "*" on the base station keypad) indicated by a number (1-9) on the keypad. If a autonomous search pattern is required, the "#" is pressed and a value (1-9) is then pressed to indicate the square side distance of the pattern to follow.



FPGA Datasheet

Overview

The chip's function is as follows:

- Interface to a beat oscillator metal detector
- Interface with Ramsey™ wireless transmitters and receivers
- Interface to stepper motor controllers
- Interface to generic 12 button keypad
- Interface to simple VGA display

Implementation

The VHDL design implemented onto an Altera EPFK20RC240-4 makes the FPGA into a chip that accepts input from a beat oscillating metal detector, moves itself around or by remote control and transfers data to and from a bay station via wireless link.

The metal detector is highly sensitive to aluminum cans and metal objects that have a higher surface area than the search coil. The beat oscillating metal detector operates in the region of 280KHz to 320KHz. This value is very good for differentiating between the metal naturally found in the ground and metal actually buried under the ground. This can be modified to have optimum results in different environments and search item.

The wireless controllers send and receive 4bits of data asynchronously. Upon receiving a data valid, the chip accepts the data as valid input from the sender. The main interface from the user is given through a simple 12-button keypad (similar to a telephone's keypad 0-9, *, and #.) Information on the robot is sent back to a remote station and displayed on a simple monitor.

Powered by external Lithium-Ion batteries, the remote metal detector and on-board components can sustain full operation for 3 continuous hours using 2 1500Wh batteries. The metal detector can sustain 10 hours of continuous operation on a single 1500Wh Lithium-Ion battery. Other power sources are available for use and performance will vary.

The IO for the chip is as follows:

Inputs/Outputs (# of inputs/outputs)	**Description**
Wireless Transmitters (5)	4 lines for data, 1 for data valid
Wireless Receivers (5)	4 lines for data, 1 for data valid
Metal Detector (1)	For beat frequency from metal detector
Keypad Interface (8)	Simple Keypad interface
Stepper Motor controllers (4)	Stepper motor interface board
VGA connector (15 [D-sub])	Standard interface to monitor

Specifications of chip:

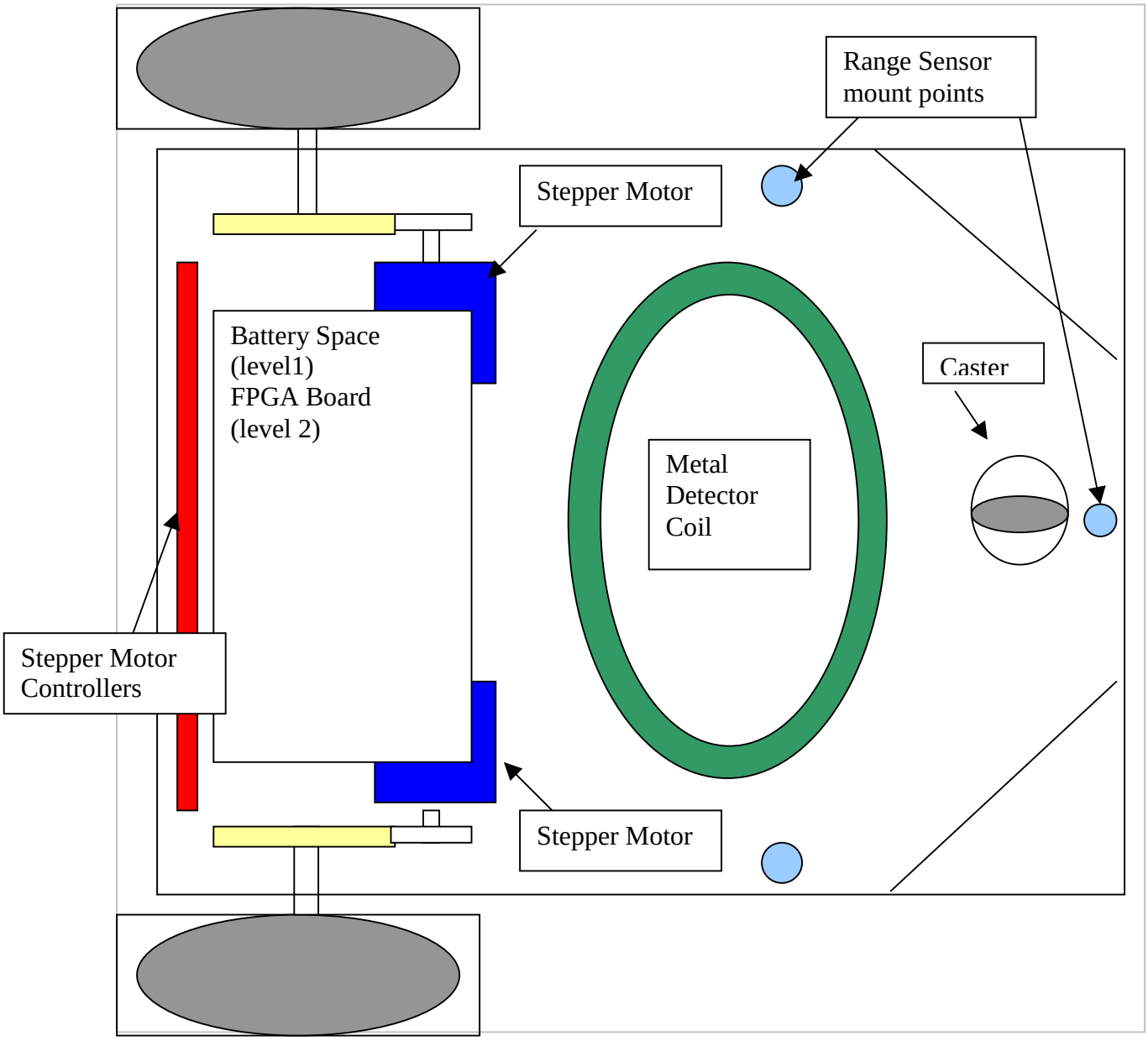
Specification	Value
Main Clock	25.175MHz
Maximum wireless transfer rate	64bps
Maximum stepper rate	90rpm
Metal Detector Sampling Frequency	6.25KHz
Metal Detector Operating Frequency	280KHz to 325 KHz
VGA Display	External (through D-sub)
VGA Max Resolution	640x480
VGA Display Max Refresh	60Hz @ 640x480
PS/2 Interface	Available [not implemented]
USB Interface	None
FireWire	None
Ethernet	None
DC Input	7 to 12 V RAW
JTAG Input	10-pin female
Operating Altitude	0m to 3000m
Maximum Storage Altitude	5000m
Relative Humidity	20% to 80% non-condensing
Operating Temperature	10^{o} C to 40^{o} C
Storage Temperature	-50^{o} C to $+60^{o}$ C

Additional expansion is available through the use of options available on-board the Altera UP1 board. Please check the UP1 documentation on Altera's site for more details.

IO Pin layout of the expansion slots is indicated on the following page:

Utilization of the FLEX_EXPAN_B Connector I/O Pins					
Row 1 Pin #	**Flex Chip Pin**	**Usage**	**Row 2 Pin #**	**Flex Chip Pin**	**Usage**
1	RAW		2	GND	
3	VCC		4	GND	
5	VCC		6	GND	
7	No Connection		8	DI1/99	
9	DI2/92		10	DI3/210	
11	DI4/212		12	DEV_CLR/209	
13	DEV_OE/213		14	DEV_CLK2/211	
15	109		16	110	
17	111	Reset	18	113	Reset
19	114	Half/Full	20	115	Half/Full
21	116	Clock	22	117	Clock
23	118	CW/CCW	24	119	CW/CCW
25	120		26	126	
27	127		28	128	
29	129		30	131	
31	132		32	133	
33	134		34	136	
35	137		36	138	
37	139		38	141	
39	142		40	143	
41	144		42	146	
43	147		44	148	
45	149	Bit A	46	151	Bit A
47	152	Bit B	48	153	Bit B
49	154	Bit C	50	156	Bit C
51	157	Bit D	52	158	Bit D
53	159	Enable	54	161	Data Valid
55	162		56	163	
57	VCC		58	GND	
59	VCC		60	GND	

Robot Body


Range Sensor
mount points
Stepper Motor
Battery Space
(level1)
FPGA Board
(level 2)
Caster
Metal
Detector
Coil
Stepper Motor
Controllers
Stepper Motor

Experiments and Results

Simulating VGA in Max Plus II v.10.0

In development of the driver for VGA, I wanted to do a test simulation on Max Plus II to ensure that we would get a good mark on our report with pages and pages of simulations.

No, really, I wanted to verify that the code works. I had setup on version 10.0 of Altera's software (for windows 2000 support) on my machine, which is pretty tough machine (Dual Celeron 466MHz, 128MB RAM, 40GB hard drive (over 3 drives, 2 of them being SCSI).)

I hit the simulation run with the system clock at 5.0ns and simulated for 1.0s. This was not a wise idea.

What happened next was just extremely poor performance and my hard drives thrashing like they never thrashed before. I was just able to open my task manager to see what the resource usage was like. I was impressed that virtual memory climbed to 1GB!!! It's good to know that windows 2000 does support that much RAM, even if it's virtual. Once it hit the 1GB mark, Max Plus II crashed. Maybe it can't address all that RAM Windows allocated for it.

Anyways, the conclusion to this experiment is that DO NOT SIMULATE A WHOLE VGA DISPLAY on Max Plus II. I wanted to see what would happen for a single screen but I couldn't.

At the end, I ended up trying my code and it worked fine the first time and did the rest of my development using trial and error. Test benches might work out here but simulation time would take a long time (640 x 480 = 307,200) and not worthwhile.

16 Colors

Another experiment I wanted to try with the VGA display was to increase the amount of colors available to the screen. There was an application note on 16 colors on the VGA by alternating the output using the clock (no half voltages are allowed for output of the pins, but clocked pulses with 50% duty cycles are allowed and would be considered ½ of the full voltage).

Implementation of this proved to be tricky, especially from the code provided. The sample code there is barely operational and does not demonstrate clearly this feature. In general, the input clock must be reduced so its effects can be more noticeable. This feature is quite the "hack" and not useful for a real project. Other techniques employed by other groups have seen to be more useful and provide better results.

Stepper Motor Timing

Originally, a finite state machine was used in combination with standard counters used in the EE552 labs. The counters would not consistantly stop after a set period and would begin to run continuously. The issue of creating timing for the stepper motor input to the controller cards required a 5 us low pulse to initiate a step. The pulse needed to be consistant and regular and had minimum time and maximum times for the delay between steps and the length of the low pulse. The timing control with the state machine/counter system was prone to missed time and run-on counting.

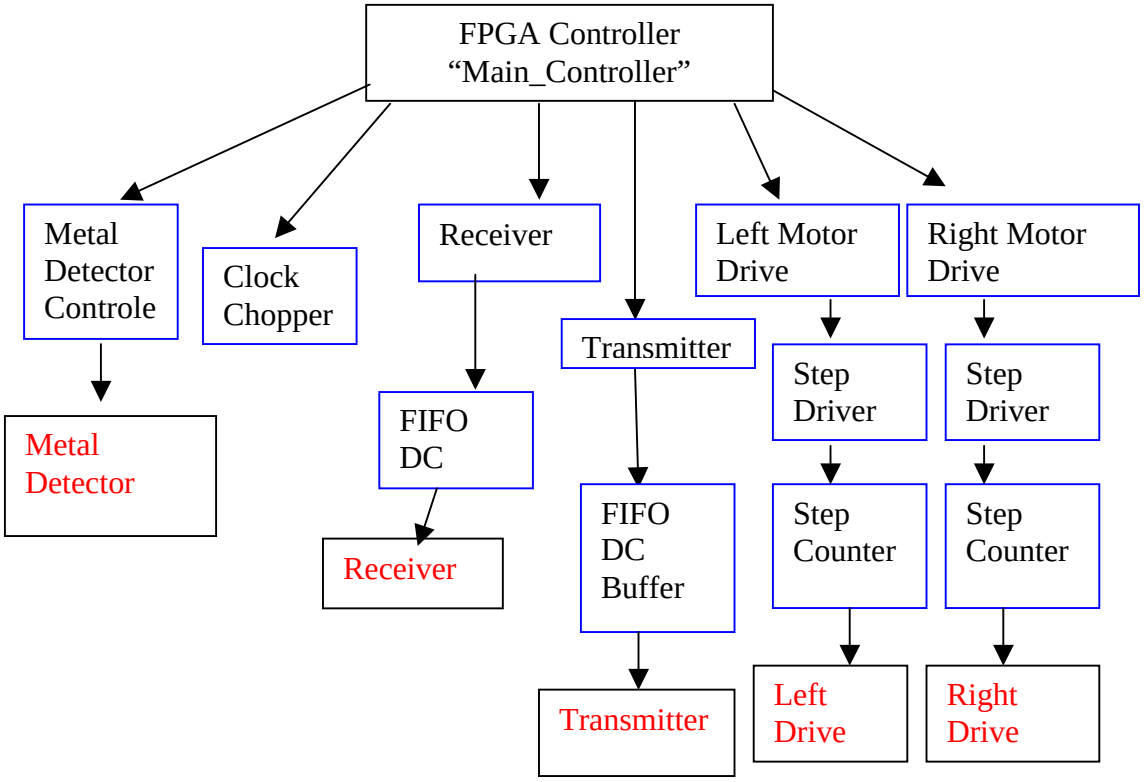
The solution was found by the use of purely RTL code with a chained lpm counter-comparitor combination. A stable output was produced and the system only required a 5us clock pulse to work perfectly in symulation.

Transmitter & Reciever IO Interface

The documentation for the transmitter and receiver was limited to a schematic a sentence describing the IO as consisting of a 4 bit word and a valid or enable signal. There was no information of the speed limitations or other information on the systems. The slow speed of the transmission required a FIFO buffer to transfer the information. Unfortunately the compilation of lpm FIFO buffer recommended by Altera would crash when an attempt to fit it on the chip space. A buffer of 8 x 5bit words was later used and the system would then compile. The results have been inconsistent and the system is still prone to faults at this point. The megafunction system in MaxPlus2 is not well documented and should probably be avoided.

ROVER Design Hierarchy

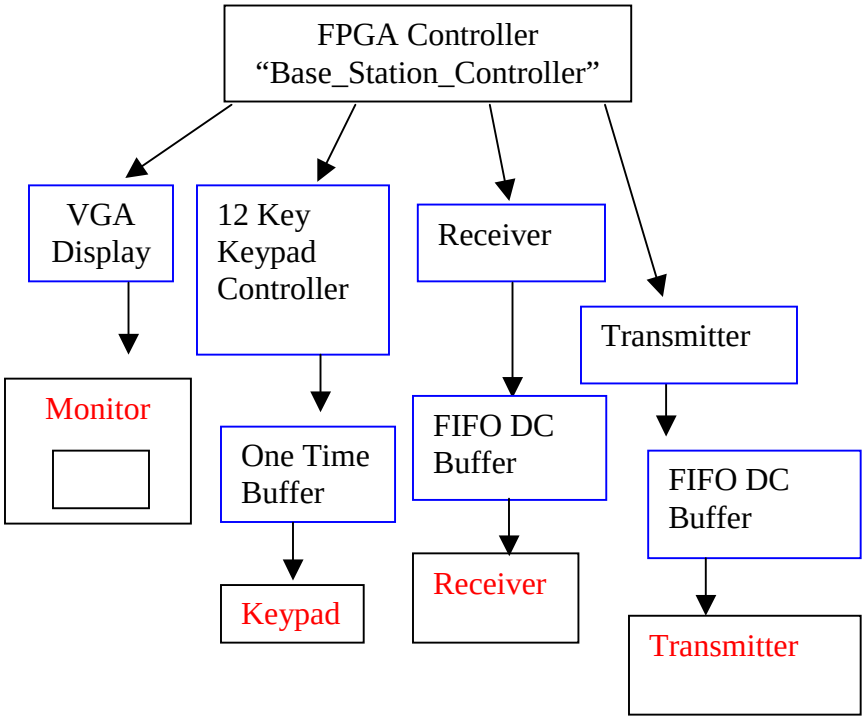
Below is the design hierarchy for the remote robot, Rover. Boxes with red text refer to end hardware components. Blue boxes refer to VHDL code blocks. Rover is to be powered by itself with its own onboard power.



*Note: red text signifies actual physical components.

Base Station Design Hierarchy

The base station is to provide feedback to the user and also provide an interface to control Rover.


FPGA Controller
“Base_Station_Controller”
VGA Display
12 Key Keypad Controller
Receiver
Transmitter
Monitor
One Time Buffer
FIFO DC Buffer
FIFO DC Buffer
Keypad
Receiver
Transmitter

VHDL Design

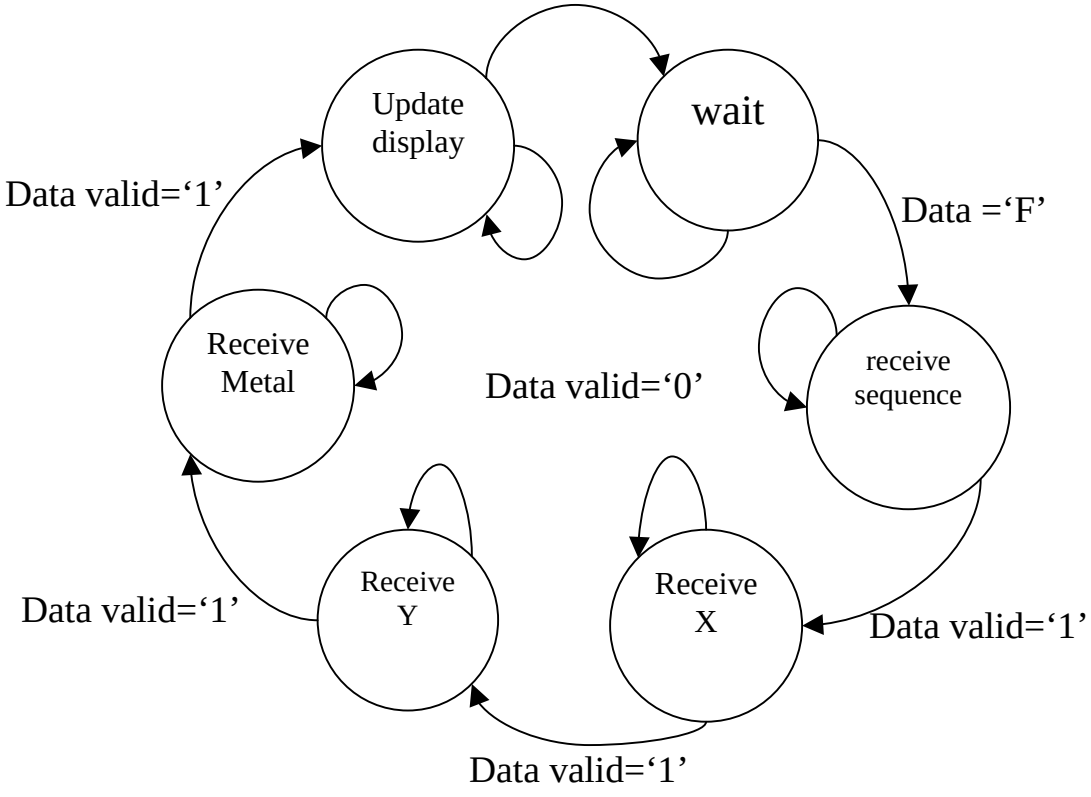
Below are various design for VHDL code blocks. Specifically it covers the control codes used for communication between Rover and the base station.

Receiving Data From Rover

Through the wireless controllers, we will pull data from "Rover" on the value of the metal detector at the particular instance. This value from Rover will display on screen in a colour. The display of the colour will be different for 2 modes of searching. In autonomous mode, a grid representing the search area will be shown and as each block on the grid is passed, the intensity of the metal in the area will be shown on screen with a colour. In manual mode, only a single block in the middle of the screen will show the intensity of the metal at that instance.

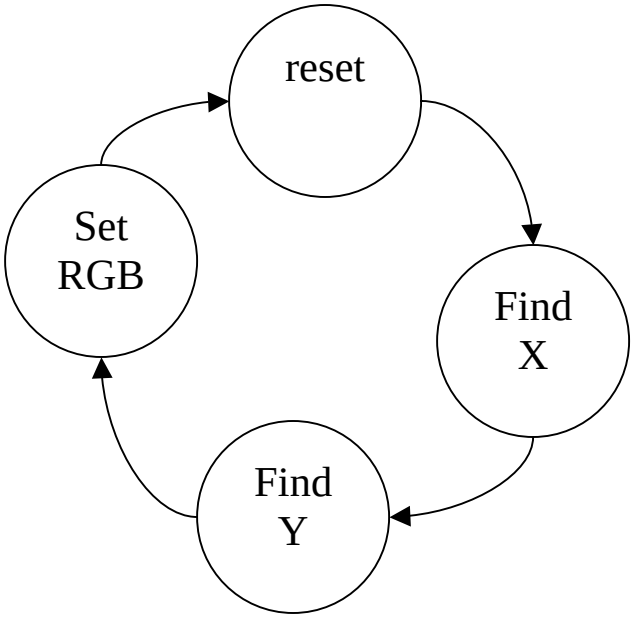
The state machine flow are below

The receive data flow:

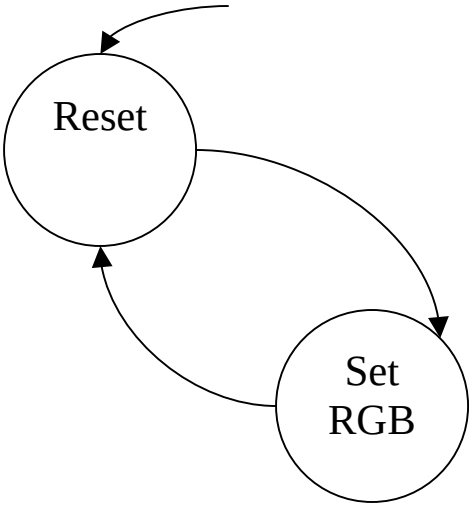


The updating of the monitor:

In autonomous mode

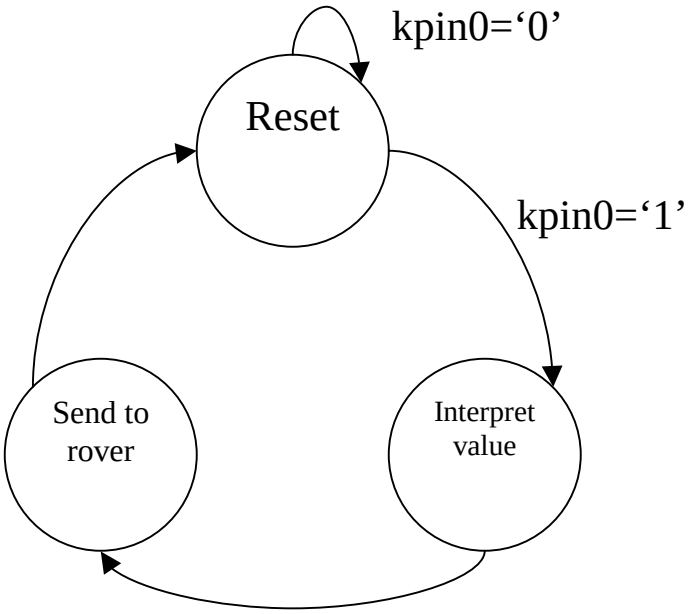


In manual mode



Sending Data To Rover

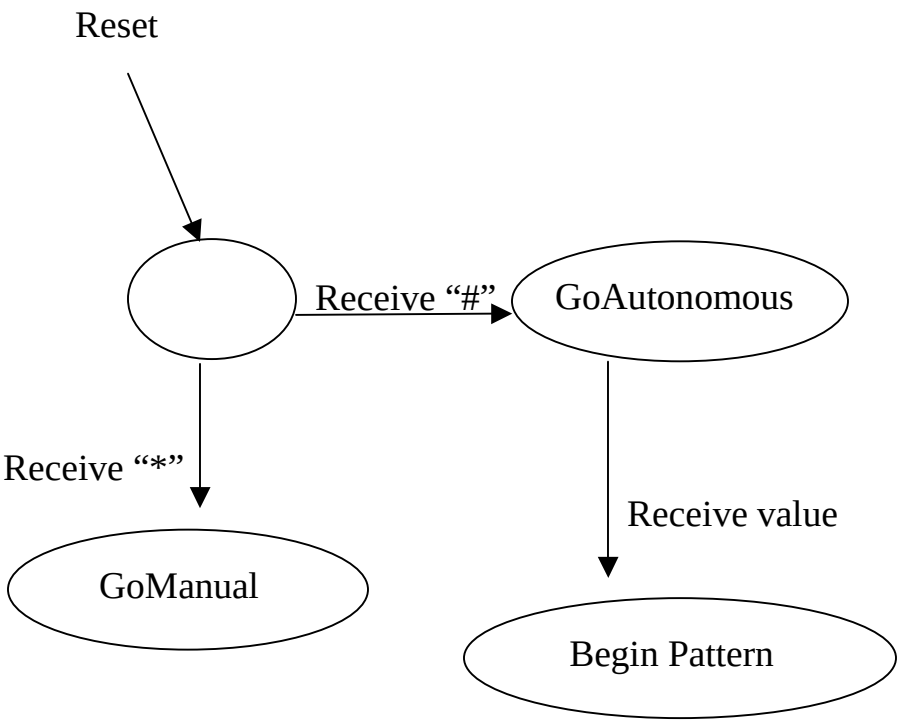
Sending data to rover is quite simple. We see when there is a key pressed on the keypad at the station. When there is a key press, we look up the value in a table and then send the value out to Rover and return to the reset state.



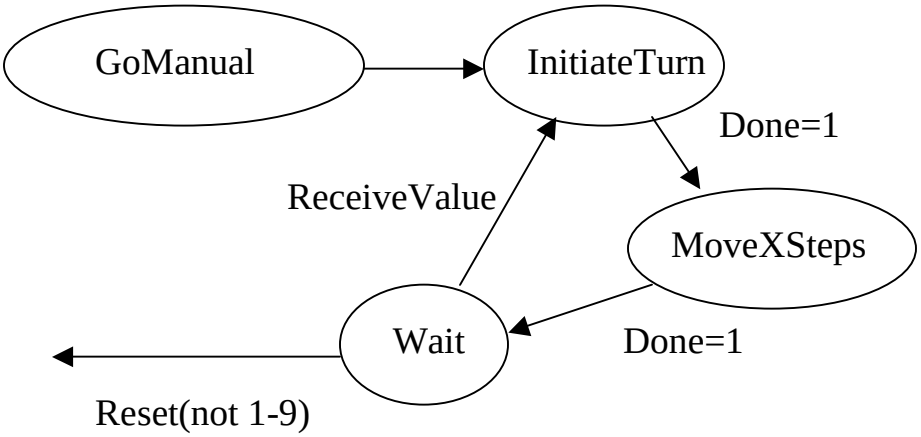
Rover

General Operation

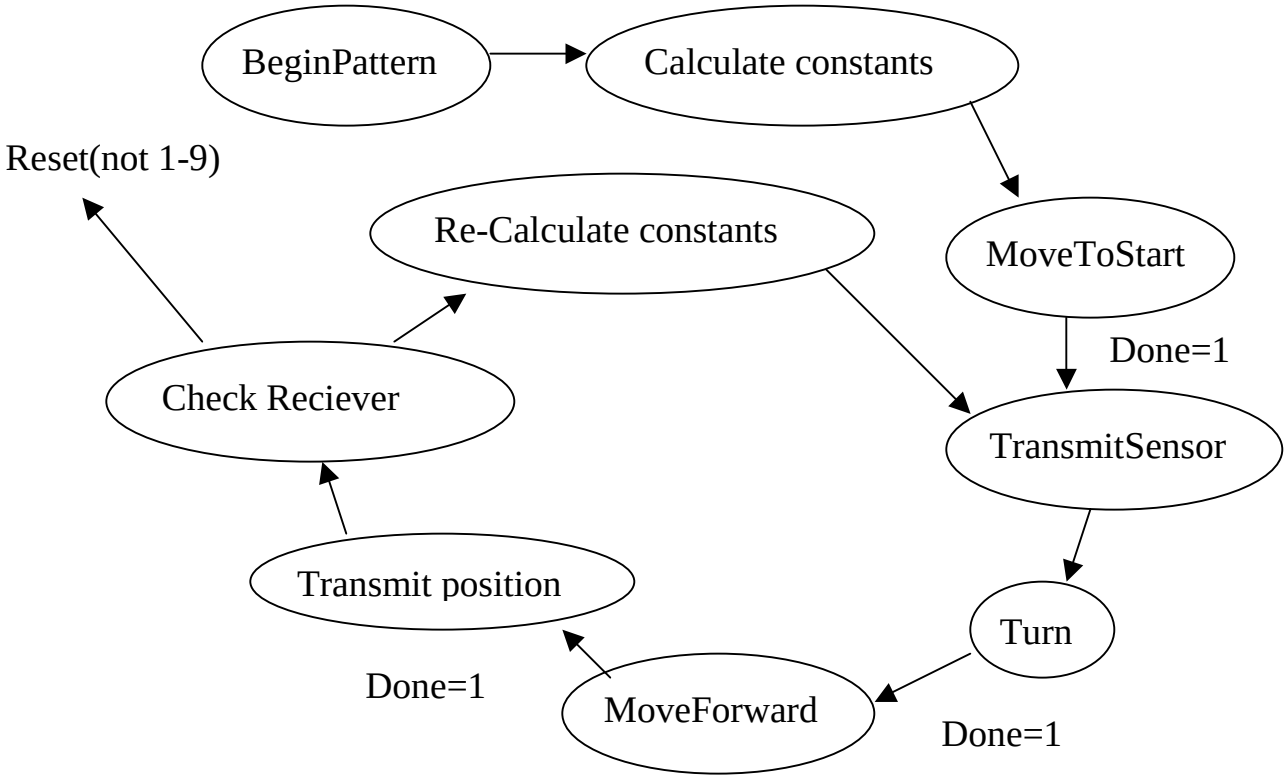
The state diagram of the rover actions:



Manual Operation



Autonomous Operation



References

1. Steve & Rachel Hagman, “Single IC Metal Detector”, EDN , Dec 1998
2. SGS Thompson Datasheet, “THE L297 STEPPER MOTOR CONTROLLER”, Application Note
3. “Mobile Robots 2nd Ed.”, J. Jones, A. Flynn, B. Seiger, A K Peters Ltd, 1999

Declaration of Original Content

The design elements of this project and report are entirely the original work of the authors except as follows:

1. The schematic of the single IC Metal Detector example in Fig. ( ) was taken from [1]
2. The original circuit design of the Stepper motor controller PCB layout in Fig. ( ) was taken from [2]
3. The stepper motor controllers were used in a previous course at the University of Alberta, EE 582.
4. The design of the body of the robot is original but the credit for its fabrication goes to Berry in the electrical engineering mechanical shop.