**DAS - Software Requirements Specifications**

Status: Draft

**Revision History**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Rev** | **Status** | **Date** | **Author** | **Reason for change/description** |
|  | D | 7/2/2019 | Sandeep | Initial version |
|  | D | 7/7/2019 | Sandeep | Updated sections 2.1 & 2.2 |
|  | D | 7/17/2019 | Sandeep | Added sections 3 & 4  Updated section 2 |
|  | D | 7/21/2019 | Sandeep | Added sections 4 & 5  References section 6 (it was 4) |
|  | D | 7/23/2019 | Sandeep | Updated section 3 & 4.1.1 |

Statuses include: D: Draft, P: Pending review/frozen, U: Update needed, A: Approved

**Approved by: Date:**

**Table of Contents**

List of Tables 5

List of Figures 5

1 Introduction 6

1.1 Purpose 6

1.2 Scope 6

1.3 Abbreviations and Acronyms 6

1.4 References 6

2 System Block Diagram 7

2.1 System Description 7

2.1.1 Overview 7

2.1.2 High level requirements 7

2.2 Packet Format 10

2.2.1 Device Config Packet 10

2.2.2 Acknowledgment Packet 10

2.2.3 Raw Data Packet 11

2.2.4 Status Request 11

2.2.5 Status Response 11

2.2.6 Log Packet 12

2.3 Firmware Upgrade 13

2.3.1 Upgrade 13

2.4 Operating System 13

2.5 Debug 13

3 Timing Requirements 14

3.1 Select top and bottom multiplexer 14

3.2 Set the DAC with fence voltage 14

3.3 Select row using row multiplexer 14

3.4 Set the DAC for excitation voltage 14

3.5 Setup ADC and read 14

3.6 FPGA to read ADC value 15

3.7 FPGA to store the read values to memory 15

4 FPGA Architecture 16

4.1 Interfaces 16

4.1.1 FPGA to ADC 16

4.1.2 FPGA to DAC 16

4.1.3 FPGA to CPU 16

5 CPU Architecture 17

5.1 DDR2 SDRAM 17

5.2 NAND Flash 17

5.2.1 Boot NAND 17

5.2.2 Data NAND 17

5.3 Micro SD Card 17

5.4 Wi-Fi Module 17

5.5 LAN Module 17

5.6 LEDs 17

5.7 FPGA Memory 17

5.8 Software Flow Diagram 18

5.8.1 High level flow diagram 18

5.8.2 SW Tasks 19

6 References 20

6.1 Datasheets 20

List of Tables

**No table of figures entries found.**

**­**

List of Figures

**No table of figures entries found.**

# Introduction

## Purpose

Describes Lenexa Software architecture details.

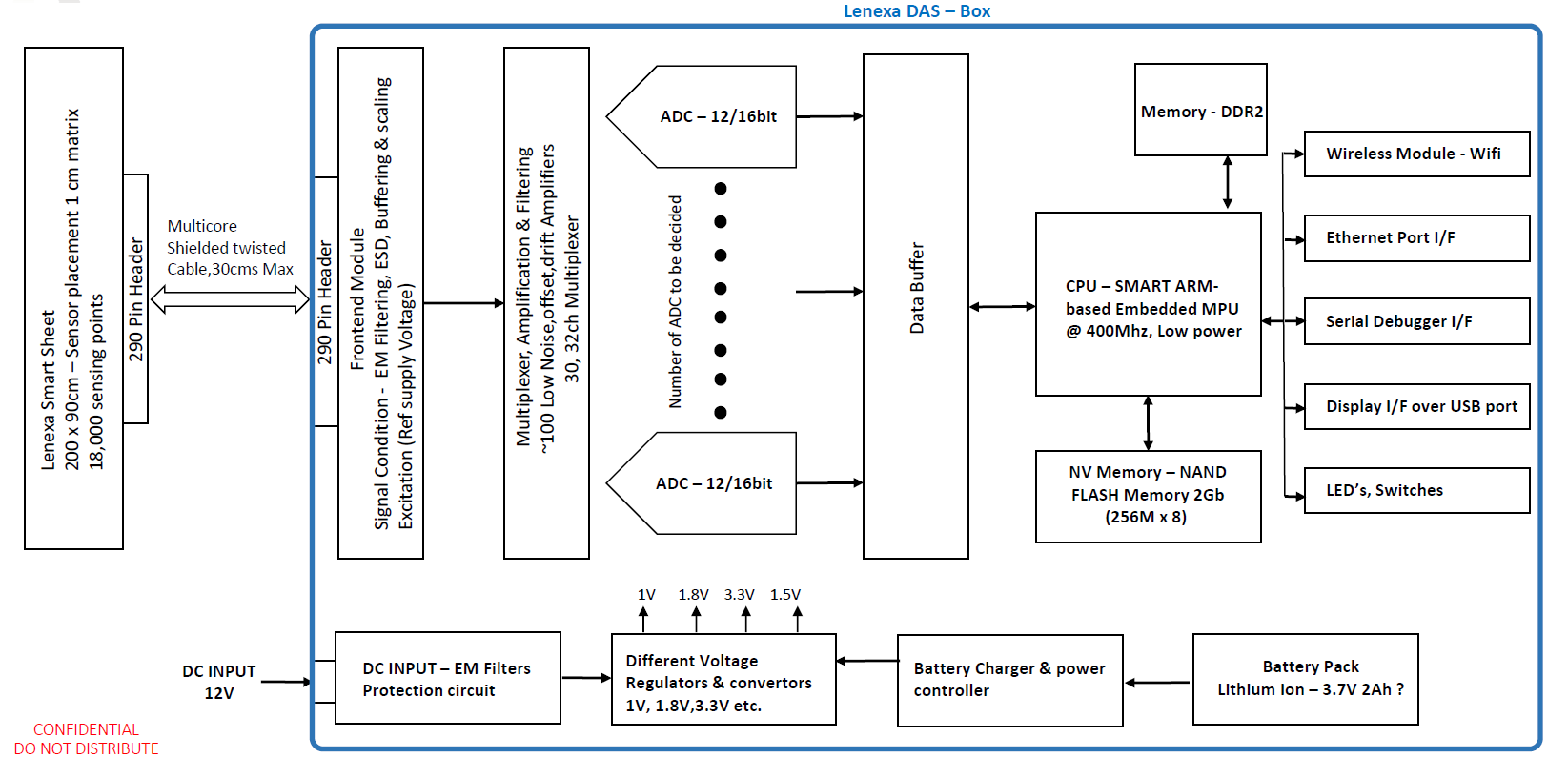
## Scope

## Abbreviations and Acronyms

|  |  |
| --- | --- |
| **Abbreviation/Acronym** | **Definition** |
| SW | Software |
| HW | Hardware |

## References

# System Block Diagram



## System Description

### Overview

System consists of various HW blocks viz, ADCs, MUXes, CPU, RAM, Flash, etc.

Software that will be designed, is responsible for controlling all these blocks to meet the system requirements.

### High level requirements

#### Design Assumptions

* Only R is calculated
* 200x100 points to be computed
* Fence voltage are on columns
* Row side is excitation point
* Every column node is digitized at rate of 30Hz

#### Timing Requirements

* Sampling rate is 30Hz. Which means HW should be able to collect data from 20,000 (200\*100) points within in 33.3ms. Which comes to 1.6 us for each sample point.
* Sampled data points to be sent to server at 30Hz
* Apart from above 2, there may be more activities to be done within 30ms.
  + Filtering
  + Debug
* Challenges
  + Can we meet the timing?
    - As per current assessment we can’t meet 30Hz timing.
    - Further assessment/changes may be needed for HW parts

#### Memory

* Considering 16bit ADC, total bits for 20,000 sample points is 320,000 bits (~40KB).

##### Non-Volatile Memory

* Space to store firmware
* Configuration files
* Number of data points to be stored for offline computation
  + Cases when network is down
  + Requirement: 48 hours of data to be stored
    - 40KB/33.3ms, 1200KB/Sec, 4218MB/hour
    - 202GB/48hrs – To be discussed!
* NVM Read/Write/Erase timings
  + Erase is expensive in terms of time
    - Erase/Write strategy to be decided
    - Background task to manage - TBD

##### RAM

* Run time memory for code/data execution
* How many of such data points to be stored on RAM?
  + May need to hold a few previous data points for filtering/analysis/ etc.

#### ADCs/Multiplexers

* To optimize the hardware, we are using multiple ADCs and multiplexers.
* Data will be sampled in time multiplexed format using Mux to select subsequent data points
* Number of multiplexes (16:1) and ADCs: Need to be updated

#### CPU

* AT91SAM9G45C-CU 400 MHz ARM926EJ-S ARM® Thumb® Processor
  1. Since data sampling is being taken care by FPGA, MIPS requirement for CPU is not very high as of now
     + Unless, we decide to run some algorithms for data processing – This needs to be worked out
  2. This CPU supports all the peripherals that we need
     + 10/100 Mbps Ethernet MAC Controller (EMAC)
     + Two Master/Slave Serial Peripheral Interfaces (SPI)
     + Dual High-Speed USB Host and a High-Speed USB Device with On-Chip Transceivers
     + Two High Speed Memory Card Hosts (SDIO, SD Card, e.MMC and CE ATA)

#### Serial Debugger

* UART interface for debug

#### Wireless Module

* Wi-Fi support
* Should support 40KB/30ms, which is about 1.3MBps

#### Ethernet Port

* Ethernet support – Yes

#### USB Port

* Per current understanding, USB to be used for charging only.
* No data transfer requirements.

#### LEDs

* LEDs to indicate the status/health of device
  1. Power LED
  2. Network status LED
  3. Heartbeat LED
     + Periodic indicator to know the FW is alive

#### Network

* Detecting network outage
  1. What is the logic?
* Actions to be taken when the network is down?
  1. Store packets in RAM till certain threshold or start storing to Flash
  2. Flash
* Order of packets to be sent when the network is back?
  1. Stored ones to be sent first?

## Packet Format

Packets are designed to support bit error detection, loss of packet.

|  |  |
| --- | --- |
| Field | Size |
| Pkt Start | 0xA5 (Device Originated)  0x05 (Server Originated) |
| Pkt Type | 1bit | 15 bits (2 bytes)  MSB bit to indicate if its new start or continuation of previous packet |
| Pkt Len (in bytes) | 4 bytes (Data size) |
| Pkt Seq Number | 2 bytes |
| Check Sum | 2 bytes (CRC16, All fields, except for the checksum itself) |
| Data | Max of 40KB (20,000 samples) |

MSB bit in Pkt type:

Data packet is 40KB, but over Wi-Fi we need to transmit in smaller chunks. Hence, we need to split the packet.

If this bit is 0, indicates it’s the first packet, subsequent packets will have this bit set to 1.

This will help the remote parser to assemble the packet.

### Device Config Packet

|  |  |
| --- | --- |
| Field | Size |
| Pkt Start | 0x05 |
| Pkt Type | 0x0001 |
| Pkt Len | TBD |
| Pkt Seq Number | Generate a sequence number |
| Check Sum | 2 bytes |
| Data | Ex:  Data acquisition frequency  Data transmit frequency  Enable/disable filtering  ADC averaging. etc. |

### Acknowledgment Packet

|  |  |
| --- | --- |
| Field | Size |
| Pkt Start | 0xA5/0x05 |
| Pkt Type | 0x0002 |
| Pkt Len | 1 |
| Pkt Seq Number | sequence number of the pkt received |
| Check Sum | 2 bytes |
| Data | Success – 0, Failure – 1  More error codes to be define. |

### Raw Data Packet

|  |  |
| --- | --- |
| Field | Size |
| Pkt Start | 0xA5 |
| Pkt Type | 0x0003 |
| Pkt Len | 0x9C40 |
| Pkt Seq Number | Generate a sequence number |
| Check Sum | 2 bytes |
| Data | 0xAA  0xBB  .  .  .  Max of 40KB (20,000 samples) |

### Status Request

|  |  |
| --- | --- |
| Field | Size |
| Pkt Start | 0x05 |
| Pkt Type | 0x0004 |
| Pkt Len | 0 |
| Pkt Seq Number | Generate a sequence number |
| Check Sum | 2 bytes |
| Data | None |

### Status Response

|  |  |
| --- | --- |
| Field | Size |
| Pkt Start | 0xA5 |
| Pkt Type | 0x0005 |
| Pkt Len | TBD |
| Pkt Seq Number | Generate a sequence number |
| Check Sum | 2 bytes |
| Data | Return device config  More params - TBD |

### Log Packet

|  |  |
| --- | --- |
| Field | Size |
| Pkt Start | 0xA5 |
| Pkt Type | 0x00010 |
| Pkt Len | TBD |
| Pkt Seq Number | Generate a sequence number |
| Check Sum | 2 bytes |
| Data | Log for debug  Content: TBD |

## Firmware Upgrade

To be stored in flash. Loaded in RAM during boot up.

Store 2 copies of firmware to support fail proof upgrade

### Upgrade

* To be upgradable over
  1. UART
  2. Ethernet
  3. Wi-Fi
* Upgrade in background supported
  1. Always upgrade the inactive copy
  2. Switch to new firmware only if upgrade is successful

## Operating System

Current decision to go ahead with Linux.

Factors to consider

* OS Response time: Linux is slow compared to RTOS, but we are ok with it as CPU MIPS requirement is not critical now, as FPGA is taking care of data sampling.
* Network feature support
* Driver support

## Debug

* At the minimum we need UART interface for debug using primitive print statements
* In circuit debugger / JTAG
  + To be decided

# Timing Requirements

Sampling each node involves below listed steps. Let’s collect data for all the above steps in details

30Hz sampling => 20,000 nodes in 33.3ms => 1 node = 1.6us

* **Target:** 
  + **1 node in 1.6us**
* **Achievable:** 
  + 46ns + 23ns + 285ns (2us/7) **= ~ 354ns = approximate to 500ns**
    - Assuming DAC can be parallelized (step 3.2 and 3.4)
    - Assumption: DAC needs to set only once. Hence, it’s not being counted for timing
    - Assumption: FPGA will have 7 SPI ports to allow reading 7 ADCs in parallel
    - FPGA execution overhead: TBD
  + Sampling rate: **60Hz** (30/500)

## Select top and bottom multiplexer

* Part: Analog mux ADG726/ADG732
* Mux transition time: 23ns
* 2 mux time: **46ns**

## Set the DAC with fence voltage

* Part: DAC AD5761/AD5721
* Settling time: **7.5us**
* SPI Interface
  + Max speed: 50MHz, 1 byte = 160ns
  + Num of bytes required to write digital value: ~3 bytes
    - Time to write: 3\*160 = **480ns**
* Total time from this step: 7.5us + 480ns **= 8us**

## Select row using row multiplexer

* Part: Analog mux ADG726/ADG732
* Mux transition time: **23ns**

## Set the DAC for excitation voltage

* Part: DAC AD5761/AD5721
* Settling time: **7.5us**
* SPI Interface
  + Max speed: 50MHz, 1 byte = 160ns
  + Num of bytes required to write digital value: ~3 bytes
    - Time to write: 3\*160 = **480ns**
* Total time from this step: 7.5us + 480ns **= 8us**

## Setup ADC and read

* Part: AD7915
* Conversion time: 1us
* SPI Interface
  + Max clock speed: 16Mhz, 1 byte: 500ns
  + Num of bytes required to read digital value: ~2 bytes
    - Time to read: 2\*500 = **1us**
* Total time to read one ADC value**: 2ns**

## FPGA to read ADC value

* SPI Interface
  + Speed limited by ADC SPI speed, 16MHz

## FPGA to store the read values to memory

* FPGA latency to store ADC value in memory
  + Should be negligible

# FPGA Architecture

Node sampling is handled by FPAG. It must perform all the steps mentioned in section 3 above.

## Interfaces

### FPGA to ADC

Connected via 4 wire SPI bus. ADC supports max SPI speed of 16MHz (for AD5940, speed for AD7915 to be updated)

* New design consideration:
  + FPGA to support multiple SPIs (7) to read all ADCs simultaneously, this parallelization will boost the data sampling rate
  + As per current design we have 7 ADCs, so 7 data points can be read simultaneously.

### FPGA to DAC

Connected via 4 wire SPI bus. ADC supports max SPI speed of 50MHz

### FPGA to CPU

* To be connected via external bus interface – a 16-bit data interface.
* PIOs connected to select the control logic. There can be 16 different operations possible
  + Depending on operation it could be a data/control command
* An interrupt from FPGA to CPU to inform the data availability. CPU will start reading the data on receiving the interrupt
* Data buffer size in FPGA – TBD
  + Depends on CPU data read speed. Consumer must be faster than producer.
  + Need to work out these details
* FPGA will pump out data on every read strobe.

Note: This work is in progress. More details to be added.

# CPU Architecture

CPU is the central control hub. It interacts with FPGA, server, other sensors and decides the flow of action

Its primary task is to periodically fetch data from FPGA and send it over to server for further processing. Data transmit will be primarily over the Wi-Fi interface.

## DDR2 SDRAM

CPU has a DDR controller which takes care of RAM access.

## NAND Flash

### Boot NAND

This holds the boot loader

### Data NAND

Connected over external bus interface. 8-bit IO interface. Used to store code and other data.

## Micro SD Card

Drivers are available for Linux. Application logic around it to be developed.

## Wi-Fi Module

TBD: Which module? Interface? Speed?

## LAN Module

LAN drivers are available for Linux. Application logic around it to be developed

Ethernet switch used - LAN9303I-ABZJ

## LEDs

Controlled using GPIOs

## FPGA Memory

See [FPGA data bus](#_Data_Bus) section above.

## Software Flow Diagram

### High level flow diagram

Linux boot from NAND Flash

Initialize all interfaces. Wi-fi, LAN, SPI, UART

Load FPGA code & start sampling

LED Indicator/ Buzzer?

Is server connected?

Yes

No

A

Note: In the coming versions we will clean up the flow charts.

A

FPGA to sample and load data in ping pong buffer

CPU to read data on interrupt from FPGA

Accumulate 1 set of data point (40KB)

Is server connected?

No

Store data to Flash

Yes

Send data buffer to server

Note: In the coming versions we will clean up the flow charts.

### SW Tasks

* Any unsolicited message either from wi-fi or any other interface to be notified to CPU via interrupts.
* CPU will run either a periodic thread or a timer of 30ms periodicity to read the data from flash.
* Read data to be sent over to wi-fi module for transmitting to server

Note: More details to be added here

# References

## Datasheets

1. <https://www.analog.com/media/en/technical-documentation/data-sheets/AD8615_8616_8618.pdf>
2. <https://www.analog.com/media/en/technical-documentation/data-sheets/ADG726_732.pdf>
3. <https://www.analog.com/media/en/technical-documentation/data-sheets/ad5761_5721.pdf>
4. <https://www.analog.com/media/en/technical-documentation/data-sheets/AD5940.pdf>
5. <https://www.xilinx.com/support/documentation/data_sheets/ds312.pdf>
6. <http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-6438-32-bit-ARM926-Embedded-Microprocessor-SAM9G45_Datasheet.pdf>