

SCHOOL *of* BUSINESS AND TECHNOLOGY

Department of Engineering and Aviation Sciences

**Design of a Ground Rover with TI mmWave Radar Sensor for Geographical Acquisition in Disaster Rescue Applications**

**Favour Dada**

**Jesudara Omidokun**

**Aduralere Sulaiman**

Advisor: Dr. Lei Zhang

May. 14, 2019

Project Title

By

Team member 1, team member 2, …

Submitted to the Department of Engineering and Aviation Sciences in partial fulfillment of the requirements for the degree of Bachelor of Science in Engineering at the

UNIVERSITY OF MARYLAND EASTERN SHORE

Date

The author hereby grants the University of Maryland Eastern Shore permission to reproduce and distribute publicly, paper and or electronic copies of this document in whole or in parts.

Authors Team member 1, team member 2, …

Signature

Date

Department of Engineering and Aviation Sciences

Certified by

**Special notice for formatting for the whole report:**

1. **Keep the formatting consistent (learn and apply the MS ‘Styles’ in formatting). Do not change any format but just apply corresponding styles as listed.**
2. **Do not add any blank rows but use BREAKS, except the end of a table. BREAKS can be found on ‘Page Layout’ → ‘Page Setup’ → ‘Breaks’.**
3. **Every figure or picture must be mentioned in the text first then can appear. When it is mentioned, use the cross-reference.**
4. **Every figure or picture in the text must be accompanied with the caption and detailed description.**
5. **Only third person is allowed (except the Conclusion and Acknowledge sections). Never use ‘we’, ‘our’, ‘you’, etc. anywhere else (use the ‘find’ function of MS Office to double check).**
6. **Use ‘do not’ other than ‘don’t’, and so forth.**
7. **You can copy any reference information from anywhere, but anything not from yourself must be refereed and the original information must be included in the Reference section.**
8. **The reference must be organized with professional software such as ‘Endnote Web’, ‘Mendeley’, etc. Use IEEE reference format.**
9. **Try Grammarly. It can help in fixing typos and grammar errors.**
10. **If you get ‘Styles’ messed up, the easiest way is to start over, to open this blank template and copy-paste contents into it.**

List of Contents

[List of Contents 2](#_Toc536393135)

[List of Figures 4](#_Toc536393136)

[List of Tables 5](#_Toc536393137)

[Abstract 6](#_Toc536393138)

[1. Introduction 7](#_Toc536393139)

[1.1 Backgound/Motivation 7](#_Toc536393140)

[1.2 Objective 7](#_Toc536393141)

[1.3 Design Requirements 7](#_Toc536393142)

[1.4 Design Constraints 7](#_Toc536393143)

[1.5 Design Methods 7](#_Toc536393144)

[2. Project Description 9](#_Toc536393145)

[2.1 System Description 9](#_Toc536393146)

[2.2 System Diagram 9](#_Toc536393147)

[2.3 System Functions 9](#_Toc536393148)

[3. Implementation Plan 11](#_Toc536393149)

[3.1 Tasks 11](#_Toc536393150)

[3.2 Team Organization 12](#_Toc536393151)

[3.2.1. Responsibility of Team Member 1. 12](#_Toc536393152)

[3.2.2. Responsibility of Team Member 2. 12](#_Toc536393153)

[3.3 Timeline/Milestones/Delivery Plan 12](#_Toc536393154)

[4. Implementation 13](#_Toc536393155)

[4.1 Implementation of Task 1. 13](#_Toc536393156)

[4.1.1. Implementation of Subtask 1.1 13](#_Toc536393157)

[4.2 Implementation of Task 1. 13](#_Toc536393158)

[5. Conclusion (Discussion and Future Plans) 14](#_Toc536393159)

[Acknowledgment 15](#_Toc536393160)

[Appendix 16](#_Toc536393161)

[A. Component Specs 16](#_Toc536393162)

[1. Specs of Arduino Due 16](#_Toc536393163)

[2. Specs of Raspberry Pi 16](#_Toc536393164)

[B. Source Code. 16](#_Toc536393165)

[1. Source Code of Graphic User Interface 16](#_Toc536393166)

[2. Source Code of Robotic Arm 16](#_Toc536393167)

[REFERENCES 17](#_Toc536393168)

List of Figures

[Table 1. Project Timeline and Delivery Plan 9](#_Toc453191241)

List of Tables

Abstract

By the end of the project, summarize the project into short text and put here.

1. Introduction

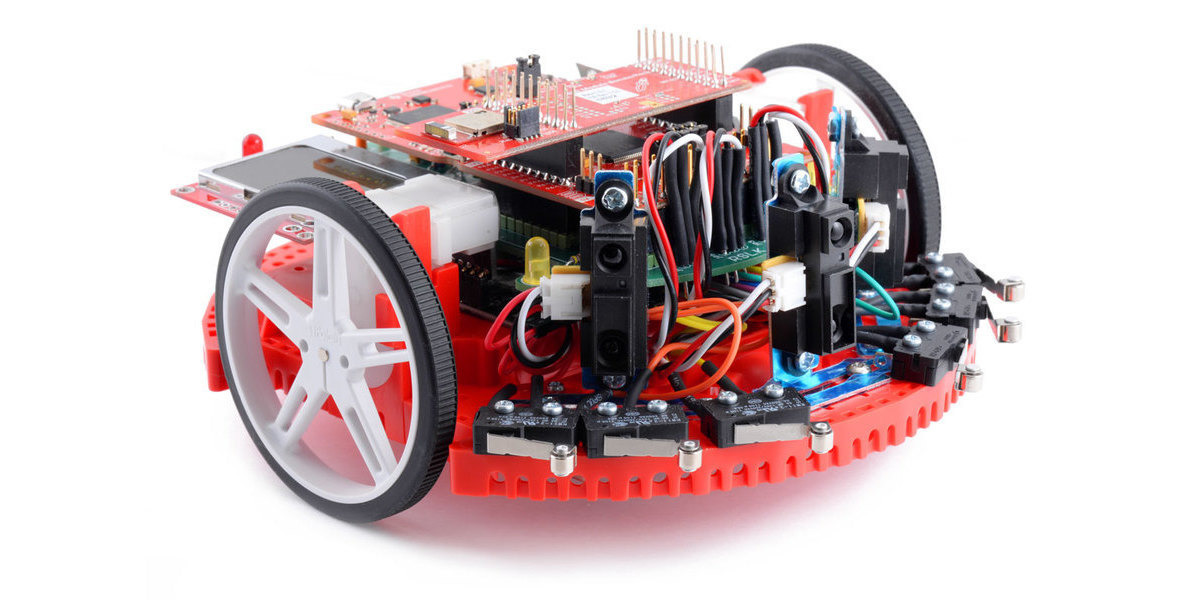
In this project, we are going to develop an autonomous car using the TI-RSLK and the mmWave sensor (IWR1443BOOST). The autonomous car will be capable of different functions such as providing geographical information (such as range, velocity and angle) and proximity & position sensing.

## Backgound/Motivation

The TI Robotics Systems Learning Kit was created by TI as a low-cost robotics kit to provide students with a profound understanding of how electronic devices work. It was developed in conjunction with Dr. Jon Valvano, professor, electrical and computer engineering at The University of Texas at Austin. There are two different RSLK kits; the basic kit which teaches the foundations of an electric system and the advanced kit which contains all components found in basic kit, adding wireless communication to enable robot to solve its way through a maze by detecting lines and obstacles.

The mmWave sensor is an extremely valuable sensing technology for detection of objects and providing the range, velocity and angle of these objects. It is a contactless-technology which operates in the spectrum between 30GHz and 300GHz. Due to the technology’s use of small wavelengths it can provide sub-mm range accuracy and is able to penetrate certain materials such as plastic, drywall, clothing, and is impervious to environmental conditions such as rain, fog, dust and snow. The mmWave sensor has a clear advantage over other sensing technologies such as ultrasonic sensors and LIDAR. It has a long detection range, narrow and wide detection angle, good range resolution, and has good night operation performance.

The rover can be used in many situations such as in hostage situations and school shootings. In such situations, instead of law enforcement agencies to go in with guns blazing, it is better to access the situation at hand. The mmWave sensor on the rover can provide 3D point cloud information on the building where the situation occurred. It will also work better than a camera because of its long detection range, it’s narrow and wide detection angle, as well as it is good range resolution. From the 3D point cloud information provided by the mmWave sensor, the rover would be able to provide the best possible route for the officers to go through without being noticed by the “bad guys”.



1. Robotic System learning kit (RSLK)



1. Ultrasonic Sensor



1. Laser distance sensor

## Objective

The objective of the project is to design a rover that can acquire geographical acquisition such as the range, velocity, angle and 3D imaging of objects in an unknown area.

## Design Requirements

1. The device will be able to produce data using a point cloud information.
2. The device will be able to avoid obstacles while providing autonomous navigation
3. The payload of the car will be 15 to 20lbs
4. The device will detect object within the range 3cm to 10 m.
5. The device will have a resolution range of 10cm

## Design Constraints

1. The tires might not be able to move well in certain terrains such as rocky areas.
2. Battery will have to be recharged after about 4 hours of use
3. The device would only work in mild weather conditions.

## Design Method (Approach)

Design methods refer to methods to explore possibilities and constraints by applying critical thinking through qualitative and quantitative research methods to develop solutions for your purposes.

Shortly, design methods are basic process to implement the project, for example, “The first step is to … The second step is to … …”.

## Standards

List in this section all (industry) standards the project complied with.

1. Project Description

## System Description

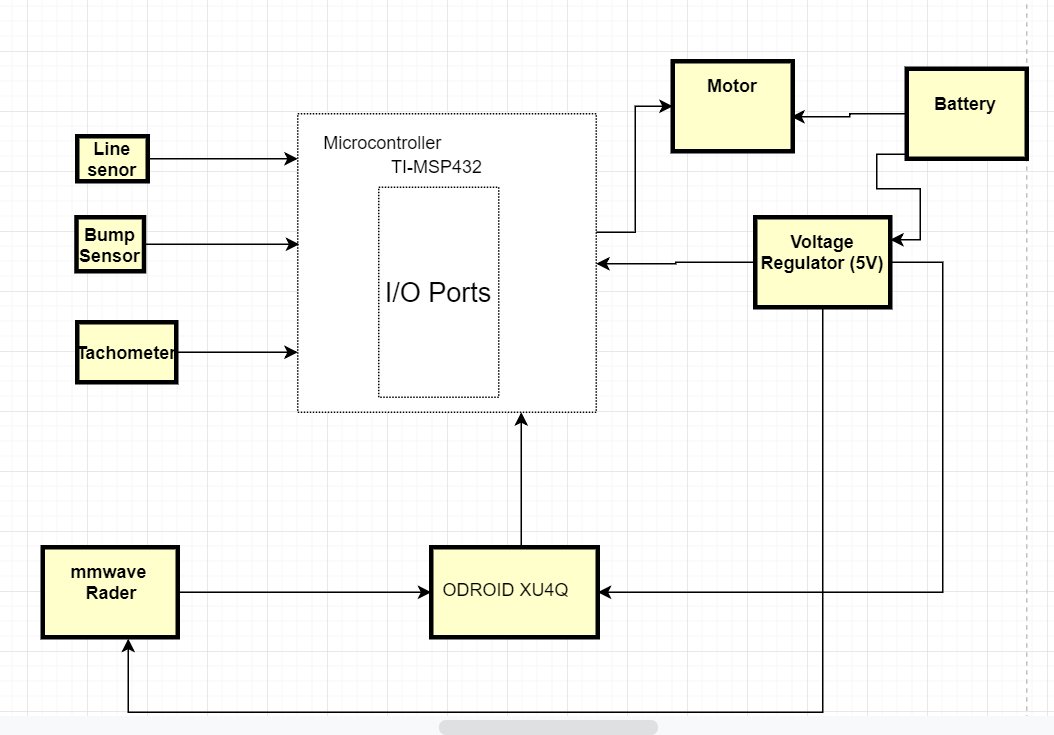
An autonomous car is controlled by a TI microcontroller MSP432 power by a 12-volt battery with a voltage regulator controlling the voltage input to the microcontrollers. The battery powers the Dc motor and sends the DC signal to a PWM input on the microcontroller. A IWR 1443 Boost mmWave Rader is connected to the microcontroller to server as a senor to the car. Also, a raspberry pi is connected to the output signal of the microcontroller to control, access and save the data acquire from the sensors.

## System Diagram

Draw a system diagram here. The diagram should include every components and show there relationships. For complex system you can have multiple system diagrams, for example, the vehicle mechanical system diagram and electrical system diagram. Or if you have several subsystems, you can also draw diagrams for each of them.

Please make sure you marked the name/function of each component on the diagram, for example, 'motor' or 'wireless receiver'.

You should also put descriptions of every major component here.



1. System Diagram

## System Functions

Here you need to clearly define every function and every state of the system. Make you’re your state graph is complete. For example:

"

1. When the caller press a button, first the message will show on the display of the device with the caller, also the message will be coded together with its ID and send to the callee.
2. …

"

1. Implementation Plan

## Tasks

* Task 1. RLSK Assembly
  + Subtask 1. Soldering, Assembling and Wiring of TI RSLK Kit
  + Subtask 2. Writing and Modifying the Code for Bumps Sensors
  + Subtask 3. Writing and Modifying of the Code for Line Follow Sensor
* Task 2. Design of the computer system
  + Subtask 1. Setting up Ubuntu on the ODRIOD XU4Q
  + Subtask 2: Design of a power adapter for mmWave
  + Subtask 2. Implementing the mmWave interface with the microcomputer
* Task 3. Implementation and development of the mmWave sensor (IWR 1443 Boost)
  + Subtask 1. Mounting the mmWave to RSLK board
  + Subtask 2. Design and print a 3D part to help mount the mmWave
* Task 4. Acquisition of the point cloud data from the radar with the microcomputer.
  + Subtask 1. Launch the ti\_mmWave\_rospkg node to start the mmWave EVM
  + Subtask2: Visualize the Radar point cloud using RVI
* Task 5. Stitching the acquired 3d point cloud data.
* Task 6. Detecting and calculating the distance of an object behind the glass wall.
* Task 7. Using the Tachometer to calculate the speed of the rover.
* Task 8. System Finalization, testing and refinement.

"

## Team Organization

### Responsibility of Aduralere Sulaiman

### Responsibility of Favour Dada.

Task 1, Subtask 2.2, …

### Responsibility of Jesudara Omidokun

…

## Timeline/Milestones/Delivery Plan

1. Project Timeline and Delivery Plan

|  |  |  |  |
| --- | --- | --- | --- |
| **Time** | **Task** | **Comments** | **Responsible Personnel** |
| Week 1-4 | Task 1 | Assembly of the TI RSLK Kit  Compilation of the code | Aduralere Sulaiman  Favour Dada  Jesudara Omidokun |
| Week 3-8 | Task 2 | Setting up of Ubuntu on ODROID XU4Q  Connecting mmWave interface with computer | Aduralere Sulaiman  Jesudara Omidokun  Favour Dada |
| Week 5-7 | Task 3 | Connecting the mmWave to the RSLK  Compiling the code for mmWave sensor | Aduralere Sulaiman  Favour Dada  Jesudara Omidokun |
| Week 11-17 | Task 4,5 | Acquisition of the point cloud data from the radar with the microcomputer.  Stitching the acquired data  Calculating the velocity independent of the wheel rotation | Jesudara Omidokun  Aduralere Sulaiman  Favour Dada |
| Week 17-19 | Task 6,7 | Detecting and calculating the distance of an object behind the glass wall.  Use of Tachometer to calculate speed of rover | Aduralere Sulaiman  Jesudara Omidokun  Favour Dada |
| Week 19-22 | Task 8 | System finalization and delivery. Finish all documentations and ready for presentation. | Favour Dada  Jesudara Omidokun  Aduralere Sulaiman |

1. Task assingment for spring 2019

|  |  |  |  |
| --- | --- | --- | --- |
| Time | Jesudara | Aduralere | Favour |
|  |  |  |
| Week 1 | Task 1, Sub 1a.  Soldering Components of RSLK task | Task 1, Subtask 2.  Write. the code for the bump Sensors. | Task 1, Subtask 3.  Write. the code for the line Sensors |
| Week 2 | Task 1, Subtask 1b.  Wiring Components of RSLK | Task 1, Subtask 2.  -Write. the code for the bump Sensors | Task 1, Subtask 3.  Write. the code for the line Sensors. |
| Week 3 | Task 1, Subtask 1c.  Assembling of RSLK component | Task 1, Subtask 2.  Helping with the Assembling of RSLK | Task 1, Subtask 3.  Writ. the code for the line Sensor |
| Week 4 | Task 1, Subtask 1c.  Assembling of RSLK component | Task 1, Subtask 2.  Assembling of RSLK, compiling code, Uploading | Task 2, Subtask 1.  Assembling of RSLK. Compiling codes, Uploading codes |
| Week 5-7 | Task 3, Subtask 1,2  Mounting technic and 3d design and printing | Task 2, Subtask 3.  Implementing the mmWwave interface with the microcomputer | Task 2, Subtask 1.  Design Power supply for ODROID Using Buck converter  Setting up Ubuntu on ODROID XU4Q |
| Week 7-11 | Task 4.  Acquisition of the point cloud data from the radar with the microcomputer | Task 4.  Acquisition of the point cloud data from the radar with the microcomputer | Task 5.  Stitching the acquired point cloud data. |

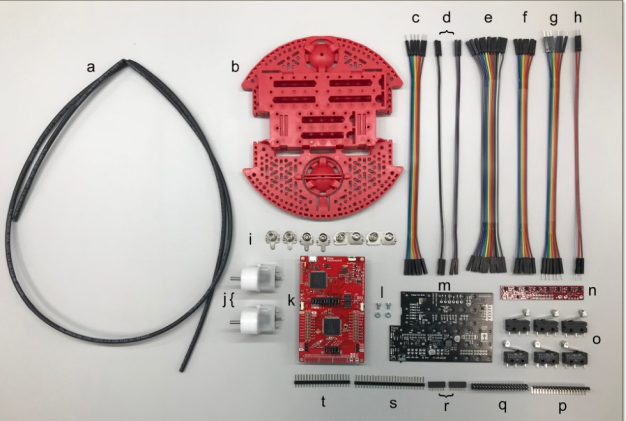
1. Task assingment for fall 2019

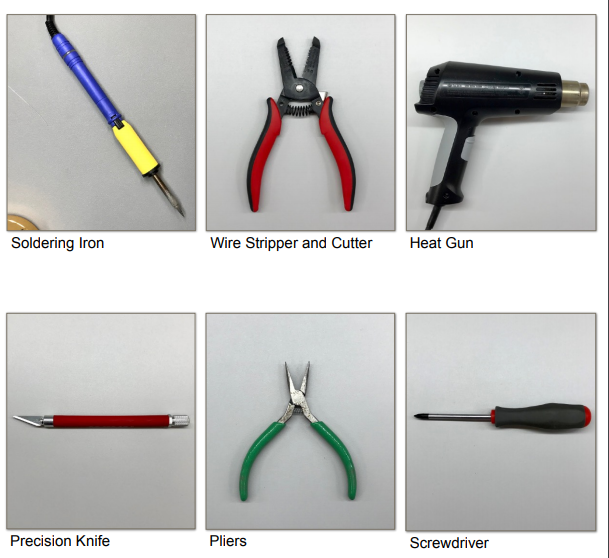
|  |  |  |  |
| --- | --- | --- | --- |
| Week 11-13 | Task 4.  Acquisition of the point cloud data from the radar with the microcomputer | Task 4.  Acquisition of the point cloud data from the radar with the microcomputer | Task 5.  Stitching the acquired data. |
| Week 14-17 | Task 7.  Using the Tachometer to calculate the speed | Task 6.  Detecting and calculating the distance of an object behind the glass wall. | Task 6.  Detecting and calculating the distance of an object behind the glass wall. |
| Week 18-20 | Task 8.  System Finalization, testing and refinement. | Task 8.  System Finalization, testing and refinement | Task 8.  System Finalization, testing and refinement |
| Week 21- 22 | . |  |  |

1. Implementation.

## Implementation of Task 1.

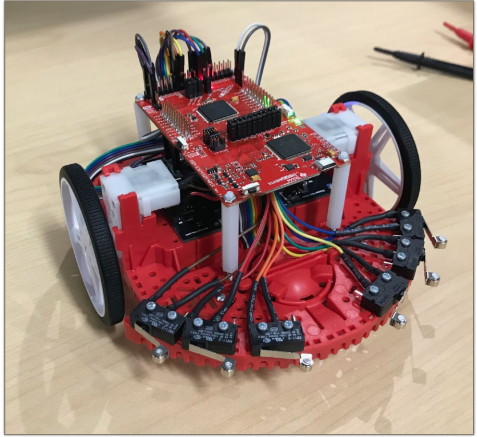
### Implementation of Subtask 1.1

This involves the use of different tools, wires and supplies to assemble the TI-RSLK. 



1. The pictures above show some of the tools needed to assemble the rover.

After the RSLK parts were assembled, the final assembly looked similar to the picture below;



1. Here is a the assembled RSLK

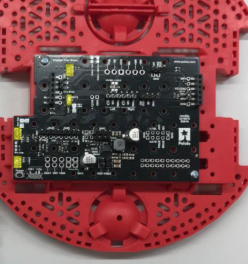
The tasks for the TI-RSLK was divided into four main sections; soldering, assembly and software.

**Soldering**

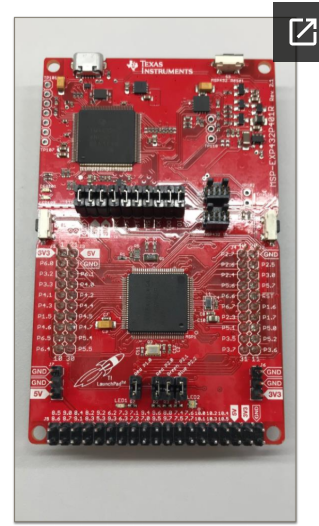
1. The components of the TI RSLK KIT

|  |  |  |  |
| --- | --- | --- | --- |
|  | Description |  |  |
| a | Heat Shrink Tube | 1 |  |
| b | Chassis | 1 |  |
| c | 6 Female to Male Wires | 1 |  |
| d | 2 Female to Male Wires | 4 |  |
| e | 11 Female to Female Wires | 1 |  |
| f | 6 Female to Female Wires | 1 |  |
| g | 6 Male to Male Wires | 1 |  |
| h | 2 Male to Male Wires | 1 |  |
| i | Battery Terminals | 1 |  |
| j | Motor | 2 |  |
| k | TI Launchpad kit | 1 |  |
| l | Motor Board Screw | 1 |  |
| M | Motor Board | 1 |  |
| N | Line Follower Sensor | 1 |  |
| O | Bump Switch | 6 |  |
| p | 90 Bent Header | 1 |  |
| q | 2x20 Header | 1 |  |
| r | 1x6 Header | 1 |  |
| s | 1x25 Header | 1 |  |
| t | 1x20 Header | 1 |  |
| u | Motor Encoders | 2 |  |
| V | Bump switch wire spade connector | 12 |  |

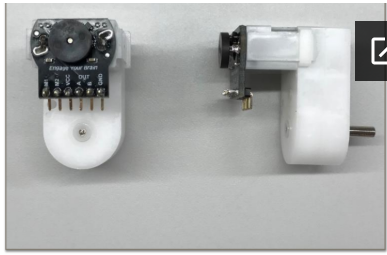
A precision knife is used to cut the VPU-VREG, VCCMD-VREG, and SLP L-R traces. A deep solid cut is then made through the small line connection between the square pads. Two 1x6 female socket headers are connected to the yellow connections before connected to the 1x2 headers and then to the ELA and ELB connections. The ELA and ELB connections need to be bent to a 45-degree angle after soldering. A 1x2 header cut is made on the VPU connection. A 1x3 header is made and so is a 1x6 header cut to the left and right motor driver connections. 1x3 and 1x2 header cuts are made to the VREG terminals. The 1x8,1x3 and 1x2 header cuts are soldered to the GND terminals. The four battery connections were soldered as part of the attachment to the chassis and the motor board was secured with two support screws. Pliers are then used to bend out the ELA and ELB & ERA and ERB connections so the connections can be attached by wires to the Launchpad.



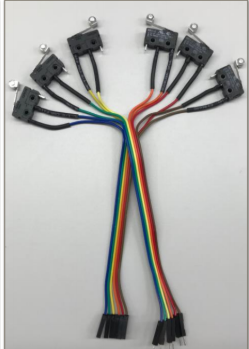
Launchpad: The 2x19 header is then soldered on the J5 pinout at the bottom of the launchpad with the long pins facing upwards.



Motors: The prepared encoders are then soldered on the motors.

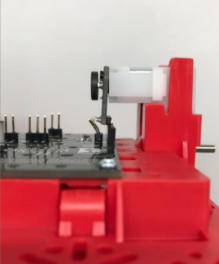


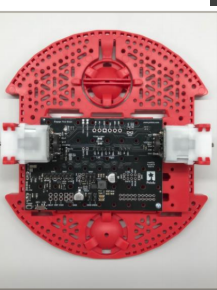
Bump switches: One male and female wire each is soldered to each bump sensor. This is done for the 6 bump sensors.



**Assembly**

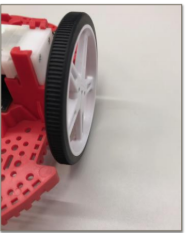
The motors are attached to the motor board on the chassis.



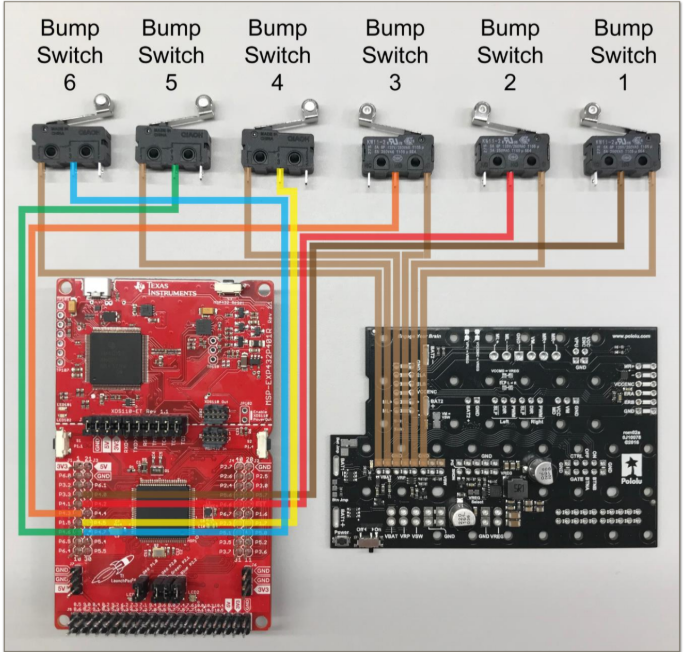


The ball caster is then attached to the chassis before the wheels are attached.

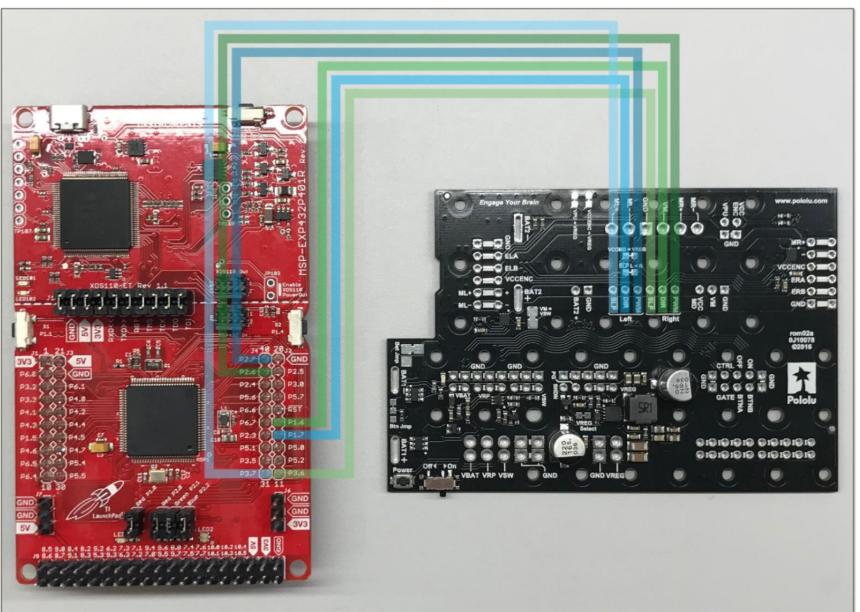
 



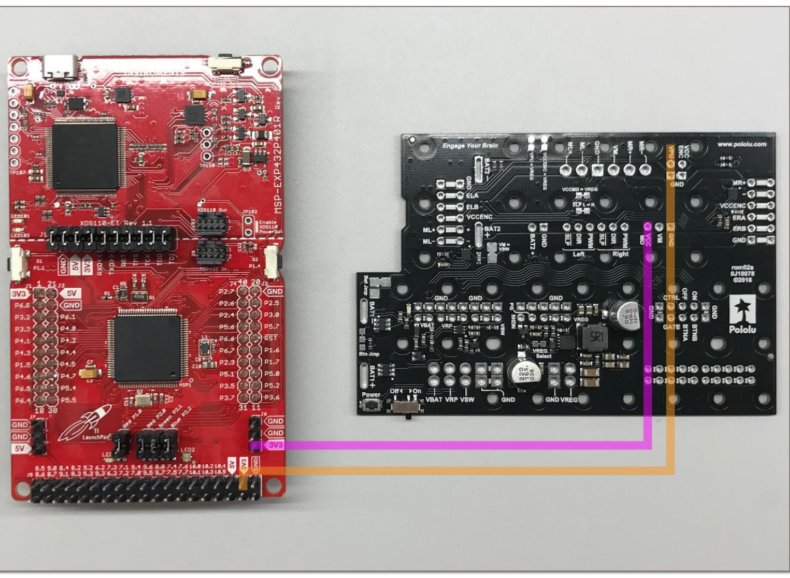
**Wiring**



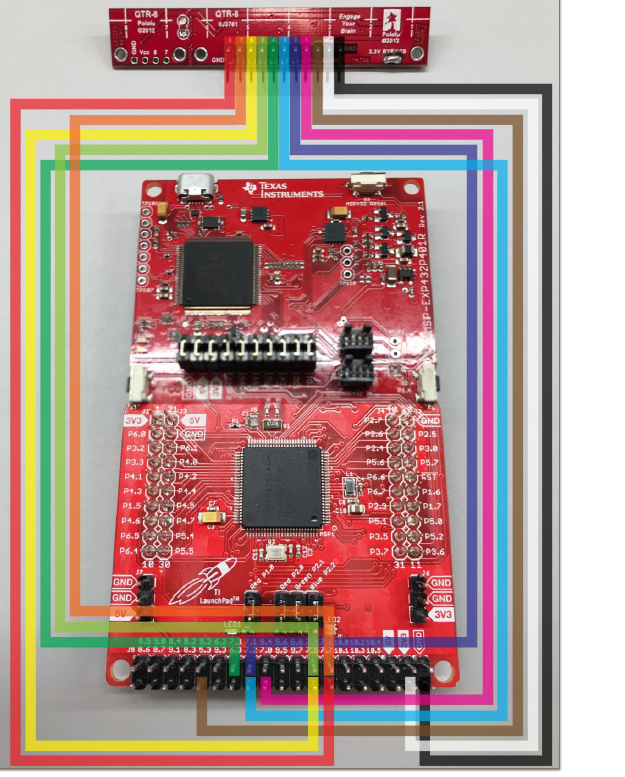
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Bump 1 | Bump 2 | Bump 3 | Bump 5 | Bump 6 |
| LaunchPad | P 4.0 | P.4.2 | P.4.5 | P.4.6 | P 4.7 |



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Motor Board | Left SLP | Left DIR | Left PWM | Right SLP | Right DIR | Right PWM |
| Launchpad | P3.7 | P1.7 | P2.7 | P3.6 | P1.6 | P2.6 |



|  |  |  |
| --- | --- | --- |
| Motor Board | VREG | GND |
| Launchpad | 5V | GND |



|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Line Sensor | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | LED ON | VCC | GND |
| Launchpad | P7.7 | P7.6 | P7.5 | P7.4 | P7.3 | P7.2 | P7.1 | P7.0 | P5.3 | 3V3 | GND |

**Software**

The steps to get the software working were;

* Install CCS (Code Composer Studios)
* Import the TI Drivers
* Download RSLK Energia Library

Troubleshooting

Soldering:

The first issue encountered when soldering was having to solder the connectors upside down as this affected the wiring because the female connector wire removed due to the fact that the pins have different diameter from top to bottom.

The solution to this issue was to de-solder the connectors and then re-solder.

Another issue that was encountered when soldering was with the encoders. The encoder connector pins are flexible which made to fragile. During the attachment of the motor board to the motor, an encoder pin broke causing the motor to stop working.

The solution to this issue was to de-solder the encoder and find a less fragile pin and connector to use.

The motor also had a similar issue, but the only solution was to buy another motor.

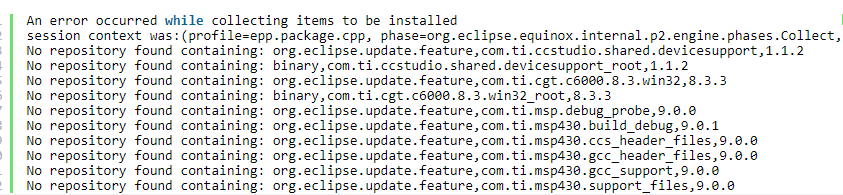
Assembly:

The main issue when assembling the parts was from the chassis motor clips. The chassis motor clips from the TI RSLK kit didn’t sit well on the chassis.

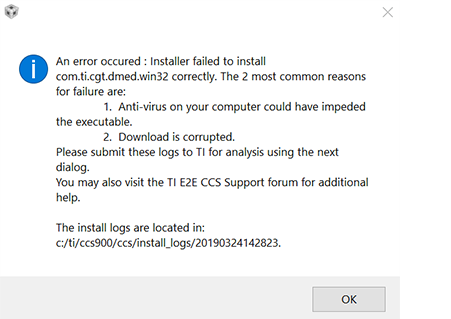
The solution to this issue was to break part of the motor clips in order to fit the motors. The motors where then slid into the motor clips without over-stressing the clips.

Software:

One of the challenges was installing ccstudio on the computer. Different errors occurred during the installations. One error message was;

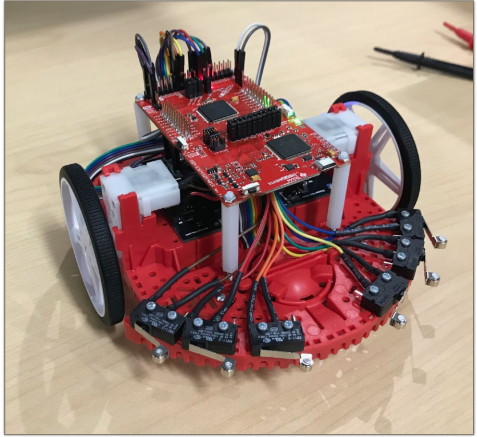


CCS was then deleted and re-downloaded. However, during the installation, another error occurred with the message;



This problem was then solved by turning off the antivirus on the computer before the installation.

After the RSLK parts were assembled, the final assembly looked similar to the picture below;



### Implementation of Subtask 1.2

### Implementation of Subtask 1.3

For this task the code for the line tracker. The RSLK has two line sensors one on the left and one for the right. The RSLK has two motors.

For the two sensors (left right)

1 1 explains that the RSLK is on the line

1 0 explains that the RSLK is off to the right

0 1 explains that the RSLK is off to the left

0 0 explains that the RSLK is lost

If the car detects black lines when there are white line the RSLK would only go through the white lines. The line sensor follows black lines only. The motor driver runs the finite state machine every 50 microseconds.

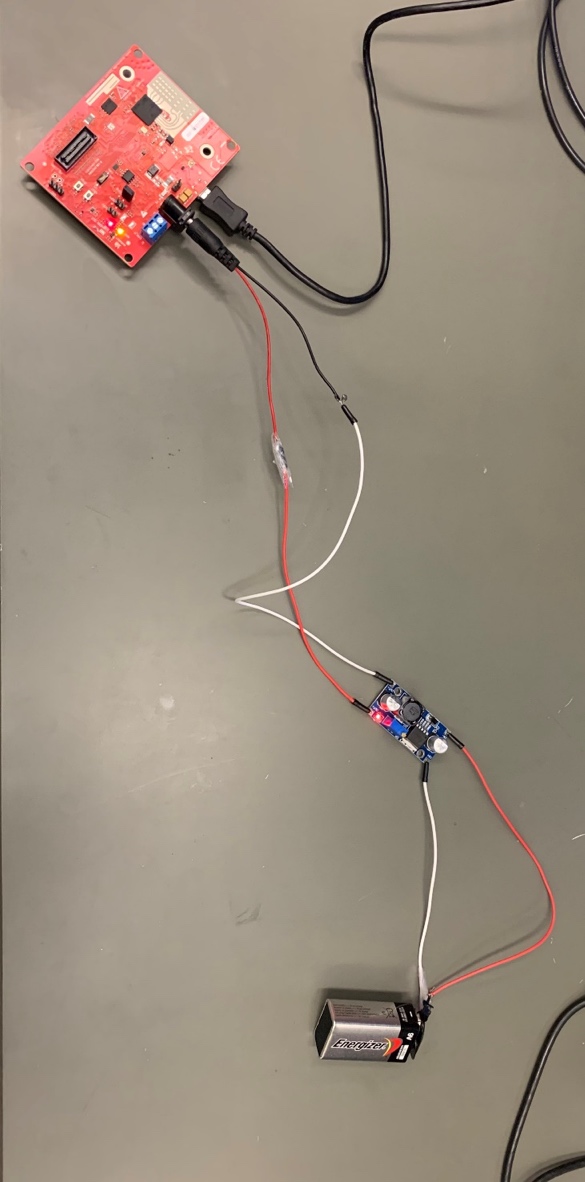
The source code for the line tracker can be found in Appendix section under source code.

…

## Implementation of Task 2

### Implementation of Task Subtask 2.1

For this task the work was to use a battery cone LM256 has a transformer and the steps are from 35 to 225 volts and has a variable resistor used to regulate the voltage. Then took the multimeter to regulate the voltage of 5v and also a maximum current of 2.5A.



1. Power adapter connected to mmWave

## Implementation of Task 5

### Implementation of Task Subtask 5.1

Fig below explains that for a stitching of a 3d point cloud data the first step is to register a collection of point cloud data that was collected from a robotic operating system visualizer. The robotic operating system visualizer stores files as a .obj file. To stitch the 3d point cloud data collected , the .obj file is converted to a .pcd file. The code for converting the .obj file to .pcd can be found in the appendix section. For this task the

### Implementation of Task Subtask 5.2

For this task the stitching algorithm was implemented on the TI mmWave sensors IWR1443 EVM. The mmWave has an hardware accelerator engine that supports the FFT processing for a 1D FFT(range) calculation. The code enables a highly reflective object to be detected in front of the EVM. The stitching code implementation shows a 2 step FFT of 1024-pointFFT with windowing of fractional interpolation turned on. The 1024 iteration of 4-point FFT. Each step of the FFT processing is handled by a parameter set. The Data path has the chirp data processed in a ping pong manner having the Mx as the local memory of the hardware accelerator.

ADC\_ping (M0) -> 1K FFT -> M2 -> 4 FFT -> M3 -> (EDMA) -> L3

ADC\_ping (M1) -> 1K FFT -> M3 -> 4 FFT -> M32-> (EDMA) -> L3

The ADC output buffer is shared between M0 and M1, there is no need for EDMA transfer until the output data is ready for transfer to radar cube memory L3.

…

1. Conclusion (Discussion and Future Plans)

By the end of the project, conclude the project and your learning experience.

Acknowledgment

Dr. Alveron Walker

Appendix

You can put reference info here, including: i) specs of components used in the system, ii) source code (must be here but not in the body text), iii) CAD figures, etc.

1. Component Specs
2. Specs of Arduino Due

...

1. Specs of Raspberry Pi

…

1. Source Code.
2. Source Code of Line Sensor
3. Source Code of Bump Sensor
4. Source Code of 4k FFT Stitching Algorithm

#include <stdint.h>

#include <stdlib.h>

#include <stddef.h>

#include <string.h>

#include <stdio.h>

/\* BIOS/XDC Include Files. \*/

#include <xdc/std.h>

#include <xdc/runtime/System.h>

#include <ti/drivers/hwa/hwa.h>

#include "config\_hwa\_util\_4k\_fft.h"

#include "data\_path\_4k\_fft.h"

#include "mmw.h"

extern mmwHwaBuf\_t gMmwHwaMemBuf[MMW\_HWA\_NUM\_MEM\_BUFS];

uint32\_t log2Approx(uint32\_t x)

{

uint32\_t idx, detectFlag = 0;

if ( x < 2)

{

return (0);

}

idx = 32U;

while((detectFlag==0U) || (idx==0U))

{

if(x & 0x80000000U)

{

detectFlag = 1;

}

x <<= 1U;

idx--;

}

if(x != 0)

{

idx = idx + 1;

}

return(idx);

}

void HWAutil\_configRangeFFT(HWA\_Handle handle,

uint32\_t paramSetStartIdx,

uint32\_t numAdcSamples,

uint32\_t numRangeBins,

uint8\_t numRxAnt,

uint32\_t windowOffsetBytes,

uint8\_t dmaTriggerSourcePing,

uint8\_t dmaTriggerSourcePong,

uint8\_t dmaDestChannelPing,

uint8\_t dmaDestChannelPong,

uint16\_t hwaMemAdcBufOffset,

uint8\_t hwaTriggerMode)

{

HWA\_InterruptConfig paramISRConfig;

int32\_t errCode = 0;

uint32\_t paramsetIdx = paramSetStartIdx;

uint32\_t pingParamSetIdx1 = 0;

uint32\_t pingParamSetIdx2 = 0;

HWA\_ParamConfig hwaParamCfg[HWAUTIL\_NUM\_PARAM\_SETS\_1D];

memset(hwaParamCfg,0,sizeof(hwaParamCfg));

hwaParamCfg[paramsetIdx].triggerMode = HWA\_TRIG\_MODE\_DMA;

hwaParamCfg[paramsetIdx].dmaTriggerSrc = dmaTriggerSourcePing;

hwaParamCfg[paramsetIdx].accelMode = HWA\_ACCELMODE\_NONE; //dummy

errCode = HWA\_configParamSet(handle,paramsetIdx,&hwaParamCfg[paramsetIdx],NULL);

/\* if (errCode != 0)

{

MmwDemo\_debugAssert (0);

return;

}\*/

paramsetIdx++;

pingParamSetIdx1 = paramsetIdx;

hwaParamCfg[paramsetIdx].triggerMode = hwaTriggerMode;

hwaParamCfg[paramsetIdx].accelMode = HWA\_ACCELMODE\_FFT;

hwaParamCfg[paramsetIdx].source.srcAddr = hwaMemAdcBufOffset;

hwaParamCfg[paramsetIdx].source.srcAcnt = numAdcSamples/4 - 1;

hwaParamCfg[paramsetIdx].source.srcAIdx = 4 \* numRxAnt \* sizeof(uint32\_t);

hwaParamCfg[paramsetIdx].source.srcBcnt = 3;

hwaParamCfg[paramsetIdx].source.srcBIdx = sizeof(uint32\_t);

hwaParamCfg[paramsetIdx].source.srcShift = 0;

hwaParamCfg[paramsetIdx].source.srcCircShiftWrap = 0;

hwaParamCfg[paramsetIdx].source.srcRealComplex = HWA\_SAMPLES\_FORMAT\_COMPLEX;

hwaParamCfg[paramsetIdx].source.srcWidth = HWA\_SAMPLES\_WIDTH\_16BIT;

hwaParamCfg[paramsetIdx].source.srcSign = HWA\_SAMPLES\_SIGNED;

hwaParamCfg[paramsetIdx].source.srcConjugate = 0;

hwaParamCfg[paramsetIdx].source.srcScale = 8;

hwaParamCfg[paramsetIdx].source.bpmEnable = 0;

hwaParamCfg[paramsetIdx].source.bpmPhase = 0;

hwaParamCfg[paramsetIdx].dest.dstAddr = ADDR\_TRANSLATE\_CPU\_TO\_HWA(MMW\_HWA\_1D\_OUT\_PING\_M0\_M2);

hwaParamCfg[paramsetIdx].dest.dstAcnt = numRangeBins/4-1;

hwaParamCfg[paramsetIdx].dest.dstAIdx = 4 \* numRxAnt \* sizeof(uint32\_t); //

hwaParamCfg[paramsetIdx].dest.dstBIdx = sizeof(uint32\_t);

hwaParamCfg[paramsetIdx].dest.dstRealComplex = HWA\_SAMPLES\_FORMAT\_COMPLEX;

hwaParamCfg[paramsetIdx].dest.dstWidth = HWA\_SAMPLES\_WIDTH\_16BIT;

hwaParamCfg[paramsetIdx].dest.dstSign = HWA\_SAMPLES\_SIGNED;

hwaParamCfg[paramsetIdx].dest.dstConjugate = 0;

hwaParamCfg[paramsetIdx].dest.dstScale = 0;

hwaParamCfg[paramsetIdx].dest.dstSkipInit = 0;

hwaParamCfg[paramsetIdx].accelModeArgs.fftMode.fftEn = 1;

hwaParamCfg[paramsetIdx].accelModeArgs.fftMode.fftSize = log2Approx(numRangeBins/4);

hwaParamCfg[paramsetIdx].accelModeArgs.fftMode.butterflyScaling = 0x3;

hwaParamCfg[paramsetIdx].accelModeArgs.fftMode.interfZeroOutEn = 0;

hwaParamCfg[paramsetIdx].accelModeArgs.fftMode.windowEn = 1;

hwaParamCfg[paramsetIdx].accelModeArgs.fftMode.windowStart = windowOffsetBytes; //start of window RAM

hwaParamCfg[paramsetIdx].accelModeArgs.fftMode.winSymm = 0;

hwaParamCfg[paramsetIdx].accelModeArgs.fftMode.winInterpolateMode = 1; //4Kstitching, enabling window interpolation

hwaParamCfg[paramsetIdx].accelModeArgs.fftMode.magLogEn = HWA\_FFT\_MODE\_MAGNITUDE\_LOG2\_DISABLED;

hwaParamCfg[paramsetIdx].accelModeArgs.fftMode.fftOutMode = HWA\_FFT\_MODE\_OUTPUT\_DEFAULT;

hwaParamCfg[paramsetIdx].complexMultiply.mode = HWA\_COMPLEX\_MULTIPLY\_MODE\_DISABLE;

errCode = HWA\_configParamSet(handle,paramsetIdx,&hwaParamCfg[paramsetIdx],NULL);

paramsetIdx++;

pingParamSetIdx2 = paramsetIdx;

hwaParamCfg[paramsetIdx].triggerMode = HWA\_TRIG\_MODE\_IMMEDIATE;

hwaParamCfg[paramsetIdx].accelMode = HWA\_ACCELMODE\_FFT; //do FFT

hwaParamCfg[paramsetIdx].source.srcAddr = ADDR\_TRANSLATE\_CPU\_TO\_HWA(MMW\_HWA\_1D\_OUT\_PING\_M0\_M2);

hwaParamCfg[paramsetIdx].source.srcAcnt = 3; //4 point FFT

hwaParamCfg[paramsetIdx].source.srcAIdx = 4;

hwaParamCfg[paramsetIdx].source.srcBcnt = numRangeBins/4 - 1;

hwaParamCfg[paramsetIdx].source.srcBIdx = 16;

hwaParamCfg[paramsetIdx].source.srcShift = 0;

hwaParamCfg[paramsetIdx].source.srcCircShiftWrap = 0;

hwaParamCfg[paramsetIdx].source.srcRealComplex = HWA\_SAMPLES\_FORMAT\_COMPLEX;

hwaParamCfg[paramsetIdx].source.srcWidth = HWA\_SAMPLES\_WIDTH\_16BIT;

hwaParamCfg[paramsetIdx].source.srcSign = HWA\_SAMPLES\_SIGNED;

hwaParamCfg[paramsetIdx].source.srcConjugate = 0;

hwaParamCfg[paramsetIdx].source.srcScale = 8;

hwaParamCfg[paramsetIdx].source.bpmEnable = 0;

hwaParamCfg[paramsetIdx].source.bpmPhase = 0;

hwaParamCfg[paramsetIdx].dest.dstAddr = ADDR\_TRANSLATE\_CPU\_TO\_HWA(MMW\_HWA\_1D\_OUT\_PING\_M2\_M3);

hwaParamCfg[paramsetIdx].dest.dstAcnt = 3;

hwaParamCfg[paramsetIdx].dest.dstAIdx = numRangeBins;

hwaParamCfg[paramsetIdx].dest.dstBIdx = 4;

hwaParamCfg[paramsetIdx].dest.dstRealComplex = HWA\_SAMPLES\_FORMAT\_COMPLEX;

hwaParamCfg[paramsetIdx].dest.dstWidth = HWA\_SAMPLES\_WIDTH\_16BIT;

hwaParamCfg[paramsetIdx].dest.dstSign = HWA\_SAMPLES\_SIGNED;

hwaParamCfg[paramsetIdx].dest.dstConjugate = 0;

hwaParamCfg[paramsetIdx].dest.dstScale = 0;

hwaParamCfg[paramsetIdx].dest.dstSkipInit = 0;

hwaParamCfg[paramsetIdx].accelModeArgs.fftMode.fftEn = 1;

hwaParamCfg[paramsetIdx].accelModeArgs.fftMode.fftSize = 2;

hwaParamCfg[paramsetIdx].accelModeArgs.fftMode.butterflyScaling = 0x3;

hwaParamCfg[paramsetIdx].accelModeArgs.fftMode.interfZeroOutEn = 0;

hwaParamCfg[paramsetIdx].accelModeArgs.fftMode.windowEn = 0;

hwaParamCfg[paramsetIdx].accelModeArgs.fftMode.windowStart = windowOffsetBytes;

hwaParamCfg[paramsetIdx].accelModeArgs.fftMode.winSymm = 0; //non-symmetric - in demo do we make this symmetric

hwaParamCfg[paramsetIdx].accelModeArgs.fftMode.winInterpolateMode = 0;

hwaParamCfg[paramsetIdx].accelModeArgs.fftMode.magLogEn = HWA\_FFT\_MODE\_MAGNITUDE\_LOG2\_DISABLED;

hwaParamCfg[paramsetIdx].accelModeArgs.fftMode.fftOutMode = HWA\_FFT\_MODE\_OUTPUT\_DEFAULT;

hwaParamCfg[paramsetIdx].complexMultiply.mode = HWA\_COMPLEX\_MULTIPLY\_MODE\_FFT\_STITCHING;

hwaParamCfg[paramsetIdx].complexMultiply.cmpMulArgs.twidIncrement = 1;

errCode = HWA\_configParamSet(handle,paramsetIdx,&hwaParamCfg[paramsetIdx],NULL);

if (errCode != 0)

{

System\_printf("Error: HWA\_configParamSet(%d) returned %d\n",errCode,paramsetIdx);

return;

}

paramISRConfig.interruptTypeFlag = HWA\_PARAMDONE\_INTERRUPT\_TYPE\_DMA;

paramISRConfig.dma.dstChannel = dmaDestChannelPing; //TODO sync this define EDMA channel to trigger to copy the data out

errCode = HWA\_enableParamSetInterrupt(handle,paramsetIdx,&paramISRConfig);

if (errCode != 0)

{

//System\_printf("Error: HWA\_enableParamSetInterrupt(PING DMA) returned %d\n",errCode);

MmwDemo\_debugAssert (0);

return;

}

paramsetIdx++;

hwaParamCfg[paramsetIdx].triggerMode = HWA\_TRIG\_MODE\_DMA;

hwaParamCfg[paramsetIdx].dmaTriggerSrc = dmaTriggerSourcePong;

hwaParamCfg[paramsetIdx].accelMode = HWA\_ACCELMODE\_NONE;

errCode = HWA\_configParamSet(handle,paramsetIdx,&hwaParamCfg[paramsetIdx],NULL);

if (errCode != 0)

{

MmwDemo\_debugAssert (0);

return;

}

paramsetIdx++;

hwaParamCfg[paramsetIdx] = hwaParamCfg[pingParamSetIdx1];

hwaParamCfg[paramsetIdx].dest.dstAddr = ADDR\_TRANSLATE\_CPU\_TO\_HWA(MMW\_HWA\_1D\_OUT\_PONG\_M1\_M3);

errCode = HWA\_configParamSet(handle,paramsetIdx,&hwaParamCfg[paramsetIdx],NULL);

if (errCode != 0)

{

System\_printf("Error: HWA\_configParamSet(%d) returned %d\n",errCode,paramsetIdx);

return;

}

paramsetIdx++;

hwaParamCfg[paramsetIdx] = hwaParamCfg[pingParamSetIdx2];

hwaParamCfg[paramsetIdx].source.srcAddr = ADDR\_TRANSLATE\_CPU\_TO\_HWA(MMW\_HWA\_1D\_OUT\_PONG\_M1\_M3);

hwaParamCfg[paramsetIdx].dest.dstAddr = ADDR\_TRANSLATE\_CPU\_TO\_HWA(MMW\_HWA\_1D\_OUT\_PONG\_M3\_M2); //hwaMemDestPongOffset;

errCode = HWA\_enableParamSetInterrupt(handle,paramsetIdx,&paramISRConfig);

if (errCode != 0)

{

MmwDemo\_debugAssert (0);

return;

}

/\* enable the DMA hookup to this paramset so that data gets copied out \*/

paramISRConfig.interruptTypeFlag = HWA\_PARAMDONE\_INTERRUPT\_TYPE\_DMA;

paramISRConfig.dma.dstChannel = dmaDestChannelPong;

errCode = HWA\_enableParamSetInterrupt(handle,paramsetIdx,&paramISRConfig);

if (errCode != 0)

{

System\_printf("Error: HWA\_enableParamSetInterrupt(PING DMA) returned %d\n",errCode);

return;

}

}

void HWAutil\_configDopplerFFT

(

HWA\_Handle handle,

uint32\_t paramSetStartIdx,

uint32\_t dopplerFftSize,

uint8\_t numVirtualAnt,

uint32\_t numRangeBinsPerIter,

uint32\_t windowOffsetBytes,

uint8\_t dmaTriggerSourcePing,

uint8\_t dmaTriggerSourcePong,

uint8\_t dmaDestChannelPing,

uint8\_t dmaDestChannelPong,

uint16\_t hwaMemSourcePingOffset,

uint16\_t hwaMemSourcePongOffset,

uint16\_t hwaMemDestPingOffset,

uint16\_t hwaMemDestPongOffset,

uint32\_t option

)

{

HWA\_ParamConfig hwaParamCfg[HWAUTIL\_NUM\_PARAM\_SETS\_2D];

uint32\_t paramsetIdx = 0;

int32\_t errCode = 0;

uint32\_t k;

HWA\_InterruptConfig paramISRConfig;

uint32\_t numDopplerBins = dopplerFftSize;

memset( (void\*) &hwaParamCfg[paramsetIdx], 0, sizeof(HWA\_ParamConfig));

//Doppler FFT

hwaParamCfg[paramsetIdx].triggerMode = HWA\_TRIG\_MODE\_DMA;

hwaParamCfg[paramsetIdx].dmaTriggerSrc = dmaTriggerSourcePing;

hwaParamCfg[paramsetIdx].accelMode = HWA\_ACCELMODE\_FFT;

hwaParamCfg[paramsetIdx].source.srcAddr = hwaMemSourcePingOffset; // address is relative to start of MEM0

hwaParamCfg[paramsetIdx].source.srcAcnt = dopplerFftSize - 1; //size in samples - 1

hwaParamCfg[paramsetIdx].source.srcAIdx = numVirtualAnt \* sizeof(uint32\_t); //

hwaParamCfg[paramsetIdx].source.srcBcnt = numVirtualAnt - 1;

hwaParamCfg[paramsetIdx].source.srcBIdx = sizeof(uint32\_t);

hwaParamCfg[paramsetIdx].source.srcShift = 0;

hwaParamCfg[paramsetIdx].source.srcCircShiftWrap = 0;

hwaParamCfg[paramsetIdx].source.srcRealComplex = 0;

hwaParamCfg[paramsetIdx].source.srcWidth = HWA\_SAMPLES\_WIDTH\_16BIT;

hwaParamCfg[paramsetIdx].source.srcSign = HWA\_SAMPLES\_SIGNED;

hwaParamCfg[paramsetIdx].source.srcConjugate = 0;

hwaParamCfg[paramsetIdx].source.srcScale = 0;

hwaParamCfg[paramsetIdx].source.bpmEnable = 0;

hwaParamCfg[paramsetIdx].source.bpmPhase = 0;

hwaParamCfg[paramsetIdx].dest.dstAddr = hwaMemDestPingOffset;

hwaParamCfg[paramsetIdx].dest.dstAcnt = dopplerFftSize - 1;

hwaParamCfg[paramsetIdx].dest.dstAIdx = numVirtualAnt \* sizeof(uint32\_t); // 16 bytes

hwaParamCfg[paramsetIdx].dest.dstBIdx = sizeof(uint32\_t); //should be dont care

hwaParamCfg[paramsetIdx].dest.dstRealComplex = 0; //same as input - complex

hwaParamCfg[paramsetIdx].dest.dstWidth = HWA\_SAMPLES\_WIDTH\_16BIT; //same as input - 16 bit

hwaParamCfg[paramsetIdx].dest.dstSign = HWA\_SAMPLES\_SIGNED; //same as input - signed

hwaParamCfg[paramsetIdx].dest.dstConjugate = 0; //no conjugate

hwaParamCfg[paramsetIdx].dest.dstScale = 8;

hwaParamCfg[paramsetIdx].dest.dstSkipInit = 0; // no skipping

hwaParamCfg[paramsetIdx].accelModeArgs.fftMode.fftEn = 1;

hwaParamCfg[paramsetIdx].accelModeArgs.fftMode.fftSize = log2Approx(dopplerFftSize);

/\* scaling is enabled in all stages except in the first one because of the

\* Hanning window scaling by half \*/

hwaParamCfg[paramsetIdx].accelModeArgs.fftMode.butterflyScaling = (dopplerFftSize - 1) >> 1;

hwaParamCfg[paramsetIdx].accelModeArgs.fftMode.interfZeroOutEn = 0; //disabled

hwaParamCfg[paramsetIdx].accelModeArgs.fftMode.windowEn = 0; //enabled

hwaParamCfg[paramsetIdx].accelModeArgs.fftMode.windowStart = 0; //start of window RAM

hwaParamCfg[paramsetIdx].accelModeArgs.fftMode.winSymm = 1; //non-symmetric - in demo do we make this symmetric

hwaParamCfg[paramsetIdx].accelModeArgs.fftMode.winInterpolateMode = 0; //fftsize is less than 1K

if(option == (uint32\_t)DATA\_PATH\_CHAIN\_SEPARATE\_LOGMAG)

{

hwaParamCfg[paramsetIdx].accelModeArgs.fftMode.magLogEn = HWA\_FFT\_MODE\_MAGNITUDE\_LOG2\_DISABLED;

}

else

{

hwaParamCfg[paramsetIdx].accelModeArgs.fftMode.magLogEn = HWA\_FFT\_MODE\_MAGNITUDE\_LOG2\_ENABLED;

}

hwaParamCfg[paramsetIdx].accelModeArgs.fftMode.fftOutMode = HWA\_FFT\_MODE\_OUTPUT\_DEFAULT;

hwaParamCfg[paramsetIdx].complexMultiply.mode = HWA\_COMPLEX\_MULTIPLY\_MODE\_DISABLE;

errCode = HWA\_configParamSet(handle, paramSetStartIdx+paramsetIdx, &hwaParamCfg[paramsetIdx], NULL);

if (errCode != 0)

{

//System\_printf("Error: HWA\_configParamSet(%d) returned %d\n", paramSetStartIdx+paramsetIdx, errCode);

MmwDemo\_debugAssert (0);

return;

}

//Doppler FFT

for (paramsetIdx = 1; paramsetIdx < numRangeBinsPerIter; paramsetIdx++)

{

hwaParamCfg[paramsetIdx] = hwaParamCfg[paramsetIdx-1];

hwaParamCfg[paramsetIdx].source.srcAddr += sizeof(uint32\_t) \* numVirtualAnt \* dopplerFftSize;

hwaParamCfg[paramsetIdx].dest.dstAddr += sizeof(uint32\_t) \* numVirtualAnt \* dopplerFftSize;

hwaParamCfg[paramsetIdx].triggerMode = HWA\_TRIG\_MODE\_IMMEDIATE;

errCode = HWA\_configParamSet(handle, paramSetStartIdx+paramsetIdx, &hwaParamCfg[paramsetIdx], NULL);

if (errCode != 0)

{

//System\_printf("Error: Doppler FFT: HWA\_configParamSet(%d) returned %d\n", paramSetStartIdx+paramsetIdx, errCode);

MmwDemo\_debugAssert (0);

return;

}

}

//log2 magnitude

memset( (void\*) &hwaParamCfg[paramsetIdx], 0, sizeof(HWA\_ParamConfig));

hwaParamCfg[paramsetIdx].triggerMode = HWA\_TRIG\_MODE\_IMMEDIATE; //Immediate following first - in demo this should be HWA\_TRIG\_MODE\_DFE

hwaParamCfg[paramsetIdx].accelMode = HWA\_ACCELMODE\_FFT; //do FFT

hwaParamCfg[paramsetIdx].source.srcAddr = hwaMemDestPingOffset; // address is relative to start of MEM0

hwaParamCfg[paramsetIdx].source.srcAcnt = numDopplerBins \* numVirtualAnt \* numRangeBinsPerIter - 1; //size in samples - 1

hwaParamCfg[paramsetIdx].source.srcAIdx = sizeof(uint32\_t); //

hwaParamCfg[paramsetIdx].source.srcBcnt = 0; //no iterations here

hwaParamCfg[paramsetIdx].source.srcBIdx = 0; //should be dont care

hwaParamCfg[paramsetIdx].source.srcShift = 0; //no shift

hwaParamCfg[paramsetIdx].source.srcCircShiftWrap = 0; //no shift

hwaParamCfg[paramsetIdx].source.srcRealComplex = 0; //complex data

hwaParamCfg[paramsetIdx].source.srcWidth = HWA\_SAMPLES\_WIDTH\_16BIT; //16-bit

hwaParamCfg[paramsetIdx].source.srcSign = HWA\_SAMPLES\_SIGNED; //signed

hwaParamCfg[paramsetIdx].source.srcConjugate = 0; //no conjugate

hwaParamCfg[paramsetIdx].source.srcScale = 0;

hwaParamCfg[paramsetIdx].source.bpmEnable = 0; //bpm removal not enabled

hwaParamCfg[paramsetIdx].source.bpmPhase = 0; //dont care

hwaParamCfg[paramsetIdx].dest.dstAddr = hwaMemSourcePingOffset; // address is relative to start of MEM0

hwaParamCfg[paramsetIdx].dest.dstAcnt = numDopplerBins \* numVirtualAnt \* numRangeBinsPerIter - 1; //this is samples - 1

hwaParamCfg[paramsetIdx].dest.dstAIdx = sizeof(uint16\_t); // 2 bytes

hwaParamCfg[paramsetIdx].dest.dstBIdx = 0; //should be dont care

hwaParamCfg[paramsetIdx].dest.dstRealComplex = 1; //same as input - complex

hwaParamCfg[paramsetIdx].dest.dstWidth = HWA\_SAMPLES\_WIDTH\_16BIT; //same as input - 16 bit

hwaParamCfg[paramsetIdx].dest.dstSign = HWA\_SAMPLES\_UNSIGNED; //same as input - signed

hwaParamCfg[paramsetIdx].dest.dstConjugate = 0; //no conjugate

hwaParamCfg[paramsetIdx].dest.dstScale = 0;

hwaParamCfg[paramsetIdx].dest.dstSkipInit = 0; // no skipping

hwaParamCfg[paramsetIdx].accelModeArgs.fftMode.fftEn = 0;

hwaParamCfg[paramsetIdx].accelModeArgs.fftMode.fftSize = 1;//TODO remove later after driver corrects

if(option == (uint32\_t)DATA\_PATH\_CHAIN\_SEPARATE\_LOGMAG)

{

hwaParamCfg[paramsetIdx].accelModeArgs.fftMode.magLogEn = HWA\_FFT\_MODE\_MAGNITUDE\_LOG2\_ENABLED;

}

else

{

hwaParamCfg[paramsetIdx].accelModeArgs.fftMode.magLogEn = HWA\_FFT\_MODE\_MAGNITUDE\_LOG2\_DISABLED;

}

errCode = HWA\_configParamSet(handle, paramSetStartIdx+paramsetIdx, &hwaParamCfg[paramsetIdx], NULL);

if (errCode != 0)

{

//System\_printf("Error: Log2 HWA\_configParamSet(%d) returned %d \n", paramSetStartIdx+paramsetIdx, errCode);

//System\_printf("numDopplerBins %d numVirtualAnt %d numRangeBinsPerIter %d\n", numDopplerBins,numVirtualAnt, numRangeBinsPerIter);

MmwDemo\_debugAssert (0);

return;

}

paramsetIdx++;

//Sum of magnitudes

memset( (void\*) &hwaParamCfg[paramsetIdx], 0, sizeof(HWA\_ParamConfig));

hwaParamCfg[paramsetIdx].triggerMode = HWA\_TRIG\_MODE\_IMMEDIATE; //Immediate

hwaParamCfg[paramsetIdx].accelMode = HWA\_ACCELMODE\_FFT; //do FFT

hwaParamCfg[paramsetIdx].source.srcAddr = hwaMemSourcePingOffset; // address is relative to start of MEM0

hwaParamCfg[paramsetIdx].source.srcAcnt = numVirtualAnt-1; //size in samples - 1

hwaParamCfg[paramsetIdx].source.srcAIdx = sizeof(uint16\_t); //

hwaParamCfg[paramsetIdx].source.srcBcnt = numDopplerBins \* numRangeBinsPerIter - 1; //no iterations here

hwaParamCfg[paramsetIdx].source.srcBIdx = numVirtualAnt \* sizeof(uint16\_t); //should be dont care

hwaParamCfg[paramsetIdx].source.srcShift = 0; //no shift

hwaParamCfg[paramsetIdx].source.srcCircShiftWrap = 0; //no shift

hwaParamCfg[paramsetIdx].source.srcRealComplex = 1; //real data

hwaParamCfg[paramsetIdx].source.srcWidth = HWA\_SAMPLES\_WIDTH\_16BIT; //16-bit

hwaParamCfg[paramsetIdx].source.srcSign = HWA\_SAMPLES\_UNSIGNED; //signed

hwaParamCfg[paramsetIdx].source.srcConjugate = 0; //no conjugate

hwaParamCfg[paramsetIdx].source.srcScale = 2;

hwaParamCfg[paramsetIdx].source.bpmEnable = 0; //bpm removal not enabled

hwaParamCfg[paramsetIdx].source.bpmPhase = 0; //dont care

hwaParamCfg[paramsetIdx].dest.dstAddr = (uint16\_t) (hwaMemDestPingOffset + numDopplerBins \*

numVirtualAnt \* numRangeBinsPerIter \* sizeof(uint32\_t)); // address is relative to start of MEM0

hwaParamCfg[paramsetIdx].dest.dstAcnt = 1 - 1; //this is samples - 1

hwaParamCfg[paramsetIdx].dest.dstAIdx = sizeof(uint16\_t); // 16 bytes

hwaParamCfg[paramsetIdx].dest.dstBIdx = sizeof(uint16\_t); //should be dont care

hwaParamCfg[paramsetIdx].dest.dstRealComplex = 1; //same as input - complex

hwaParamCfg[paramsetIdx].dest.dstWidth = HWA\_SAMPLES\_WIDTH\_16BIT; //same as input - 16 bit

hwaParamCfg[paramsetIdx].dest.dstSign = HWA\_SAMPLES\_UNSIGNED; //same as input - signed

hwaParamCfg[paramsetIdx].dest.dstConjugate = 0; //no conjugate

hwaParamCfg[paramsetIdx].dest.dstScale = 8;

hwaParamCfg[paramsetIdx].dest.dstSkipInit = 0; // no skipping

if(numVirtualAnt == 1)

{

/\*If number of virtual antennas is 1, do not use FFT to compute sum magnitude.

There is nothing to sum and azimuth can not be estimated.\*/

hwaParamCfg[paramsetIdx].accelModeArgs.fftMode.fftEn = 0;

hwaParamCfg[paramsetIdx].accelModeArgs.fftMode.fftSize = 1;

}

else

{

hwaParamCfg[paramsetIdx].accelModeArgs.fftMode.fftEn = 1;

hwaParamCfg[paramsetIdx].accelModeArgs.fftMode.fftSize = log2Approx(numVirtualAnt);

}

hwaParamCfg[paramsetIdx].accelModeArgs.fftMode.butterflyScaling = 0x3FF; //LSB fftSize bits are relevant - revisit this for all FFT size and data size

hwaParamCfg[paramsetIdx].accelModeArgs.fftMode.interfZeroOutEn = 0; //disabled

hwaParamCfg[paramsetIdx].accelModeArgs.fftMode.windowEn = 0; //enabled

hwaParamCfg[paramsetIdx].accelModeArgs.fftMode.windowStart = 0; //start of window RAM

hwaParamCfg[paramsetIdx].accelModeArgs.fftMode.winSymm = 1; //non-symmetric - in demo do we make this symmetric

hwaParamCfg[paramsetIdx].accelModeArgs.fftMode.winInterpolateMode = 0; //fftsize is less than 1K

hwaParamCfg[paramsetIdx].accelModeArgs.fftMode.magLogEn = HWA\_FFT\_MODE\_MAGNITUDE\_LOG2\_DISABLED; //disabled

hwaParamCfg[paramsetIdx].accelModeArgs.fftMode.fftOutMode = HWA\_FFT\_MODE\_OUTPUT\_DEFAULT; // output FFT samples

hwaParamCfg[paramsetIdx].complexMultiply.mode = HWA\_COMPLEX\_MULTIPLY\_MODE\_DISABLE;

errCode = HWA\_configParamSet(handle, paramSetStartIdx+paramsetIdx, &hwaParamCfg[paramsetIdx], NULL);

if (errCode != 0)

{

//System\_printf("Error: Sum of mag HWA\_configParamSet(%d) returned %d\n", paramSetStartIdx+paramsetIdx, errCode);

MmwDemo\_debugAssert (0);

return;

}

/\* Enable the DMA hookup to this paramset so that data gets copied out \*/

paramISRConfig.interruptTypeFlag = HWA\_PARAMDONE\_INTERRUPT\_TYPE\_DMA;

paramISRConfig.dma.dstChannel = dmaDestChannelPing; //EDMA channel to trigger to copy the data out

paramISRConfig.cpu.callbackArg = NULL;

errCode = HWA\_enableParamSetInterrupt(handle, paramSetStartIdx+paramsetIdx, &paramISRConfig);

if (errCode != 0)

{

//System\_printf("Error: HWA\_enableParamSetInterrupt(PING DMA) returned %d\n",errCode);

MmwDemo\_debugAssert (0);

return;

}

//programming HWACC for the pong buffer

for (k=0; k < numRangeBinsPerIter + 2; k++)

{

paramsetIdx = k + numRangeBinsPerIter + 2;

hwaParamCfg[paramsetIdx] = hwaParamCfg[k];

hwaParamCfg[paramsetIdx].source.srcAddr += (hwaMemSourcePongOffset - hwaMemSourcePingOffset);

hwaParamCfg[paramsetIdx].dest.dstAddr += (hwaMemDestPongOffset - hwaMemDestPingOffset);//#def??

if (k == 0)

{

hwaParamCfg[paramsetIdx].dmaTriggerSrc = dmaTriggerSourcePong;

}

errCode = HWA\_configParamSet(handle, paramSetStartIdx + paramsetIdx, &hwaParamCfg[paramsetIdx], NULL);

if (errCode != 0)

{

//System\_printf("Error: HWA\_configParamSet(%d) returned %d\n", paramSetStartIdx + paramsetIdx, errCode);

MmwDemo\_debugAssert (0);

return;

}

}

/\* Enable the DMA hookup to the last paramset \*/

paramISRConfig.interruptTypeFlag = HWA\_PARAMDONE\_INTERRUPT\_TYPE\_DMA;

paramISRConfig.dma.dstChannel = dmaDestChannelPong; //EDMA channel to trigger to copy the data out

paramISRConfig.cpu.callbackArg = NULL;

errCode = HWA\_enableParamSetInterrupt(handle, paramSetStartIdx + 2\*(numRangeBinsPerIter+2)-1, &paramISRConfig);

if (errCode != 0)

{

//System\_printf("Error: HWA\_enableParamSetInterrupt(PONG DMA) returned %d\n",errCode);

MmwDemo\_debugAssert (0);

return;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* CONFIG WINDOW RAM \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* if windowing is enabled, load the window coefficients in RAM \*/

//errCode = HWA\_configRam(handle, HWA\_RAM\_TYPE\_WINDOW\_RAM, (uint8\_t \*)win\_data16, sizeof(win\_data16), windowOffsetBytes);

}

void HWAutil\_configDopplerFFTSingleRangeBin(HWA\_Handle handle,

uint32\_t paramSetStartIdx,

uint32\_t dopplerFftSize,

uint8\_t numVirtualAnt,

uint32\_t windowOffsetBytes,

uint8\_t dmaTriggerSource,

uint16\_t hwaMemAzimSource,

uint16\_t hwaMemAzimDest)

{

HWA\_ParamConfig hwaParamCfg;

int32\_t errCode = 0;

memset( (void\*) &hwaParamCfg, 0, sizeof(HWA\_ParamConfig));

hwaParamCfg.dmaTriggerSrc = dmaTriggerSource;

hwaParamCfg.accelMode = HWA\_ACCELMODE\_FFT; //do FFT

hwaParamCfg.source.srcAddr = hwaMemAzimSource;

hwaParamCfg.source.srcAcnt = dopplerFftSize - 1; //size in samples - 1

hwaParamCfg.source.srcAIdx = numVirtualAnt \* sizeof(uint32\_t); //

hwaParamCfg.source.srcBcnt = numVirtualAnt -1; //no iterations here

hwaParamCfg.source.srcBIdx = sizeof(uint32\_t);

hwaParamCfg.source.srcShift = 0; //no shift

hwaParamCfg.source.srcCircShiftWrap = 0; //no shift

hwaParamCfg.source.srcRealComplex = HWA\_SAMPLES\_FORMAT\_COMPLEX; //complex data

hwaParamCfg.source.srcWidth = HWA\_SAMPLES\_WIDTH\_16BIT; //16-bit

hwaParamCfg.source.srcSign = HWA\_SAMPLES\_SIGNED; //signed

hwaParamCfg.source.srcConjugate = 0; //no conjugate

hwaParamCfg.source.srcScale = 0;

hwaParamCfg.source.bpmEnable = 0; //bpm removal not enabled

hwaParamCfg.source.bpmPhase = 0; //dont care

hwaParamCfg.dest.dstAddr = hwaMemAzimDest; // address is relative to start of MEM0

hwaParamCfg.dest.dstAcnt = dopplerFftSize - 1; //this is samples - 1

hwaParamCfg.dest.dstAIdx = numVirtualAnt \* sizeof(uint32\_t); //

hwaParamCfg.dest.dstBIdx = sizeof(uint32\_t) ; //should be dont care

hwaParamCfg.dest.dstRealComplex = HWA\_SAMPLES\_FORMAT\_COMPLEX; //same as input - complex

hwaParamCfg.dest.dstWidth = HWA\_SAMPLES\_WIDTH\_16BIT; //same as input - 16 bit

hwaParamCfg.dest.dstSign = HWA\_SAMPLES\_SIGNED; //same as input - signed

hwaParamCfg.dest.dstConjugate = 0; //no conjugate

hwaParamCfg.dest.dstScale = 8 ;

hwaParamCfg.dest.dstSkipInit = 0; // no skipping

hwaParamCfg.accelModeArgs.fftMode.fftEn = 1;

hwaParamCfg.accelModeArgs.fftMode.fftSize = log2Approx(dopplerFftSize);

hwaParamCfg.accelModeArgs.fftMode.butterflyScaling = 0x3FF; //LSB fftSize bits are relevant - revisit this for all FFT size and data size

hwaParamCfg.accelModeArgs.fftMode.interfZeroOutEn = 0; //disabled

hwaParamCfg.accelModeArgs.fftMode.windowEn = 1; //enabled

hwaParamCfg.accelModeArgs.fftMode.windowStart = windowOffsetBytes; //start of window RAM

hwaParamCfg.accelModeArgs.fftMode.winSymm = 1; //non-symmetric - in demo do we make this symmetric

hwaParamCfg.accelModeArgs.fftMode.winInterpolateMode = 0; //fftsize is less than 1K

hwaParamCfg.accelModeArgs.fftMode.magLogEn = HWA\_FFT\_MODE\_MAGNITUDE\_LOG2\_DISABLED;//HWA\_FFT\_MODE\_MAGNITUDE\_LOG2\_DISABLED; //disabled

hwaParamCfg.accelModeArgs.fftMode.fftOutMode = HWA\_FFT\_MODE\_OUTPUT\_DEFAULT; // output FFT samples

hwaParamCfg.complexMultiply.mode = HWA\_COMPLEX\_MULTIPLY\_MODE\_DISABLE;

errCode = HWA\_configParamSet(handle, paramSetStartIdx, &hwaParamCfg, NULL);

if (errCode != 0)

{

System\_printf("Error: HWA\_configParamSet(%d) returned %d\n", paramSetStartIdx, errCode);

return;

}

}

void HWAutil\_configCFAR(HWA\_Handle handle,

uint32\_t paramSetStartIdx,

uint32\_t numRangeBins,

uint32\_t numDopplerBins,

uint32\_t winLen,

uint32\_t guardLen,

uint32\_t noiseDivRightShift,

uint8\_t peakGrouping,

uint8\_t cyclicMode,

uint8\_t nAvgMode,

uint16\_t detObjectListSize,

uint8\_t dmaTriggerSource,

uint8\_t dmaDestChannel,

uint16\_t hwaSourceBufOffset,

uint16\_t hwaDestBufOffset

)

{

HWA\_ParamConfig hwaParamCfg;

HWA\_InterruptConfig paramISRConfig;

int32\_t errCode = 0;

memset( (void\*) &hwaParamCfg, 0, sizeof(hwaParamCfg));

hwaParamCfg.triggerMode = HWA\_TRIG\_MODE\_DMA;

hwaParamCfg.dmaTriggerSrc = dmaTriggerSource;

hwaParamCfg.accelModeArgs.cfarMode.peakGroupEn = peakGrouping;

hwaParamCfg.accelMode = HWA\_ACCELMODE\_CFAR;

hwaParamCfg.accelModeArgs.cfarMode.operMode = HWA\_CFAR\_OPER\_MODE\_LOG\_INPUT\_REAL;

hwaParamCfg.source.srcAddr = hwaSourceBufOffset;

hwaParamCfg.source.srcAcnt = numRangeBins-1;

hwaParamCfg.source.srcRealComplex = HWA\_SAMPLES\_FORMAT\_REAL;

hwaParamCfg.source.srcAIdx = numDopplerBins\*2;

hwaParamCfg.source.srcBIdx = sizeof(uint16\_t);

hwaParamCfg.source.srcBcnt = numDopplerBins-1;

hwaParamCfg.source.srcScale = 8;

hwaParamCfg.dest.dstAddr = hwaDestBufOffset;

hwaParamCfg.dest.dstAcnt = detObjectListSize - 1;

hwaParamCfg.dest.dstRealComplex = HWA\_SAMPLES\_FORMAT\_COMPLEX;

hwaParamCfg.dest.dstWidth = HWA\_SAMPLES\_WIDTH\_32BIT;

hwaParamCfg.dest.dstAIdx = 8;

hwaParamCfg.dest.dstBIdx = 4096;

hwaParamCfg.dest.dstScale = 8;

hwaParamCfg.accelModeArgs.cfarMode.numGuardCells = guardLen;

hwaParamCfg.accelModeArgs.cfarMode.nAvgDivFactor = noiseDivRightShift;

hwaParamCfg.accelModeArgs.cfarMode.cyclicModeEn = cyclicMode;

hwaParamCfg.accelModeArgs.cfarMode.nAvgMode = nAvgMode;

hwaParamCfg.accelModeArgs.cfarMode.numNoiseSamplesRight = winLen >> 1;

hwaParamCfg.accelModeArgs.cfarMode.numNoiseSamplesLeft = winLen >> 1;

hwaParamCfg.accelModeArgs.cfarMode.outputMode = HWA\_CFAR\_OUTPUT\_MODE\_I\_PEAK\_IDX\_Q\_NEIGHBOR\_NOISE\_VAL;

errCode = HWA\_configParamSet(handle, paramSetStartIdx, &hwaParamCfg, NULL);

if (errCode != 0)

{

//System\_printf("Error: HWA\_configParamSet(%d) returned %d\n", paramSetStartIdx, errCode);

MmwDemo\_debugAssert (0);

return;

}

/\* Enable the DMA hookup to the last paramset \*/

paramISRConfig.interruptTypeFlag = HWA\_PARAMDONE\_INTERRUPT\_TYPE\_DMA;

paramISRConfig.dma.dstChannel = dmaDestChannel; //EDMA channel to trigger to copy the data out

paramISRConfig.cpu.callbackArg = NULL;

errCode = HWA\_enableParamSetInterrupt(handle, paramSetStartIdx, &paramISRConfig);

if (errCode != 0)

{

//System\_printf("Error: HWA\_enableParamSetInterrupt(PONG DMA) returned %d\n",errCode);

MmwDemo\_debugAssert (0);

return;

}

}

void HWAutil\_configAngleEstAzimuth(HWA\_Handle handle,

uint32\_t paramSetStartIdx,

uint32\_t numVirtualAnt,

uint32\_t fftOutSize,

uint32\_t numIter,

uint16\_t hwaSourceBufOffset,

uint16\_t hwaDestBufOffset)

{

HWA\_ParamConfig hwaParamCfg;

int32\_t errCode = 0;

memset( (void\*) &hwaParamCfg, 0, sizeof(hwaParamCfg));

hwaParamCfg.triggerMode = HWA\_TRIG\_MODE\_IMMEDIATE;

hwaParamCfg.source.srcAddr = hwaSourceBufOffset;

hwaParamCfg.source.srcAcnt = numVirtualAnt-1;

hwaParamCfg.source.srcAIdx = sizeof(uint32\_t);

hwaParamCfg.source.srcBIdx = numVirtualAnt \* sizeof(uint32\_t);

hwaParamCfg.source.srcBcnt = numIter-1;

hwaParamCfg.source.srcSign = HWA\_SAMPLES\_SIGNED;

hwaParamCfg.source.srcScale = 8;

hwaParamCfg.complexMultiply.mode = HWA\_COMPLEX\_MULTIPLY\_MODE\_DISABLE;

hwaParamCfg.accelModeArgs.fftMode.fftEn = 1;

hwaParamCfg.accelModeArgs.fftMode.magLogEn = HWA\_FFT\_MODE\_MAGNITUDE\_ONLY\_ENABLED;

hwaParamCfg.accelModeArgs.fftMode.fftSize = log2Approx(fftOutSize);//assumes power of 2;

hwaParamCfg.accelModeArgs.fftMode.windowEn = 0;

hwaParamCfg.accelModeArgs.fftMode.winSymm = 1;

hwaParamCfg.accelModeArgs.fftMode.windowStart = 512; //do not care

hwaParamCfg.accelModeArgs.fftMode.butterflyScaling = 0; //no scaling

hwaParamCfg.dest.dstAddr = hwaDestBufOffset;

hwaParamCfg.dest.dstAcnt = fftOutSize-1;

hwaParamCfg.dest.dstAIdx = sizeof(uint16\_t);//abs

hwaParamCfg.dest.dstBIdx = fftOutSize \* sizeof(uint16\_t);

hwaParamCfg.dest.dstSign = HWA\_SAMPLES\_UNSIGNED;

hwaParamCfg.dest.dstScale = 3;

hwaParamCfg.dest.dstRealComplex = HWA\_SAMPLES\_FORMAT\_REAL;

errCode = HWA\_configParamSet(handle, paramSetStartIdx, &hwaParamCfg, NULL);

if (errCode != 0)

{

MmwDemo\_debugAssert (0);

return;

}

}

void HWAutil\_configAngleEstAzimuthElevation(HWA\_Handle handle,

uint32\_t paramSetStartIdx,

uint32\_t numVirtualAntAzim,

uint32\_t numVirtualAntElev,

uint32\_t fftOutSize,

uint32\_t numIter,

uint16\_t hwaSourceAzimBufOffset,

uint32\_t hwaSourceElevBufOffset,

uint16\_t hwaDestAzimAbsBufOffset,

uint16\_t hwaDestAzimCplxBufOffset,

uint16\_t hwaDestElevCplxBufOffset)

{

HWA\_ParamConfig hwaParamCfg;

int32\_t errCode = 0;

memset( (void\*) &hwaParamCfg, 0, sizeof(hwaParamCfg));

hwaParamCfg.triggerMode = HWA\_TRIG\_MODE\_IMMEDIATE;

hwaParamCfg.source.srcAddr = (uint16\_t) hwaSourceElevBufOffset;

hwaParamCfg.source.srcAcnt = numVirtualAntElev-1;

hwaParamCfg.source.srcAIdx = sizeof(uint32\_t);

hwaParamCfg.source.srcBIdx = numVirtualAntElev \* sizeof(uint32\_t);

hwaParamCfg.source.srcBcnt = numIter-1;

hwaParamCfg.source.srcSign = HWA\_SAMPLES\_SIGNED;

hwaParamCfg.source.srcScale = 8;

hwaParamCfg.complexMultiply.mode = HWA\_COMPLEX\_MULTIPLY\_MODE\_DISABLE;

hwaParamCfg.accelModeArgs.fftMode.fftEn = 1;

hwaParamCfg.accelModeArgs.fftMode.fftSize = log2Approx(fftOutSize);

hwaParamCfg.accelModeArgs.fftMode.windowEn = 0;

hwaParamCfg.accelModeArgs.fftMode.butterflyScaling = 0;

hwaParamCfg.dest.dstAddr = (uint16\_t) hwaDestElevCplxBufOffset;

hwaParamCfg.dest.dstAcnt = fftOutSize-1;

hwaParamCfg.dest.dstAIdx = sizeof(uint32\_t);

hwaParamCfg.dest.dstBIdx = fftOutSize \* sizeof(uint32\_t);

hwaParamCfg.dest.dstSign = HWA\_SAMPLES\_SIGNED;

hwaParamCfg.dest.dstScale = 3;

errCode = HWA\_configParamSet(handle, paramSetStartIdx, &hwaParamCfg, NULL);

if (errCode != 0)

{

MmwDemo\_debugAssert (0);

return;

}

memset( (void\*) &hwaParamCfg, 0, sizeof(hwaParamCfg));

hwaParamCfg.triggerMode = HWA\_TRIG\_MODE\_IMMEDIATE;

hwaParamCfg.source.srcAddr = (uint16\_t) hwaSourceAzimBufOffset;

hwaParamCfg.source.srcAcnt = numVirtualAntAzim-1;

hwaParamCfg.source.srcAIdx = sizeof(uint32\_t);

hwaParamCfg.source.srcBIdx = numVirtualAntAzim \* sizeof(uint32\_t);

hwaParamCfg.source.srcBcnt = numIter-1;

hwaParamCfg.source.srcSign = HWA\_SAMPLES\_SIGNED;

hwaParamCfg.source.srcScale = 8;

hwaParamCfg.complexMultiply.mode = HWA\_COMPLEX\_MULTIPLY\_MODE\_DISABLE;

hwaParamCfg.accelModeArgs.fftMode.fftEn = 1;

hwaParamCfg.accelModeArgs.fftMode.magLogEn = HWA\_FFT\_MODE\_MAGNITUDE\_ONLY\_ENABLED;

hwaParamCfg.accelModeArgs.fftMode.fftSize = log2Approx(fftOutSize);

hwaParamCfg.accelModeArgs.fftMode.windowEn = 0;

hwaParamCfg.accelModeArgs.fftMode.butterflyScaling = 0;

hwaParamCfg.dest.dstAddr = (uint16\_t) hwaDestAzimAbsBufOffset;

hwaParamCfg.dest.dstAcnt = fftOutSize-1;

hwaParamCfg.dest.dstAIdx = sizeof(uint16\_t);//abs

hwaParamCfg.dest.dstBIdx = fftOutSize \* sizeof(uint16\_t);

hwaParamCfg.dest.dstSign = HWA\_SAMPLES\_UNSIGNED;

hwaParamCfg.dest.dstScale = 3;

hwaParamCfg.dest.dstRealComplex = HWA\_SAMPLES\_FORMAT\_REAL;

errCode = HWA\_configParamSet(handle, paramSetStartIdx + 1, &hwaParamCfg, NULL);

if (errCode != 0)

{

MmwDemo\_debugAssert (0);

return;

}

memset( (void\*) &hwaParamCfg, 0, sizeof(hwaParamCfg));

hwaParamCfg.triggerMode = HWA\_TRIG\_MODE\_IMMEDIATE;

hwaParamCfg.source.srcAddr = (uint16\_t) hwaSourceAzimBufOffset;

hwaParamCfg.source.srcAcnt = numVirtualAntAzim-1;

hwaParamCfg.source.srcAIdx = sizeof(uint32\_t);

hwaParamCfg.source.srcBIdx = numVirtualAntAzim \* sizeof(uint32\_t);

hwaParamCfg.source.srcBcnt = numIter-1;

hwaParamCfg.source.srcSign = HWA\_SAMPLES\_SIGNED;

hwaParamCfg.source.srcScale = 8;

hwaParamCfg.complexMultiply.mode = HWA\_COMPLEX\_MULTIPLY\_MODE\_DISABLE;

hwaParamCfg.accelModeArgs.fftMode.fftEn = 1;

hwaParamCfg.accelModeArgs.fftMode.fftSize = log2Approx(fftOutSize);

hwaParamCfg.accelModeArgs.fftMode.windowEn = 0;

hwaParamCfg.accelModeArgs.fftMode.butterflyScaling = 0;

hwaParamCfg.dest.dstAddr = (uint16\_t) hwaDestAzimCplxBufOffset;

hwaParamCfg.dest.dstAcnt = fftOutSize-1;

hwaParamCfg.dest.dstAIdx = sizeof(uint32\_t);

hwaParamCfg.dest.dstBIdx = fftOutSize \* sizeof(uint32\_t);

hwaParamCfg.dest.dstSign = HWA\_SAMPLES\_SIGNED;

hwaParamCfg.dest.dstScale = 3;

errCode = HWA\_configParamSet(handle, paramSetStartIdx + 2, &hwaParamCfg, NULL);

if (errCode != 0)

{

MmwDemo\_debugAssert (0);

return;

}

}

1. Source Code for 3d Stitching of Point Cloud Data
2. Source Code for Tachometer
3. Source Code for

…

REFERENCES

[1] D. Vantrease, R. Schreiber, M. Monchiero, M. McLaren, N. P. Jouppi, M. Fiorentino*, et al.*, "Corona: System Implications of Emerging Nanophotonic Technology," in *Computer Architecture, 2008. ISCA '08. 35th International Symposium on*, 2008, pp. 153-164.

[2] X. Zhang and A. Louri, "A multilayer nanophotonic interconnection network for on-chip many-core communications," in *Design Automation Conference (DAC), 2010 47th ACM/IEEE*, 2010, pp. 156-161.

[3] C. Batten, A. Joshi, J. Orcutt, A. Khilo, B. Moss, C. Holzwarth*, et al.*, "Building manycore processor-to-DRAM networks with monolithic silicon photonics," in *High Performance Interconnects, 2008. HOTI '08. 16th IEEE Symposium on*, 2008, pp. 21-30.

[4] Y. Pan, P. Kumar, J. Kim, G. Memik, Y. Zhang, and A. Choudhary, "Firefly: illuminating future network-on-chip with nanophotonics," in *IEEE/ACM Intl. Symp. on Computer Architecture (ISCA)*, 2009, pp. 429-440.

[5] N. Kirman, M. Kirman, R. K. Dokania, J. F. Martinez, A. B. Apsel, M. A. Watkins*, et al.*, "Leveraging Optical Technology in Future Bus-based Chip Multiprocessors," in *Microarchitecture, 2006. MICRO-39. 39th Annual IEEE/ACM International Symposium on*, 2006, pp. 492-503.

[6] J. M. Cianchetti, C. J. Kerekes, and H. D. Albonesi, "Phastlane: a rapid transit optical routing network," in *Proceeding of: 36th International Symposium on Computer Architecture (ISCA)*, 2009, pp. 441-450.

[7] A. Shacham, K. Bergman, and L. P. Carloni, "Photonic Networks-on-Chip for Future Generations of Chip Multiprocessors," *Computers, IEEE Transactions on,* vol. 57, pp. 1246-1260, 2008.

[8] A. Shacham, K. Bergman, and L. P. Carloni, "On the Design of a Photonic Network-on-Chip," in *First International Symposium on Networks-on-Chip, 2007. NOCS 2007*, 2007, pp. 53-64.

[9] M. Kwai Hung, Y. Yaoyao, W. Xiaowen, Z. Wei, L. Weichen, and X. Jiang, "A Hierarchical Hybrid Optical-Electronic Network-on-Chip," in *VLSI (ISVLSI), 2010 IEEE Computer Society Annual Symposium on*, 2010, pp. 327-332.

[10] D. Ding and D. Z. Pan, "OIL: a nano-photonics optical interconnect library for a new photonic networks-on-chip architecture," presented at the Proceedings of the 11th international workshop on System level interconnect prediction, San Francisco, CA, USA, 2009.

[11] A. Joshi, C. Batten, K. Yong-Jin, S. Beamer, I. Shamim, K. Asanovic*, et al.*, "Silicon-photonic clos networks for global on-chip communication," in *Networks-on-Chip, 2009. NoCS 2009. 3rd ACM/IEEE International Symposium on*, 2009, pp. 124-133.

[12] D. Vantrease, R. Schreiber, M. Monchiero, M. McLaren, N. P. Jouppi, M. Fiorentino*, et al.*, "Corona: system implications of emerging nanophotonic technology," in *Proc. 35th IEEE/ACM Int'l Symp. Computer Architecture (ISCA)*, 2008, pp. 153-164.

[13] L. Zhang, E. Regentova, and X. Tan, "A 2D-Torus Based Packet Switching Optical Network-on-Chip Architecture," presented at the *IEEE International Symposium on Photonics and Optoelectronics* (SOPO 2011), Wuhan, China, 2011.

[14] L. Zhang, E. E. Regentova, and X. Tan, "Packet switching optical network-on-chip architectures," *Comput. Electr. Eng.,* vol. 39, pp. 697-714, 2013.

[15] G. Huaxi, X. Jiang, and W. Zheng, "A novel optical mesh network-on-chip for gigascale systems-on-chip," in *Circuits and Systems, 2008. APCCAS 2008. IEEE Asia Pacific Conference on*, 2008, pp. 1728-1731.

[16] G. Huaxi, X. Jiang, and Z. Wei, "A low-power fat tree-based optical Network-On-Chip for multiprocessor system-on-chip," in *Design, Automation & Test in Europe Conference & Exhibition, 2009. DATE '09.*, 2009, pp. 3-8.

[17] Y. Yaoyao, X. Jiang, H. Baihan, W. Xiaowen, Z. Wei, W. Xuan*, et al.*, "3-D Mesh-Based Optical Network-on-Chip for Multiprocessor System-on-Chip," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on,* vol. 32, pp. 584-596, 2013.

[18] A. Shacham, K. Bergman, and L. P. Carloni, "Photonic networks-on-chip for future generations of chip multiprocessors," *IEEE Trans. Computers,* vol. 57, pp. 1246-1260, 2008.

[19] A. W. Poon, F. X. Xu, and X. Luo, "Cascaded active silicon microresonator array cross-connect circuits for WDM networks-on-chip," in *Proc. SPIE*, 2008, p. 689812.

[20] M. Lipson, "Compact Electro-Optic Modulators on a Silicon Chip," *IEEE Journal of Selected Topics in Quantum Electronics,* vol. 12, pp. 1520-1526, 2006.

[21] M. Lipson, "Guiding, modulating, and emitting light on Silicon-challenges and opportunities," *Lightwave Technology, Journal of,* vol. 23, pp. 4222-4238, 2005.

[22] T. Xianfang, Y. Mei, Z. Lei, J. Yingtao, and Y. Jianyi, "Wavelength-routed optical networks-on-chip built with comb switches," in *Photonics Conference (IPC), 2013 IEEE*, 2013, pp. 46-47.