

Android Bluetooth Digital Power Meter

By

Anushka Madhavi Samraweera

A project submitted in partial fulfillment of the requirements for the

Degree of Bachelor of Area Name in

Electronics Engineering

Examination Committee: Dr. Mongkol Ekpanyapong (Chairperson)

Dr. Attaphongse Taparugssanagorn, Dr. Apinun Tunpan

Nationality: Sri Lankan

Asian Institute of Technology

School of Engineering & Technology

Thailand

May 2013

Table of Contents		
Chapter	Title	Page
	Title Page	i
	Table of Contents	ii
	List of Figures	iii
	List of Tables	iv
1	Introduction	1
	1.1 Background	1
	1.2 Problem Statement	2
	1.3 Objectives	2
	1.4 Contributions	2
	1.5 Limitations and Scope	2
	1.6 Outline	3
2	Literature Review	4
	2.1 Information	4
	2.2 Bluetooth Technology	4
	2.3 What is Android?	5
	2.4 Arduino	7
	2.5 AC Power	8
3	Methodology	10
4	Resources	21
	4.1 Hardware	21
	4.2 Software	22
	4.3 Cost Analysis	23
5	Work Plan	24
	5.1 Time Line	24

List of Figures

Figure	Title
1	Power Meter
2	Devices which uses Bluetooth technology
3	Android system architecture
4	Android home screen
5	Arduino UNO development board
6	Instantaneous Power
7	Work plan
8	Block diagram
9	Front View
10	Back View
11	Bluetooth Application
12	AC Current sensing unit
13	Hall Effect
14	Pin out Diagram
15	AC Voltage sensing unit
16	MCP3221
17	Microcontroller unit
18	Pin Mapping
19	Timer unit
20	Power Supply Unit
21	Bluetooth module
22	Top layer
23	Bottom layer

List of Tables

Table	Title	Page
1	Estimated cost	23
2	Time Plan	24

Chapter 1 Introduction

1.1 Background

As world goes on due to globalization energy usage has been a basic need for every human being. 'Energy' the word also acquired a significant attention in this globalization era. But most people never realize how much energy they waste to accomplish their basic needs in their day to day life. So if people don't try to use energy in a sustainable manner it will be a real crisis for future righteous existence of every human being. So it is better if we try to save energy now. We can start this from our homes. If we can check how much power our home appliances use we can get a rough idea about, how we are going to reduce our daily or monthly power consumption. What about a power meter which can measure power consumption of home appliances?

In this project I developed a low cost wireless digital auto range power meter which uses Bluetooth technology to display outputs through a Bluetooth enabled Android mobile phone. Not only you can read data by phone but also there is a small display which shows the details. Bellow shows you the front face of this power meter.



Figure 1- power meter

Let's say if we want to measure a power consumption of an electric kettle. As shown in the figure 1, first the three pins should be plugged to a power socket. And then the electrical kettle should be plugged in to power meter socket. Then we can on the power socket switch and the power meter switch. When we boil the water we can see how much current it acquires and according to that how much power it consumes. Just like this we can plugged in any home appliance to this power meter and check how much power it consumes. The most important thing is when you are engage in another work enabling Bluetooth connection of our smart phone we can check all the above mentioned details very accurately an reliably.

Problem Statement

Most mainstream power meters suffer a number of distinct design shortcomings. They are hard to carry. Portable power meters exist commercially. They are portable when compared against other power meters, but still larger than one would carry around every day. Another thing is that, there is no such an option to save measured data in the existing power meters. It is too difficult task and actually a waste of time to take the measurements and write them down again. Having a feature of storing measured data, will be very useful and a time saver for users who work in busy environments. The Bluetooth enabled smartphones, are quickly becoming commonplace, being both convenient and affordable. They can be used to make, man's day to day activities much easier.

1.2 Objectives

The target is to design a digital power meter that connects via Bluetooth to a mobile phone device that the engineer would likely already own. This power meter will be designed to be inexpensive and small enough to fit in the user's pocket.

Objectives

- To build a portable device that fits to the user's pocket.
- To measure voltage, current and directly displays the power. (we don't want to calculate the power manually)
- To display measured values and save them in the mobile phone.

1.3 Contributions

This project will be beneficial to,

- Electronic Engineers
- Technicians
- Electronic Engineering Students

1.4 Limitations and Scope

Scope of my project is to develop a low cost wireless digital auto range power meter which uses Bluetooth technology to display outputs through a Bluetooth enabled Android mobile phone. So I uses ATmega328P microcontroller, LMX9838 Bluetooth module and Android SDK .This microcontroller is used for the fast Analogue to Digital data conversion and I used Arduino software to programme this microcontroller.

1.6 Outline

Chapter 1 explains the background of Android Bluetooth digital power meter, the project objective, the project scope.

Chapter 2 describes the literature review of the project.

Chapter 3 focuses on the methodologies for the development of the electrical structure and the implementations of microcontroller programming.

Chapter 4 describes resources that going to use this project and cost analysis.

Chapter 5 describes proposed planning and time line for the project.

Chapter 2

Literature Review

2.1 Information

Yus 2010, an engineering student in Philippines designed a portable wireless oscilloscope for android phones. This device can display visual images of varying electrical voltages over time. This project is a device that can be hooked up to an Android device via Bluetooth. It captures two channels of data at 2MHz/sec using a PIC microcontroller that converts the analog signals to digital signals and sends signals to an android mobile phone via Bluetooth for display them.

2.2 Bluetooth Technology

Bluetooth technology is a short-range communications technology that is simple, secure, and everywhere. You can find it in billions of devices ranging from mobile phones and computers to medical devices and home entertainment products. It is intended to replace the cables connecting devices, while maintaining high levels of security. The key features of Bluetooth technology are robustness, low power, and low cost. The Bluetooth Specification defines a uniform structure for a wide range of devices to connect and communicate with each other. When two Bluetooth enabled devices connect to each other, this is called pairing. The structure and the global acceptance of Bluetooth technology means any Bluetooth enabled device, almost everywhere in the world, can connect to other Bluetooth enabled devices located in proximity to one another. Bluetooth technology operates in the unlicensed industrial, scientific and medical (ISM) band at 2.4 to 2.485 GHz, using a spread spectrum, frequency hopping, full-duplex signal at a nominal rate of 1600 hops/sec. The 2.4 GHz ISM band is available and unlicensed in most countries. Range varies from class to class.

Class 3 radios – have a range of up to 1 meter or 3 feet

Class 2 radios – most commonly found in mobile devices – have a range of 10 meters or 33 feet

Class 1 radios – used primarily in industrial use cases – have a range of 100 meters or 300 feet



Figure 2- Devices which uses Bluetooth technology

Bluetooth technology's adaptive frequency hopping (AFH) capability was designed to reduce interference between wireless technologies sharing the 2.4 GHz spectrum. AFH works within the spectrum to take advantage of the available frequency. This is done by the technology detecting other devices in the spectrum and avoiding the frequencies they are using. This adaptive hopping among 79 frequencies at 1 MHz intervals gives a high degree of interference immunity and also allows for more efficient transmission within the spectrum. For users of Bluetooth technology this hopping provides greater performance even when other technologies are being used along with Bluetooth technology. The most commonly used radio is Class 2 and uses 2.5 mW of power. Bluetooth technology is designed to have very low power consumption. This is reinforced in the specification by allowing radios to be powered down when inactive.

The Generic Alternate MAC/PHY in Version 3.0 HS enables the discovery of remote AMPs for high speed devices and turns on the radio only when needed for data transfer giving a power optimization benefit as well as aiding in the security of the radios. Bluetooth low energy technology, optimized for devices requiring maximum battery life instead of a high data transfer rate, consumes between 1/2 and 1/100 the power of classic Bluetooth technology.

2.3 What is Android?

Android is a Linux-based operating system for mobile devices such as smartphones and tablet computers. It is developed by the Open Handset Alliance led by Google. Android was built from the ground-up to enable developers to create compelling mobile applications that take full advantage of all a handset has to offer. It was built to be truly open. For example, an application can call upon any of the phone's core

functionality such as making calls, sending text messages, or using the camera, allowing developers to create richer and more cohesive experiences for users. Android is built on the open Linux Kernel. Furthermore, it utilizes a custom virtual machine that was designed to optimize memory and hardware resources in a mobile environment. Android is open source; it can be liberally extended to incorporate new cutting edge technologies as they emerge. The platform will continue to evolve as the developer community works together to build innovative mobile applications.

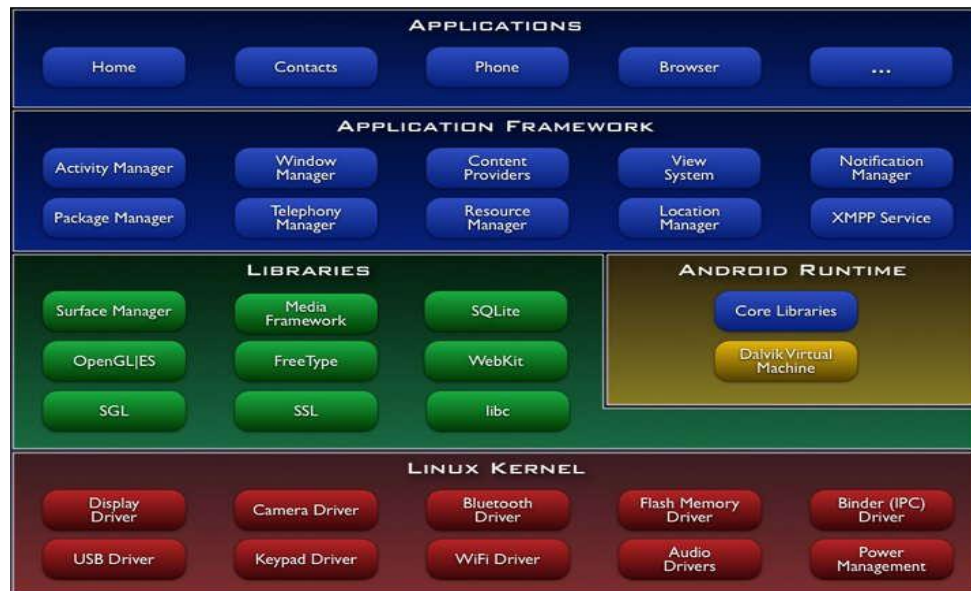


Figure 3– Android system architecture

Android does not differentiate between the phone's core applications and third-party applications. They can all be built to have equal access to a phone's capabilities providing users with a broad spectrum of applications and services. With devices built on the Android Platform, users are able to fully tailor the phone to their interests. They can swap out the phone's home screen, the style of the dialer, or any of the applications. They can even instruct their phones to use their favorite photo viewing application to handle the viewing of all photos.



Figure 4– Android home screen

Android breaks down the barriers to building new and innovative applications. For example, a developer can combine information from the web with data on an individual's mobile phone such as the user's contacts, calendar, or geographic location to provide a more relevant user experience. With Android, a developer can build an application that enables users to view the location of their friends and be alerted when they are in the vicinity giving them a chance to connect.

Android provides access to a wide range of useful libraries and tools that can be used to build rich applications. For example, Android enables developers to obtain the location of the device, and allows devices to communicate with one another enabling rich peer-to-peer social applications. In addition, Android includes a full set of tools that have been built from the ground up alongside the platform providing developers with high productivity and deep insight into their applications.

2.4 Arduino

Arduino is an open-source single-board microcontroller, descendant of the open-source Wiring platform, designed to make the process of using electronics in multidisciplinary projects more accessible. The hardware consists of a simple open hardware design for the Arduino board with an Atmel AVR processor and on-board input/output support. The software consists of a standard programming language compiler and the boot loader that runs on the board.

Arduino hardware is programmed using a Wiring-based language (syntax and libraries), similar to C++ with some slight simplifications and modifications, and a Processing-based integrated development environment.

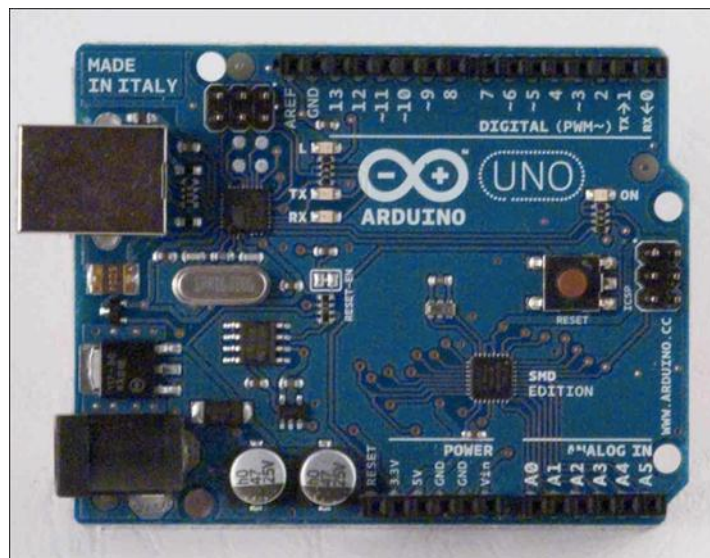


Figure 5– Arduino UNO development board

2.5 AC Power

In DC currents and voltages we calculate power with $P = VI$. But in AC currents and voltages $P = VI$ gives the instantaneous voltage, but these quantities are always varying with the time. So here we calculate the average power which is given by

$$P_{Avg} = VI \cos \phi$$

Where ϕ is the phase angle between the current and the voltage and where V and I are understood to be the effective or rms values of the voltage and current. The term $\cos \phi$ is called the "**power factor**" for the circuit.

2.6 Instantaneous Power

As in DC circuits, the instantaneous electric power in an AC circuit is given by $P=VI$ where V and I are the instantaneous voltage and current.

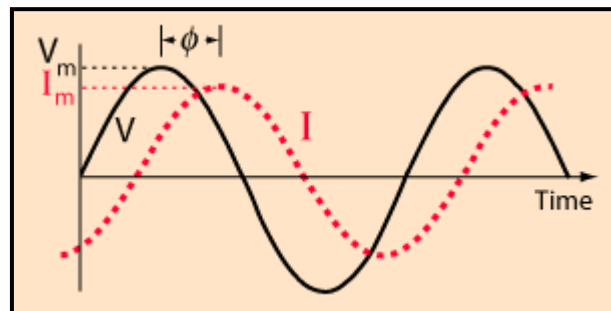


Figure 6

Since, $V = V_m \sin \omega t$, $I = I_m \sin (\omega t - \phi)$

Now the instantaneous power can be written as follows

$$P_{Instantaneous} = V_m I_m \sin \omega t \sin (\omega t - \phi)$$

$$P_{Instantaneous} = V_m I_m \cos \phi \sin^2 \omega t + V_m I_m \sin \phi \sin(\omega t) \cos(\omega t)$$

2.7 Average Power

Normally the average power is the power of interest in AC circuits. Since the expression for the instantaneous power is as follows

$$P_{\text{Instantaneous}} = V_m I_m \cos\phi \sin^2 \omega t + V_m I_m \sin\phi \sin(\omega t) \cos(\omega t)$$

Is a continuously varying one with time, the average must be obtained by integration. Averaging over one period T of the sinusoidal function will give the average power. The second term in the power expression above averages to zero since it is an odd function of t. The average of the first term is given by

$$\begin{aligned} P_{\text{Average}} &= V_m I_m \cos\phi \left\{ \int \sin^2 \omega t dt / T \right\} \\ &= (V_m I_m \cos\phi / 2) \end{aligned}$$

Since

$$V_{\text{rms}} = V_m / \sqrt{2}$$

$$I_{\text{rms}} = I_m / \sqrt{2}$$

$$P_{\text{Avg}} = VI \cos\phi$$

Chapter 3 Methodology

This chapter will explain the methodology for Android Bluetooth Digital Power Meter. Simple idea about the development of the project can be taken from, flowchart illustrated below.

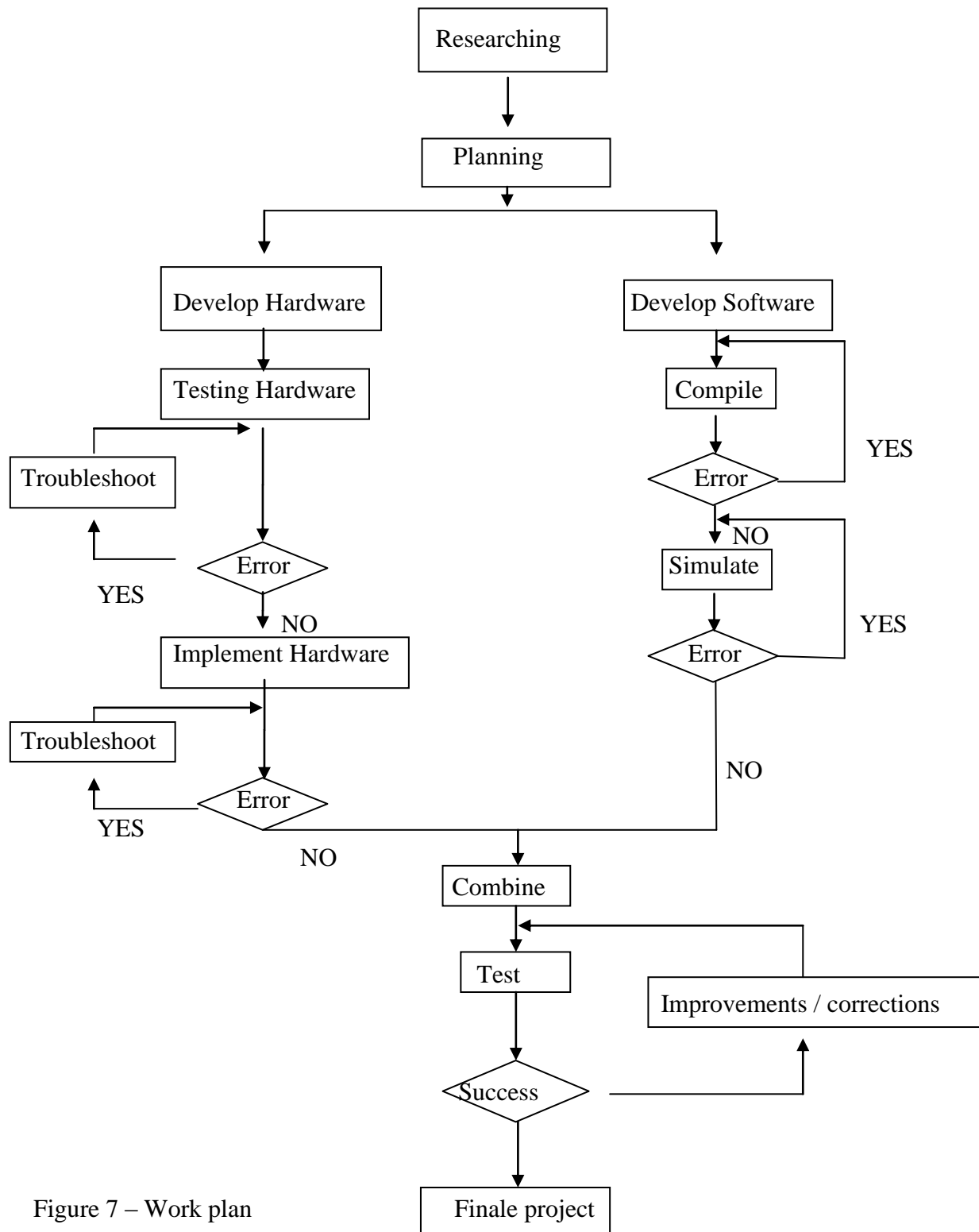


Figure 7 – Work plan

3.1 System Overview

The first stages in the work plan are to research about the background of the project and technologies which are used. It also includes literature review. After the researching is done, aims and objectives for the project are created. In the next step, planning of the project completion process is done. The way of doing the project in a successful manner is observed. Technical knowledge for the project is gathered. Then the project is divided into two parts called hardware part and the software part. Hardware development of the Bluetooth power meter device is done in the hardware part. The circuitry designing, simulation using breadboards, basic casings designing for the finale device are done in here. In the software part, Android application development and the program code for the microcontroller are done. Both hardware part and the software part have done simultaneously in order to achieve a better, successful outcome. After both parts have done correctly with no issues they were combined together then pcb and the casings for the finale device were designed.

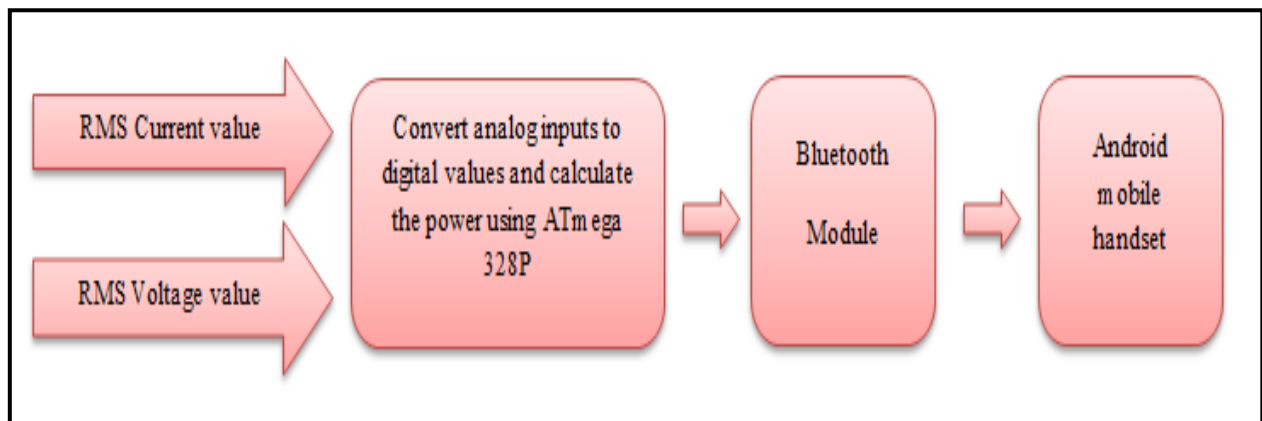


Figure 8 – The Blog diagram



Figure 9-Front View



Figure 10 Back View

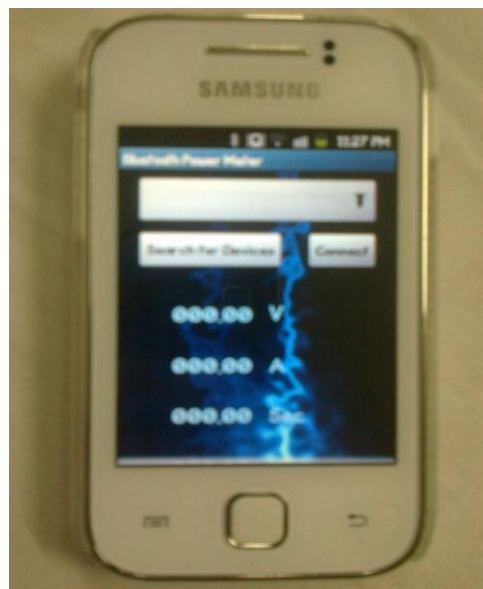


Figure 11-Bluetooth Application

There are six main systems in this power meter design.

- ❖ AC current sensing unit
- ❖ AC voltage sensing unit
- ❖ Microcontroller unit
- ❖ Timer unit
- ❖ Power supply unit
- ❖ Bluetooth Module

AC Current sensing unit

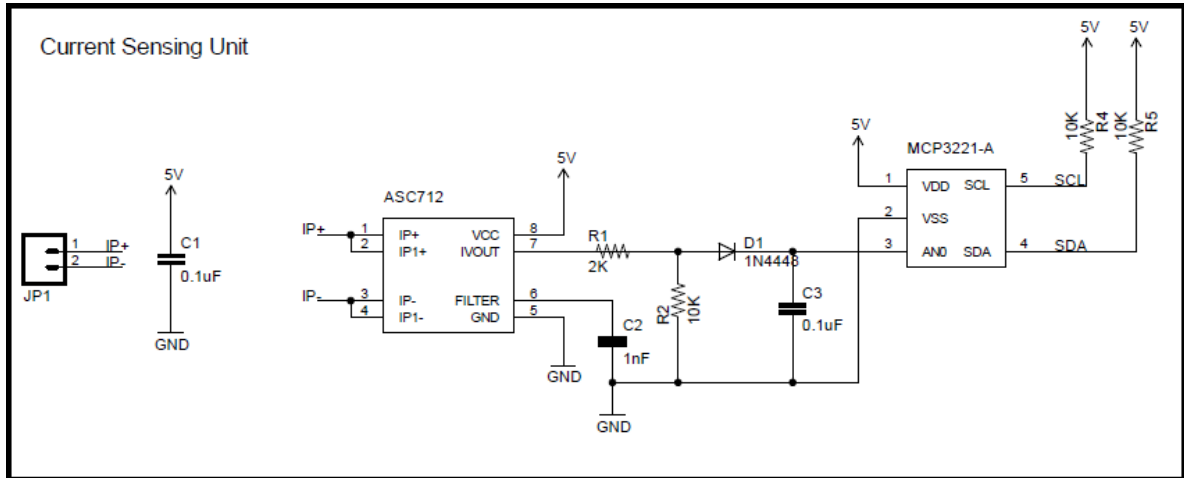


Figure 12

To measure power of an electrical element we need two readings, one is the current and the other one is the voltage. So to measure the current I used Allegro ACS712 current sensor. Let's discuss about this current sensor in more detail.

Allergo ACS712 current sensor is based on the principle of Hall-effect. According to this principle, when a current carrying conductor is placed into a magnetic field, a voltage is generated across its edges perpendicular to the directions of both the current and the magnetic field. The strength of the magnetic field is proportional to the magnitude of the current through the conduction path, providing a linear relationship between the output Hall voltage and input conduction current. The on-chip signal conditioner and filter circuit stabilizes and enhances the induced Hall voltage to an appropriate level so that it could be measured through an ADC channel of a microcontroller.

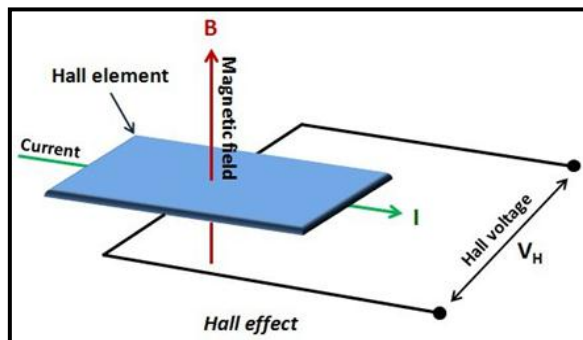


Figure 13

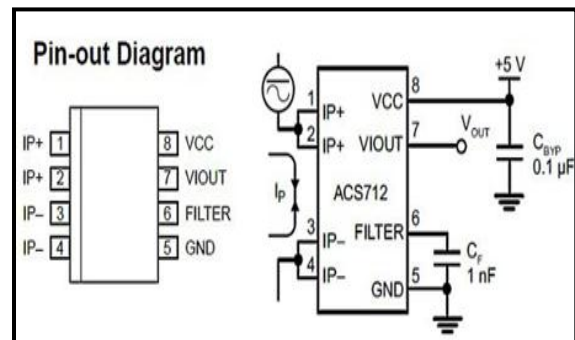


Figure 14

The pin diagram of ACS712 device and its typical application circuit is shown above. Pins 1, 2 and 3, 4 form the copper conduction path which is used for current sensing. The internal resistance of this path is around 1.2 mΩ, thus providing low power loss. As the terminals of this conduction path are electrically isolated from the sensor leads (pins 5 through 8), the ACS712 device eliminates the risk of damaging the current monitoring circuit due to the high voltage on the conduction side. The electrical isolation between the conduction current and the sensor circuit also minimizes the safety concerns while dealing with high voltage systems.

AC Voltage sensing unit

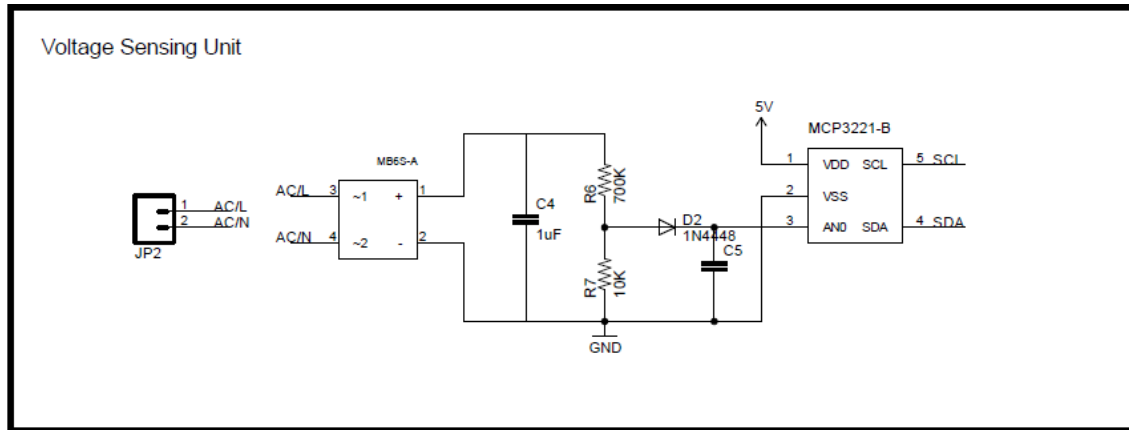


Figure 15

Second sub system is for the AC voltage measuring. First the input AC signal is rectified. Then the rectified signal will be reduced by a voltage divider circuit and then the reduced signal will be fed to the ADC of the microcontroller for sampling. Sampling is done inside the microcontroller for rms voltage calculation. Here I used MB6S bridge rectifier to rectify the AC voltage.

Root mean square (abbreviated RMS or rms), also known as the quadratic mean, is a statistical measure of the magnitude of a varying quantity. It is especially useful when varieties are positive and negative. Waveforms made by summing known simple waveforms have an RMS that is the root of the sum of squares of the component RMS values, if the component waveforms are orthogonal (that is, if the average of the product of one simple waveform with another is zero for all pairs other than a waveform times itself).

MCP3221

In both these units I used MCP3221 Low Power 12-Bit A/D Converter to increase the resolution. In ATmega328P there are only 10 bits. So resolution is low but when we use MCP3221 there are 12 bits so resolution is higher than the ATmega328P. So by increasing the resolution we can observe more accurate and correct reading from the power meter.

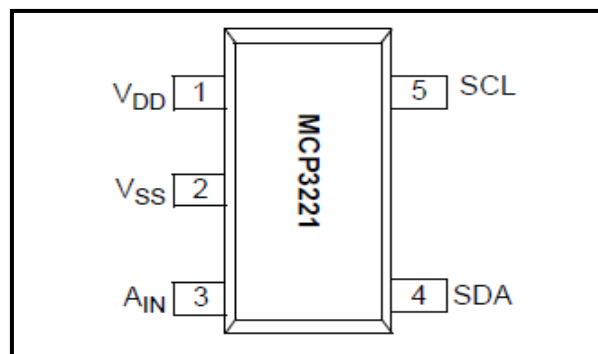


Figure 16

Serial Data (SDA)

SDA is a bidirectional pin used to transfer addresses and data into and out of the device. Since it is an open drain-terminal, the SDA bus requires a pull-up resistor to VDD (typically 10 k Ω for 100 kHz and 2 k Ω for 400 kHz SCL clock speeds). For normal data transfer, SDA is allowed to change only during SCL low. Changes during SCL high are reserved for indicating the START and STOP conditions.

Serial Clock (SCL)

SCL is an input pin used to synchronize the data transfer to and from the device on the SDA pin and is an open-drain terminal. Therefore, the SCL bus requires a pull-up resistor to VDD (typically 10 k Ω for 100 kHz and 2 k Ω for 400 kHz SCL clock speeds). For normal data transfer, SDA is allowed to change only during SCL low. Changes during SCL high are reserved for indicating the START and STOP conditions.

Microcontroller unit

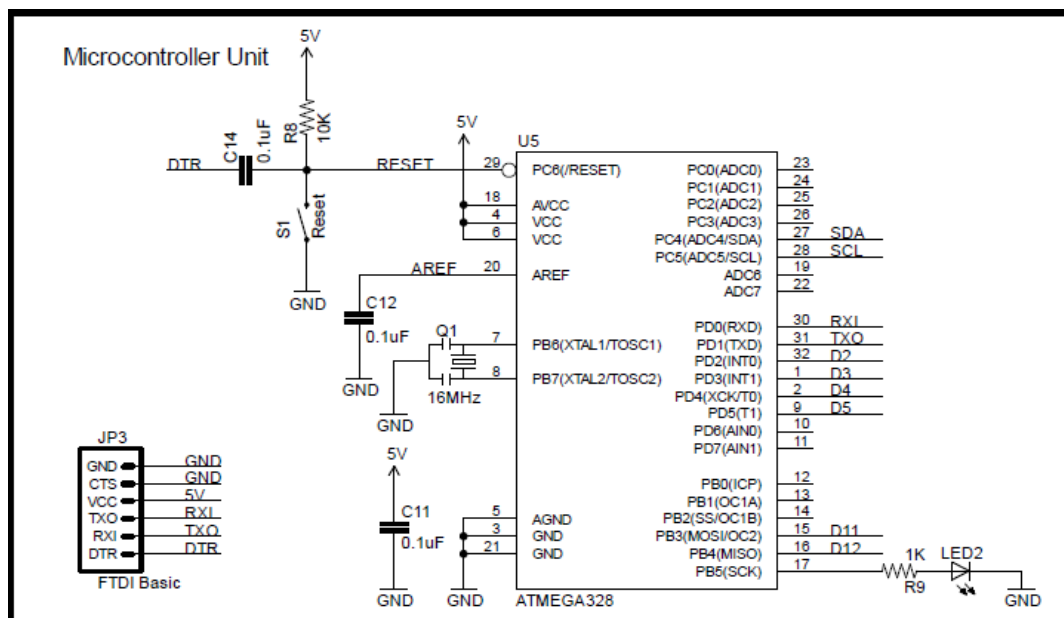


Figure 17

In this system I used ATMEGA328 microcontroller. The basic tasks of the microcontroller are, convert the analog data in to digital and send data to both LCD display and Bluetooth module. Pin mapping has been shown below.

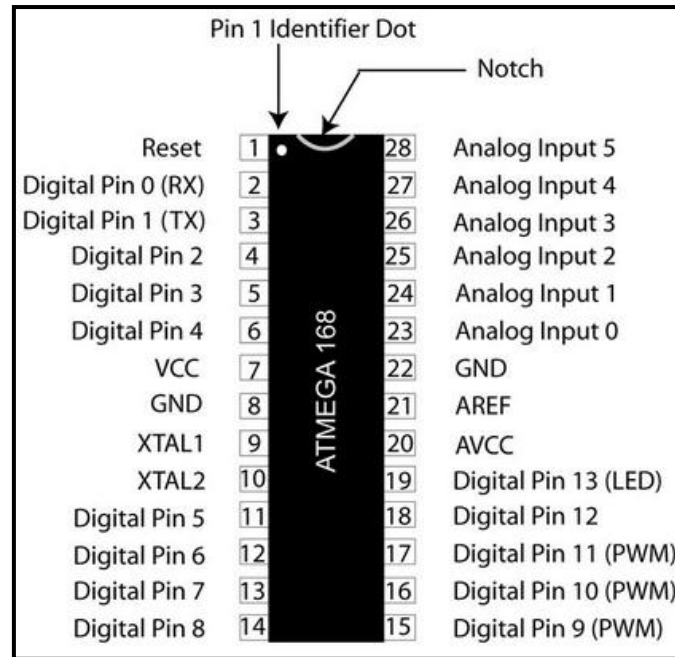


Figure 18

As shown in figure 13 I have to connect all the SDA outputs to the 27th pin and all the SCL outputs to the 28th pin. SDA and SCL outputs carry analog inputs to the Microcontroller. Two of these SDA and SCL inputs carries the signals corresponding to the voltage and current sensing units.

Timer unit

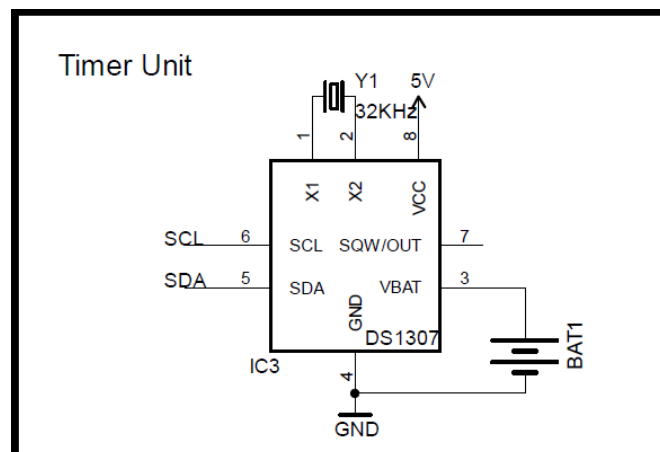


Figure 19

We need a stable clock to run the microcontroller. So I used a crystal oscillator here.

In some systems the Microcontroller and bus timing is a limit to the highest and sometimes the lowest Microcontroller clock frequency, so the clock needs to be stable enough to guarantee timing requirements are met according to the design and the specification.

In other systems the Microcontroller timing might be used for timing of application tasks. For example, a real time clock, or a counter timer used to measure periods or generate accurate clock ticks may be involved. These need a reference clock that is sufficiently accurate for the needs of that particular application.

Power supply unit

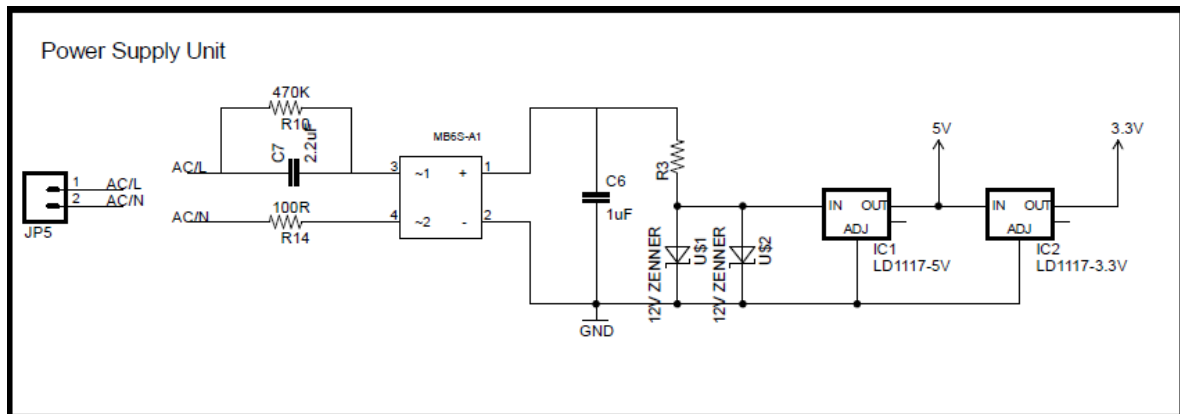


Figure 20

Bluetooth module

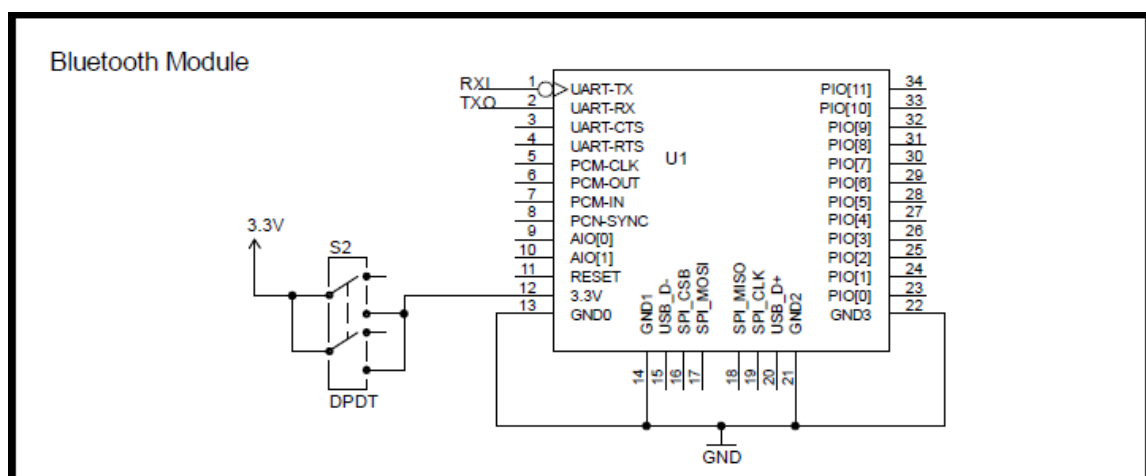


Figure 21

To transmit the data to the smart phone we need a Bluetooth module.

3.2 System Design

Before the design of the circuit I search what kinds of power meters are available in the market today. There were many products but there were no such a device that can remotely access the meter and obtain the data.

Then I search about power meter circuits and how do they work. There were various kinds of circuit diagrams but most of them were analog ones. As we manually calculate the power using the $P = VI$ equation, I thought that if I can obtain the RMS voltages and RMS currents, power can be easily calculated. Keeping these points in mind I search methods to obtain RMS voltage and RMS currents. Finally I found that using the **Allegro ACS712 Hall-Effect sensor** and the **MB6S bridge rectifier** with voltage divider circuit we can obtain the RMS currents and RMS voltages respectively. Using the data sheets of these components I designed the circuit.

In designing the circuit I used **EAGLE PCB design software**.

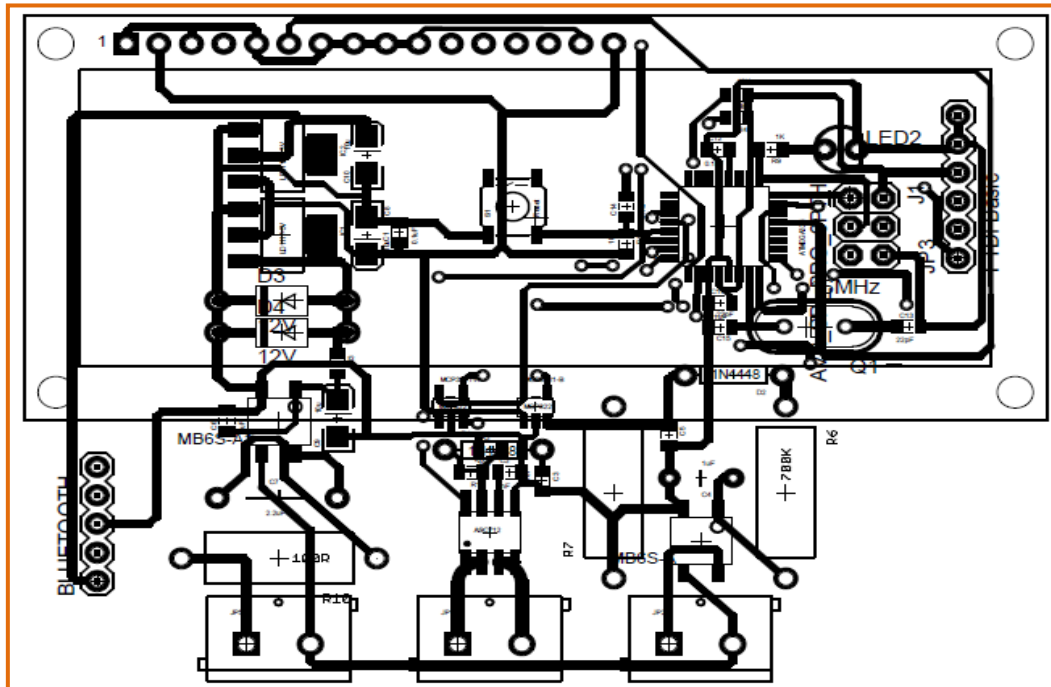


Figure 22-Top layer

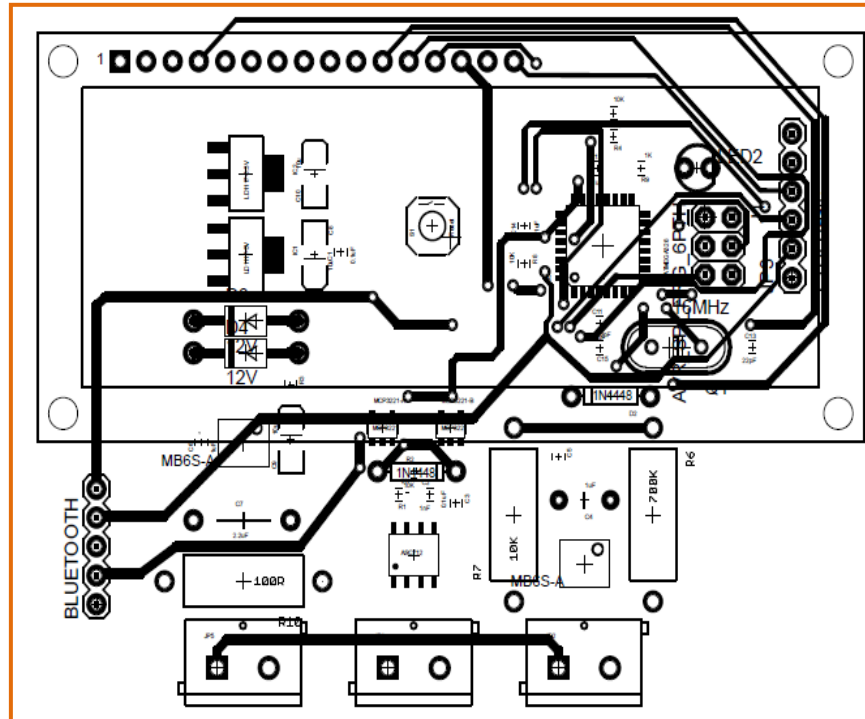


Figure 23-Bottom layer

3.3 Evaluation and Discussion

The Wireless Power Meter is a simplistic ATmega328P and Bluetooth enabled true V-I power meter. AC voltage measurement is obtained by a MB6S bridge rectifier with voltage divider circuit, and current measurement is made with the pass-through Allegro ACS712 Hall-Effect sensor. The microcontroller and wireless transceiver are magnetically and optically isolated from the high voltage power circuitry.

This project has some important issues and proposed solutions that are worth pointing out to anyone interested in building such a power meter:

- ❖ The rectification for the voltage measurement signal isn't perfect, as there is some clipping and loss in the diode full-wave bridge rectifier, so this clipping can be remedied by using an analog full-wave precision rectifier.
- ❖ Leaked magnetic flux from the power supply transformer influences the allegro current sensor measurement, imposing a waveform based on the measurement circuit's very own current load.

There were unexpected practical issues when implementing this project IC s and ATmega328P chips have fried several times during the testing. They were replaced and the process wasted some valuable time due to some of them are not available in the local market. Finding components in the local market was another challenge. Some components had to be imported. The broader accomplishment of

this project is the development of a very modular framework. Each element has been chosen with care and each interface has been designed such that a simple swap of any given component can be made without detriment to the whole of the design. Final effort is to build this project to meet with the industrial standard.

Chapter 4 Resources

4.1 Hardware

4.1.1 Arduino Uno board with ATmega328P

It has enough ADC pins and its low power consuming. After programming the ATmega328P is done, microcontroller will pull from the Arduino Uno board and placed in the PCB. An Arduino board consists of an 8-bit Atmel AVR microcontroller with complementary components to facilitate programming and incorporation into other circuits. An important aspect of the Arduino is the standard way that connectors are exposed, allowing the CPU board to be connected to a variety of interchangeable add-on modules known as shields. Some shields communicate with the Arduino board directly over various pins, but many shields are individually addressable via an I²C serial bus, allowing many shields to be stacked and used in parallel. Official Arduinos have used the megaAVR series of chips, specifically the ATmega8, ATmega168, ATmega328, ATmega1280, and ATmega2560. A handful of other processors have been used by Arduino compatibles. Most boards include a 5 volt linear regulator and a 16 MHz crystal oscillator (or ceramic resonator in some variants), although some designs such as the LilyPad run at 8 MHz and dispense with the onboard voltage regulator due to specific form-factor restrictions. An Arduino's microcontroller is also pre-programmed with a boot loader that simplifies uploading of programs to the on-chip flash memory, compared with other devices that typically need an external programmer.

4.1.2 Bluetooth module

- Chipset CSR BC417143 (BlueCore4External)
- Bluetooth version V2.0+EDR
- Output power Class II
- Flash 8Mbit
- Power Supply 3.3V
- Interface I²C UART PCM USB1.2
- Size 26.9mm*13mm*2.2mm
- Rohs: Yes
- Range 10m

Bluetooth module uses CSR BlueCore4- External chipsets. It embeds 8Mbit Flash for software storage, and supports 3.3V power supply. BC04 is a multi-function module. It can be used in different products according to the embedded • firmware settings. It is especially targeted for data transfer. The second generation Bluetooth UART module has two working mode: AT command mode, and automatic binding transparent data mode. In automatic binding data transparent mode, it can be configured to Master, Slave or Loopback three different modes, and it will connect to or be connected by other devices that support SPP protocol per configuration. In AT command mode, user can configure the module and send control commands.

4.2 Software

4.2.1 Arduino IDE

The Arduino IDE is a cross-platform application written in Java, and is derived from the IDE for the Processing programming language and the Wiring project. It is designed to introduce

programming to artists and other newcomers unfamiliar with software development. It includes a code editor with features such as syntax highlighting, brace matching, and automatic indentation, and is also capable of compiling and uploading programs to the board with a single click. There is typically no need to edit make files or run programs on a command-line interface. Although building on command-line is possible if required with some third-party tools.

4.2.2 Android SDK

Android SDK is a software development kit that enables developers to create applications for the Android platform. The Android SDK includes sample projects with source code, development tools, an emulator, and required libraries to build Android applications. Applications are written using the Java programming language and run on Dalvik, a custom virtual machine designed for embedded use which runs on top of a Linux kernel.

4.2.3 Eagle PCB design software

EAGLE stands for Easily Applicable Graphical Layout Editor, and is a program aimed at letting you design clean, precise, and complex circuit board layouts.

EAGLE actually contains two distinct programs that function and communicate with each other, in order to make the layout process as readable as possible. These two programs are the Eagle Schematic Editor, which allows the user to either author custom designs, or redraw schematics found from the internet, magazines, and other sources. These schematics can then be brought into the other half of Eagle, the Board Layout Editor. The Layout Editor reads files created in the schematic editor, and translates them into web components that can be rearranged much like a jig-saw puzzle. Once these parts have been arranged in a coherent manner, board traces can be drawn and edited, creating a complete PCB layout.

4.3 Cost Analysis

I purchased all the components from SriLanka due to some practical problems in Thailand

Estimated cost for the project (1Thai Baht=4.3 LKR)

Component	Quantity	Cost(LKR)
ACS712 current sensor	1	1200
MCCP3221 A2D convertor	2	910
MB6S bridge rectifier	2	70
LD1117-5V	1	60
LD1117-3.3V	1	60
Bluetooth module	1	2500
LCD 2*16	1	400
DS1307	1	100
32KHz Crystal	1	20
16MHz Crystal	1	20
Coin Cell Battery	1	100
ATMega328P	1	500
1N4448	2	20
12v Zenner	2	20
Resistor	-	100
Capasitor	-	200
Male Header 0.1"	1	40
Female Header 0.1"	1	40
Power Terminal	3	60
PCB	-	500
Enclosure Box	1	1000
Total		7920 (approximately 2000BHT)

Table 1 – Estimated cost

Chapter 5

Work Plan

This chapter describes my proposed planning in order to implement and improve my methodology for Android Bluetooth digital power meter

5.1 Time Line

Work	Start	Finish
Finalize project selection.	-	03/12/2012
Explore and understanding about the project	03/12/2012	07/12/2012
Consult in order to understand and investigate about project problem.	07/12/2012	05/11/2012
Present the proposal	-	24/12/2012
Purchasing Components	28/12/2012	02/01/2013
Simulate Using Softwares	02/01/2013	15/01/2013
Hardware designing and testing	15/01/2013	01/03/2013
Create Android Software	01/03/2013	15/04/2013
Troubleshooting and Improvements	15/04/2013	01/05/2013
Finalize the all project and present Final report and presentation	01/05/2013	09/05/2013

Table 2-Time Plan

References

What is android?

http://www.openhandsetalliance.com/android_overview.html

Android developers

<http://www.android.com/developers/>.

What is Bluetooth technology?

www.bluetooth.com/

LN7805 datasheet

<http://www.datasheetarchive.com/ln%207805-datasheet.html>

Arduino reference

<http://arduino.cc/en/Reference/HomePage>

Yus (2010) Bluetooth oscilloscope project

<http://projectproto.blogspot.com/2010/09/android-bluetooth-oscilloscope.html>

Appendices

Appendix A – Atmega 328P datasheet

Features

- High Performance, Low Power AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
 - 131 Powerful Instructions – Most Single Clock Cycle Execution
 - 32 x 8 General Purpose Working Registers
 - Fully Static Operation
 - Up to 20 MIPS Throughput at 20 MHz
 - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
 - 4/8/16/32K Bytes of In-System Self-Programmable Flash program memory (ATmega48PA/88PA/168PA/328P)
 - 256/512/512/1K Bytes EEPROM (ATmega48PA/88PA/168PA/328P)
 - 512/1K/1K/2K Bytes Internal SRAM (ATmega48PA/88PA/168PA/328P)
 - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
 - Data retention: 20 years at 85°C/100 years at 25°C⁽¹⁾
 - Optional Boot Code Section with Independent Lock Bits
In-System Programming by On-chip Boot Program
True Read-While-Write Operation
 - Programming Lock for Software Security
- Peripheral Features
 - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
 - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
 - Real Time Counter with Separate Oscillator
 - Six PWM Channels
 - 8-channel 10-bit ADC in TQFP and QFN/MLF package
Temperature Measurement
 - 6-channel 10-bit ADC in PDIP Package
Temperature Measurement
 - Programmable Serial USART
 - Master/Slave SPI Serial Interface
 - Byte-oriented 2-wire Serial Interface (Philips PC compatible)
 - Programmable Watchdog Timer with Separate On-chip Oscillator
 - On-chip Analog Comparator
 - Interrupt and Wake-up on Pin Change
- Special Microcontroller Features
 - Power-on Reset and Programmable Brown-out Detection
 - Internal Calibrated Oscillator
 - External and Internal Interrupt Sources
 - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby
- I/O and Packages
 - 23 Programmable I/O Lines
 - 28-pin PDIP, 32-lead TQFP, 28-pad QFN/MLF and 32-pad QFN/MLF
- Operating Voltage:
 - 1.8 - 5.5V for ATmega48PA/88PA/168PA/328P
- Temperature Range:
 - -40°C to 85°C
- Speed Grade:
 - 0 - 20 MHz @ 1.8 - 5.5V
- Low Power Consumption at 1 MHz, 1.8V, 25°C for ATmega48PA/88PA/168PA/328P:
 - Active Mode: 0.2 mA
 - Power-down Mode: 0.1 µA
 - Power-save Mode: 0.75 µA (Including 32 kHz RTC)



**8-bit AVR®
Microcontroller
with 4/8/16/32K
Bytes In-System
Programmable
Flash**

**ATmega48PA
ATmega88PA
ATmega168PA
ATmega328P**

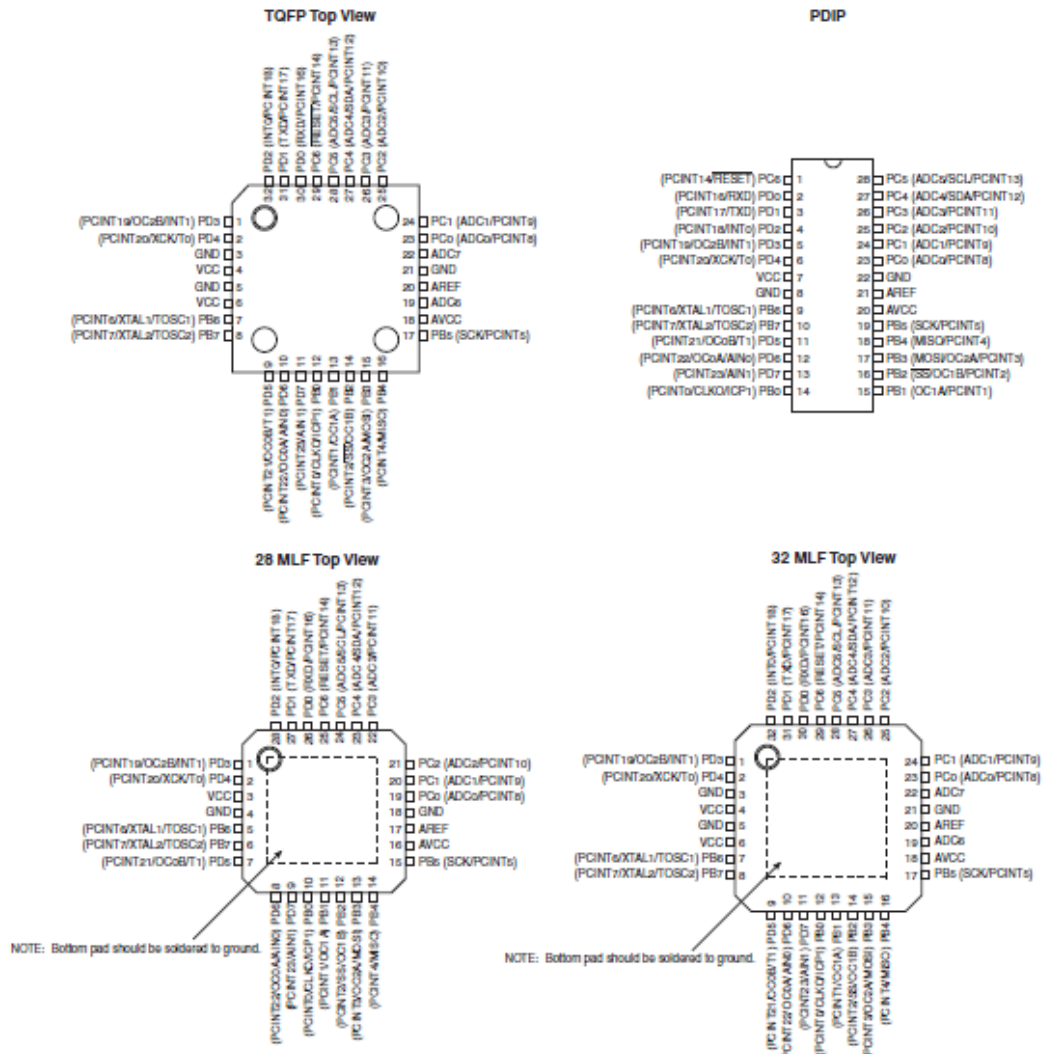
Summary

Rev. 8161DS-AVR-10/00



1. Pin Configurations

Figure 1-1. Pinout ATmega48PA/88PA/168PA/328P



ATmega48PA/88PA/168PA/328P

1.1 Pin Descriptions

1.1.1 VCC

Digital supply voltage.

1.1.2 GND

Ground.

1.1.3 Port B (PB7:0) XTAL1/XTAL2/TOSC1/TOSC2

Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Depending on the clock selection fuse settings, PB6 can be used as input to the Inverting Oscillator amplifier and input to the internal clock operating circuit.

Depending on the clock selection fuse settings, PB7 can be used as output from the Inverting Oscillator amplifier.

If the internal Calibrated RC Oscillator is used as chip clock source, PB7..6 is used as TOSC2..1 input for the Asynchronous Timer/Counter2 if the AS2 bit in ASSR is set.

The various special features of Port B are elaborated in "Alternate Functions of Port B" on page 76 and "System Clock and Clock Options" on page 26.

1.1.4 Port C (PC5:0)

Port C is a 7-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The PC5..0 output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated. The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running.

1.1.5 PC6/RESET

If the RSTDISBL Fuse is programmed, PC6 is used as an I/O pin. Note that the electrical characteristics of PC6 differ from those of the other pins of Port C.

If the RSTDISBL Fuse is unprogrammed, PC6 is used as a Reset input. A low level on this pin for longer than the minimum pulse length will generate a Reset, even if the clock is not running. The minimum pulse length is given in Table 28-3 on page 308. Shorter pulses are not guaranteed to generate a Reset.

The various special features of Port C are elaborated in "Alternate Functions of Port C" on page 79.

1.1.6 Port D (PD7:0)

Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running.

8121DS-AVR-10/09



3

ATmega48PA/88PA/168PA/328P

The various special features of Port D are elaborated in "Alternate Functions of Port D" on page 82.

1.1.7 AV_{CC}

AV_{CC} is the supply voltage pin for the A/D Converter, PC3:0, and ADC7:6. It should be externally connected to V_{CC}, even if the ADC is not used. If the ADC is used, it should be connected to V_{CC} through a low-pass filter. Note that PC6..4 use digital supply voltage, V_{CC}.

1.1.8 AREF

AREF is the analog reference pin for the A/D Converter.

1.1.9 ADC7:6 (TQFP and QFN/MLF Package Only)

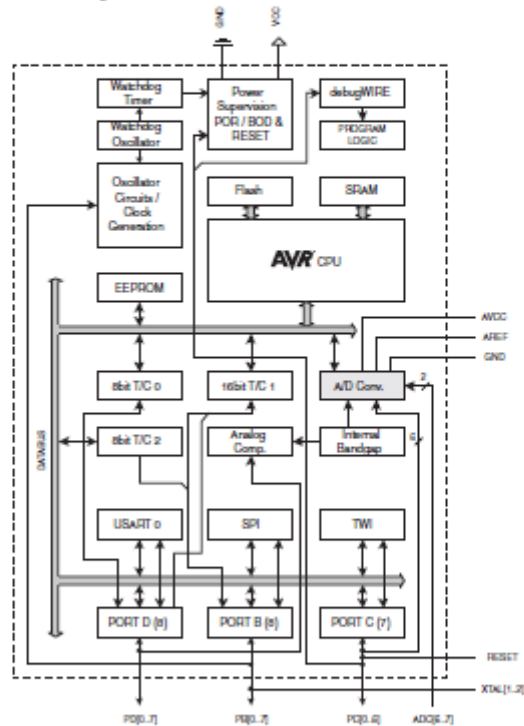
In the TQFP and QFN/MLF package, ADC7:6 serve as analog inputs to the A/D converter. These pins are powered from the analog supply and serve as 10-bit ADC channels.

2. Overview

The ATmega48PA/88PA/168PA/328P is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega48PA/88PA/168PA/328P achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

2.1 Block Diagram

Figure 2-1. Block Diagram



The AVR core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting

ATmega48PA/88PA/168PA/328P

3. Resources

A comprehensive set of development tools, application notes and datasheets are available for download on <http://www.atmel.com/avr>.

4. Data Retention

Reliability Qualification results show that the projected data retention failure rate is much less than 1 PPM over 20 years at 85°C or 100 years at 25°C.

Appendix B – ACS712 datasheet

Features and Benefits

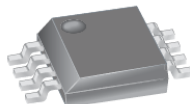
- Low-noise analog signal path
- Device bandwidth is set via the new FILTER pin
- 5 μ s output rise time in response to step input current
- 80 kHz bandwidth
- Total output error 1.5% at $T_A = 25^\circ\text{C}$
- Small footprint, low-profile SOIC8 package
- 1.2 m Ω internal conductor resistance
- 2.1 kVRMS minimum isolation voltage from pins 1-4 to pins 5-8
- 5.0 V, single supply operation
- 66 to 185 mV/A output sensitivity
- Output voltage proportional to AC or DC currents
- Factory-trimmed for accuracy
- Extremely stable output offset voltage
- Nearly zero magnetic hysteresis
- Ratiometric output from supply voltage



TUV America
Certificate Number:
U8V 06 05 54214 010



Package: 8 Lead SOIC (suffix LC)



Approximate Scale 1:1



Description

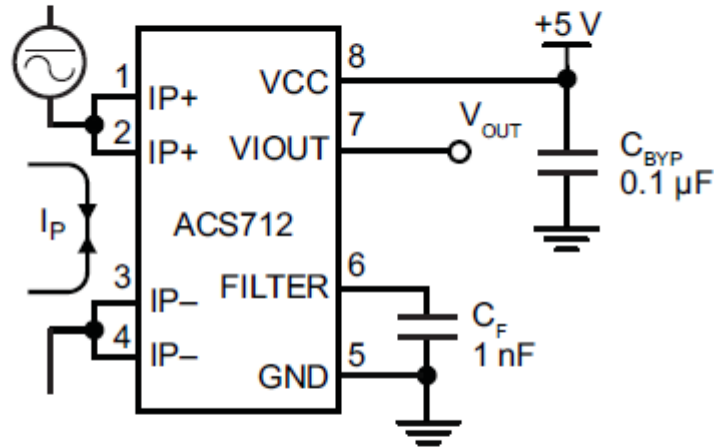
The Allegro® ACS712 provides economical and precise solutions for AC or DC current sensing in industrial, commercial, and communications systems. The device package allows for easy implementation by the customer. Typical applications include motor control, load detection and management, switch-mode power supplies, and overcurrent fault protection. The device is not intended for automotive applications.

The device consists of a precise, low-offset, linear Hall circuit with a copper conduction path located near the surface of the die. Applied current flowing through this copper conduction path generates a magnetic field which the Hall IC converts into a proportional voltage. Device accuracy is optimized through the close proximity of the magnetic signal to the Hall transducer. A precise, proportional voltage is provided by the low-offset, chopper-stabilized BiCMOS Hall IC, which is programmed for accuracy after packaging.

The output of the device has a positive slope ($>V_{IOUT(Q)}$) when an increasing current flows through the primary copper conduction path (from pins 1 and 2, to pins 3 and 4), which is the path used for current sampling. The internal resistance of this conductive path is 1.2 m Ω typical, providing low power loss. The thickness of the copper conductor allows survival of

Continued on the next page...

Typical Application



Application 1. The ACS712 outputs an analog signal, V_{OUT} , that varies linearly with the uni- or bi-directional AC or DC primary sampled current, I_P , within the range specified. C_F is recommended for noise management, with values that depend on the application.

Description (continued)

the device at up to $5\times$ overcurrent conditions. The terminals of the conductive path are electrically isolated from the signal leads (pins 5 through 8). This allows the ACS712 to be used in applications requiring electrical isolation without the use of opto-isolators or other costly isolation techniques.

The ACS712 is provided in a small, surface mount SOIC8 package. The leadframe is plated with 100% matte tin, which is compatible with standard lead (Pb) free printed circuit board assembly processes. Internally, the device is Pb-free, except for flip-chip high-temperature Pb-based solder balls, currently exempt from RoHS. The device is fully calibrated prior to shipment from the factory.

Selection Guide

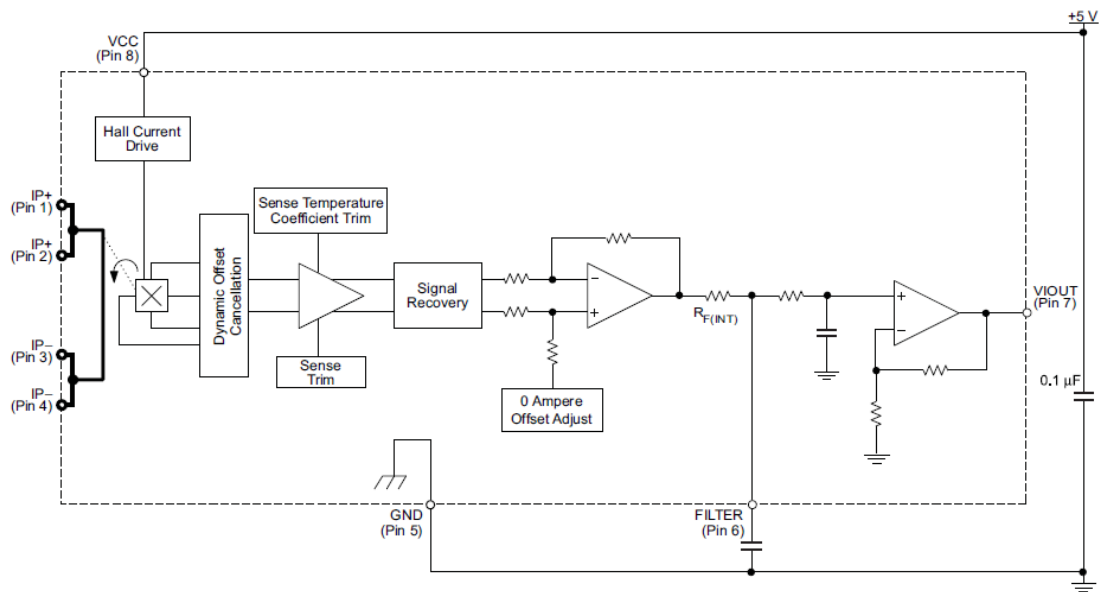
Part Number	Packing*	T_A (°C)	Optimized Range, I_P (A)	Sensitivity, Sens (Typ) (mV/A)
ACS712ELCTR-05B-T	Tape and reel, 3000 pieces/reel	-40 to 85	± 5	185
ACS712ELCTR-20A-T	Tape and reel, 3000 pieces/reel	-40 to 85	± 20	100
ACS712ELCTR-30A-T	Tape and reel, 3000 pieces/reel	-40 to 85	± 30	66

*Contact Allegro for additional packing options.

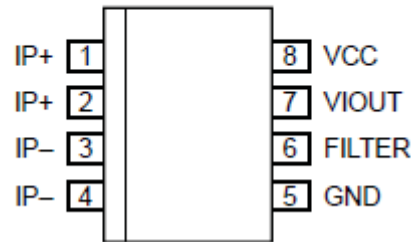
Absolute Maximum Ratings

Characteristic	Symbol	Notes	Rating	Units
Supply Voltage	V_{CC}		8	V
Reverse Supply Voltage	V_{RCC}		-0.1	V
Output Voltage	V_{IOUT}		8	V
Reverse Output Voltage	V_{RIOUT}		-0.1	V
Reinforced Isolation Voltage	V_{ISO}	Pins 1-4 and 5-8; 60 Hz, 1 minute, $T_A=25^{\circ}\text{C}$	2100	VAC
		Maximum working voltage according to UL60950-1	184	V_{peak}
Basic Isolation Voltage	$V_{ISO(bsc)}$	Pins 1-4 and 5-8; 60 Hz, 1 minute, $T_A=25^{\circ}\text{C}$	1500	VAC
		Maximum working voltage according to UL60950-1	354	V_{peak}
Output Current Source	$I_{IOUT(Source)}$		3	mA
Output Current Sink	$I_{IOUT(Sink)}$		10	mA
Overcurrent Transient Tolerance	I_P	1 pulse, 100 ms	100	A
Nominal Operating Ambient Temperature	T_A	Range E	-40 to 85	$^{\circ}\text{C}$
Maximum Junction Temperature	$T_J(max)$		165	$^{\circ}\text{C}$
Storage Temperature	T_{stg}		-65 to 170	$^{\circ}\text{C}$

Functional Block Diagram



Pin-out Diagram



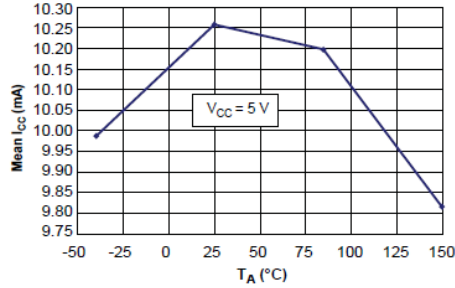
Terminal List Table

Number	Name	Description
1 and 2	IP+	Terminals for current being sampled; fused internally
3 and 4	IP–	Terminals for current being sampled; fused internally
5	GND	Signal ground terminal
6	FILTER	Terminal for external capacitor that sets bandwidth
7	VIOUT	Analog output signal
8	VCC	Device power supply terminal

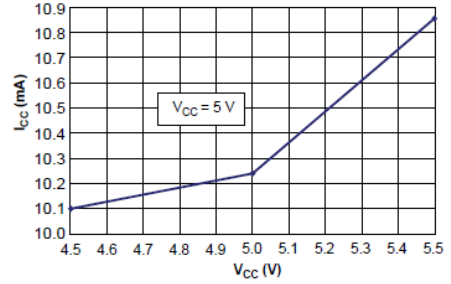
Characteristic Performance

$I_P = 5\text{ A}$, unless otherwise specified

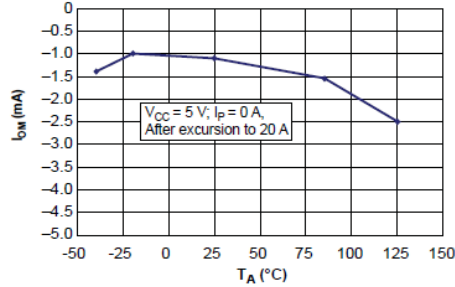
Mean Supply Current versus Ambient Temperature



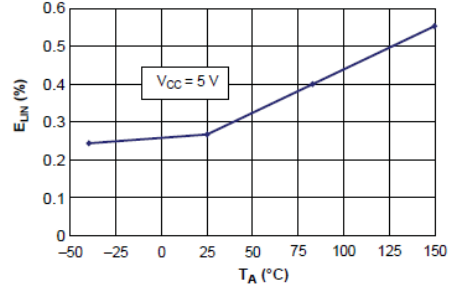
Supply Current versus Supply Voltage



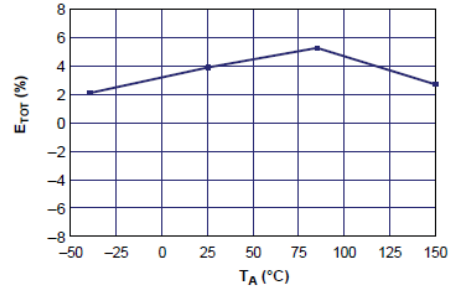
Magnetic Offset versus Ambient Temperature



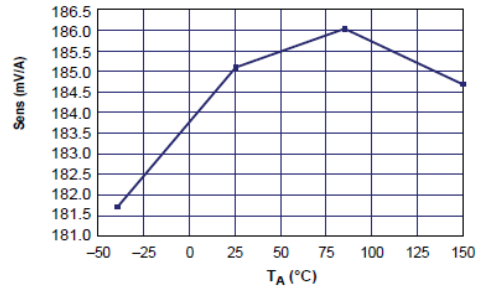
Nonlinearity versus Ambient Temperature



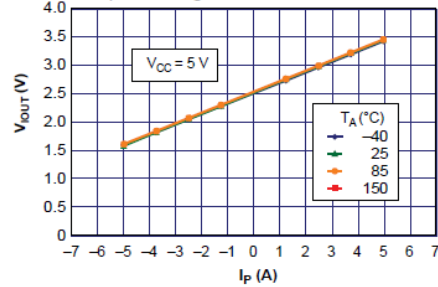
Mean Total Output Error versus Ambient Temperature



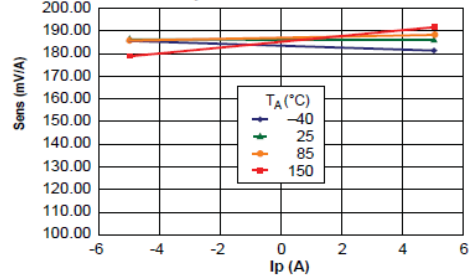
Sensitivity versus Ambient Temperature



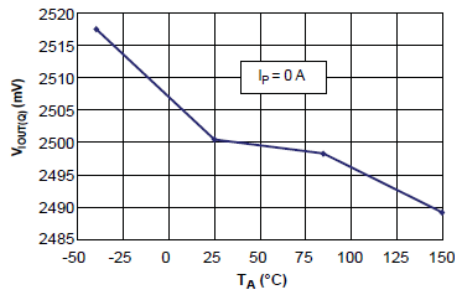
Output Voltage versus Sensed Current



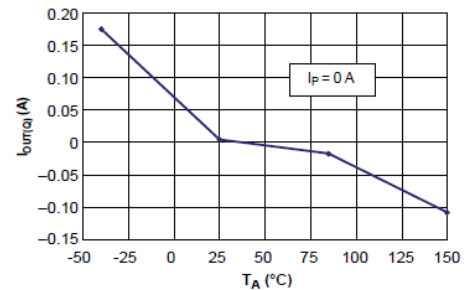
Sensitivity versus Sensed Current



0 A Output Voltage versus Ambient Temperature



0 A Output Voltage Current versus Ambient Temperature



Definitions of Accuracy Characteristics

Sensitivity (Sens). The change in device output in response to a 1 A change through the primary conductor. The sensitivity is the product of the magnetic circuit sensitivity (G/A) and the linear IC amplifier gain (mV/G). The linear IC amplifier gain is programmed at the factory to optimize the sensitivity (mV/A) for the full-scale current of the device.

Noise (V_{NOISE}). The product of the linear IC amplifier gain (mV/G) and the noise floor for the Allegro Hall effect linear IC (≈ 1 G). The noise floor is derived from the thermal and shot noise observed in Hall elements. Dividing the noise (mV) by the sensitivity (mV/A) provides the smallest current that the device is able to resolve.

Linearity (E_{LIN}). The degree to which the voltage output from the IC varies in direct proportion to the primary current through its full-scale amplitude. Nonlinearity in the output can be attributed to the saturation of the flux concentrator approaching the full-scale current. The following equation is used to derive the linearity:

$$100 \left\{ 1 - \left[\frac{\Delta \text{gain} \times \% \text{ sat } (V_{\text{IOUT_full-scale amperes}} - V_{\text{IOUT(Q)}})}{2 (V_{\text{IOUT_half-scale amperes}} - V_{\text{IOUT(Q)}})} \right] \right\}$$

where $V_{\text{IOUT_full-scale amperes}}$ = the output voltage (V) when the sampled current approximates full-scale $\pm I_P$.

Symmetry (E_{SYM}). The degree to which the absolute voltage output from the IC varies in proportion to either a positive or negative full-scale primary current. The following formula is used to derive symmetry:

$$100 \left(\frac{V_{\text{IOUT_+ full-scale amperes}} - V_{\text{IOUT(Q)}}}{V_{\text{IOUT(Q)}} - V_{\text{IOUT_full-scale amperes}}} \right)$$

Quiescent output voltage ($V_{\text{IOUT(Q)}}$). The output of the device when the primary current is zero. For a unipolar supply voltage, it nominally remains at $V_{\text{CC}}/2$. Thus, $V_{\text{CC}} = 5$ V translates into $V_{\text{IOUT(Q)}} = 2.5$ V. Variation in $V_{\text{IOUT(Q)}}$ can be attributed to the resolution of the Allegro linear IC quiescent voltage trim and thermal drift.

Electrical offset voltage (V_{OE}). The deviation of the device output from its ideal quiescent value of $V_{\text{CC}}/2$ due to nonmagnetic causes. To convert this voltage to amperes, divide by the device sensitivity, Sens.

Accuracy (E_{TOT}). The accuracy represents the maximum deviation of the actual output from its ideal value. This is also known as the total output error. The accuracy is illustrated graphically in the output voltage versus current chart at right.

Accuracy is divided into four areas:

- **0 A at 25°C.** Accuracy at the zero current flow at 25°C, without the effects of temperature.
- **0 A over Δ temperature.** Accuracy at the zero current flow including temperature effects.
- **Full-scale current at 25°C.** Accuracy at the the full-scale current at 25°C, without the effects of temperature.
- **Full-scale current over Δ temperature.** Accuracy at the full-scale current flow including temperature effects.

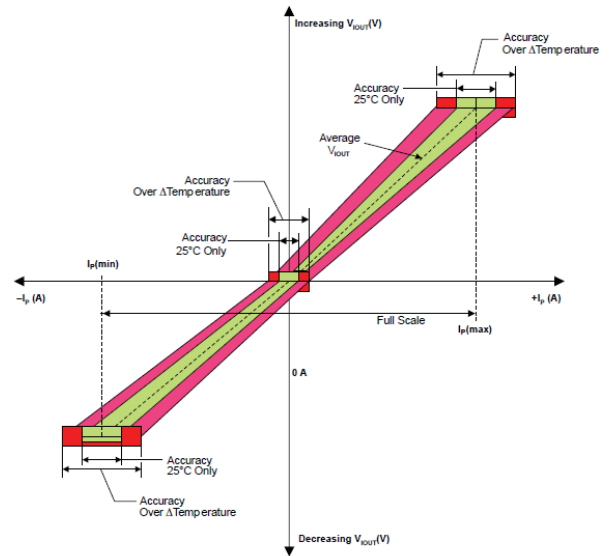
Ratiometry. The ratiometric feature means that its 0 A output, $V_{\text{IOUT(Q)}}$, (nominally equal to $V_{\text{CC}}/2$) and sensitivity, Sens, are proportional to its supply voltage, V_{CC} . The following formula is used to derive the ratiometric change in 0 A output voltage, $\Delta V_{\text{IOUT(Q)RAT}}$ (%).

$$100 \left(\frac{V_{\text{IOUT(Q)VCC}} / V_{\text{IOUT(Q)5V}}}{V_{\text{CC}} / 5 \text{ V}} \right)$$

The ratiometric change in sensitivity, $\Delta \text{Sens}_{\text{RAT}}$ (%), is defined as:

$$100 \left(\frac{\text{Sens}_{\text{VCC}} / \text{Sens}_{5\text{V}}}{V_{\text{CC}} / 5 \text{ V}} \right)$$

Output Voltage versus Sampled Current
Accuracy at 0 A and at Full-Scale Current



Appendix C – MCP3221 datasheet



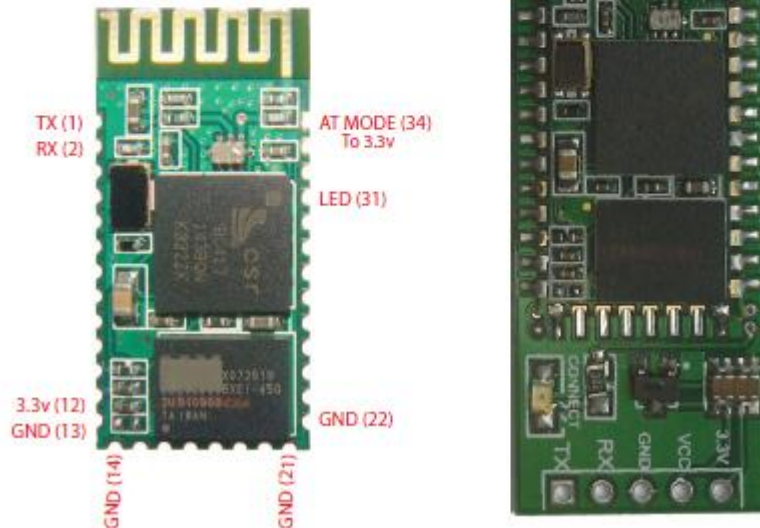
Bluetooth module uses CSR BlueCore4- External chipsets. It embeds 8Mbit flash for software storage, and supports 3.3V power supply.

BC04 is a multi-function module. It can be used in different products according to the embedded firmware settings. It is especially targeted for data transfer. The second generation Bluetooth UART module has two working mode: AT command mode, and automatic binding transparent data mode. In automatic binding data transparent mode, it can be configured to Master, Slave or Loopback three different modes, and it will connect to or be connected by other devices that support SPP protocol per configuration. In AT command mode, user can configure the module and send control commands.

HOW TO ENTER AT command mode

- 1) Bluetooth module of IO pin PIO11 is need to high for AT.

TYPE A



PLEASE READ the Page 18 and 19 for schematics

STEP 1

How to Entering AT-Mode?

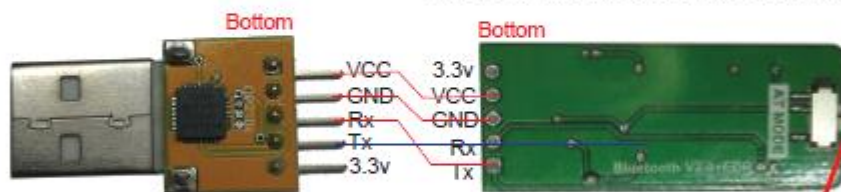
Page 2



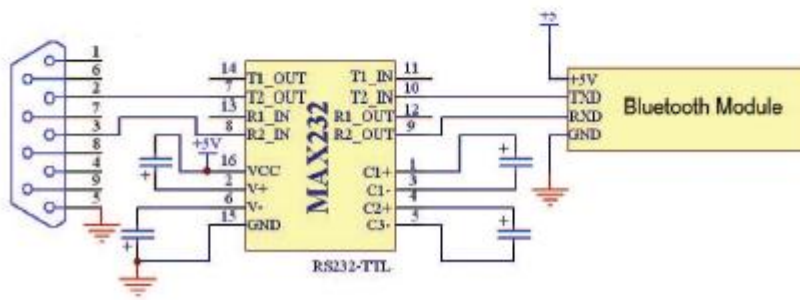
3.3v
TX
RX
GND
5v

USB-TO-USART IN CP2102
on board with 3.3v,5v, TX,RX and GND
for devices.

For AT-Mode, the LED with flash in Slowly.
CONNECTED WITH USB-TO-USART or RS232
Module to Bluetooth Module, PLUG the USB to YOUR PC.



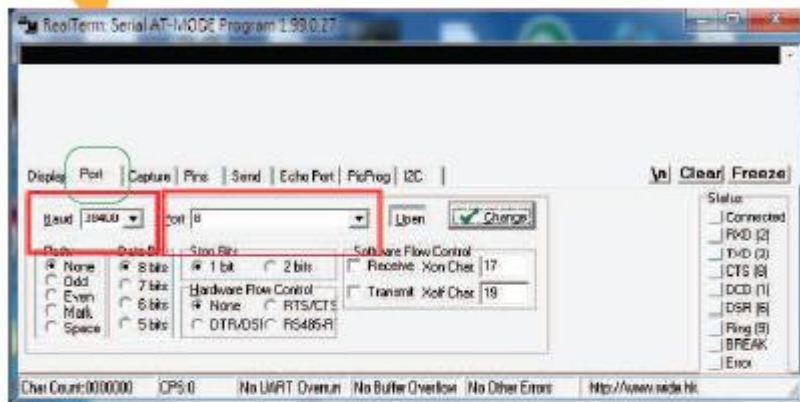
Switch to on for AT mode



STEP 2

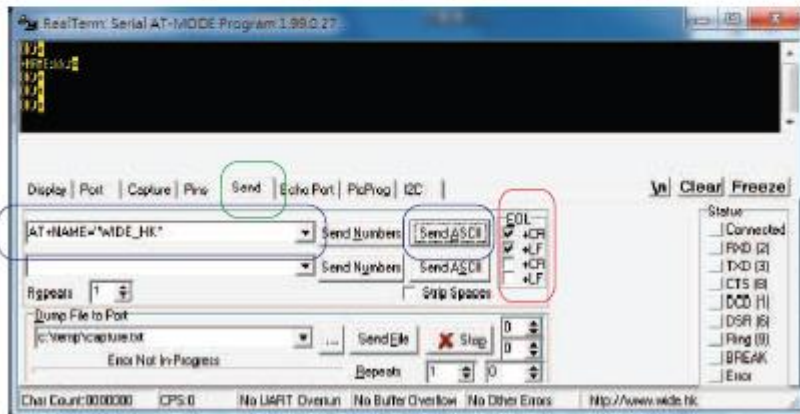
OPEN THE SOFTWARE FOR COMMAND

Page 3



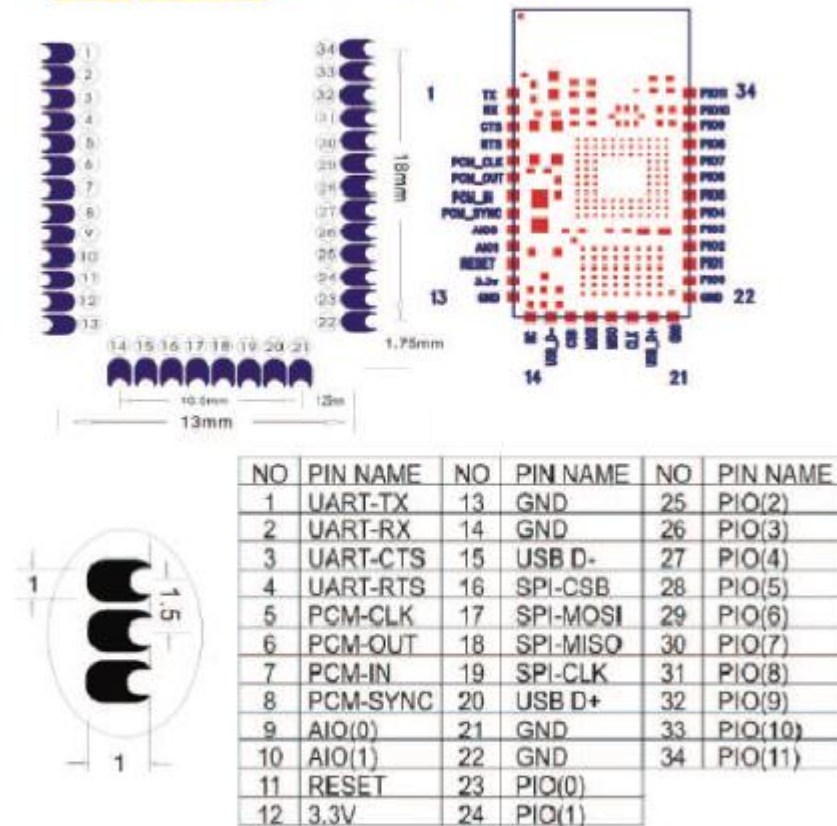
Use hyper terminal or Serial AT-Mode Program software ,SET the Baud for Entering the AT, AT baud rate is "38400", the COM Port for USART, Click "Change"

(set baud rate 38400, data 8 bit, stop bit 1, no parity,no flow control)



Click to SEND tap, IN "EOL", selected the "+CR" and "+LF", NOW you can input the command in line1 or 2 and press "SEND ASCLL", if success, the black screen will show the message.

Specification



Other pins used by Bluetooth UART module:

1. PIO8 is used to control LED indicating the status. It will blink after power on. Different blink intervals are used to indicate different status.
2. PIO9 is used to control LED indicating pairing. It will be steady on when pairing is successful.
3. PIO11 is used to switch the working mode. High level-> AT command mode; Floating or low level-> normal transparent data mode.
4. The module has built-in power on reset circuitry.

Appendix D – MCP3221 datasheet

MCP3221

1.0 ELECTRICAL CHARACTERISTICS

Absolute Maximum Ratings †

V_{DD}	7.0V
Analog input pin w.r.t. V_{SS}	-0.6V to $V_{DD} + 0.6V$
SDA and SCL pins w.r.t. V_{SS}	-0.6V to $V_{DD} + 1.0V$
Storage temperature.....	-65°C to +150°C
Ambient temp. with power applied.....	-65°C to +125°C
Maximum Junction Temperature.....	150°C
ESD protection on all pins (HBM).....	≥ 4 kV

† Stresses above those listed under "Maximum ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at those or any other conditions above those indicated in the operational listings of this specification is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

DC ELECTRICAL SPECIFICATIONS

Electrical Characteristics: Unless otherwise noted, all parameters apply at $V_{DD} = 5.0V$, $V_{SS} = GND$, $R_{PU} = 2 k\Omega$, $T_{AMB} = -40^{\circ}C$ to $+85^{\circ}C$, I^2C Fast Mode Timing: $f_{SCL} = 400 kHz$ (Note 3).

Parameters	Sym	Min	Typ	Max	Units	Conditions
DC Accuracy						
Resolution			12		bits	
Integral Nonlinearity	INL	—	±0.75	±2	LSB	
Differential Nonlinearity	DNL	—	±0.5	±1	LSB	No missing codes
Offset Error		—	±0.75	±2	LSB	
Gain Error		—	-1	±3	LSB	
Dynamic Performance						
Total Harmonic Distortion	THD	—	-82	—	dB	$V_{IN} = 0.1V$ to $4.9V$ @ 1 kHz
Signal-to-Noise and Distortion	SINAD	—	72	—	dB	$V_{IN} = 0.1V$ to $4.9V$ @ 1 kHz
Spurious-Free Dynamic Range	SFDR	—	86	—	dB	$V_{IN} = 0.1V$ to $4.9V$ @ 1 kHz
Analog Input						
Input Voltage Range		$V_{SS}-0.3$	—	$V_{DD}+0.3$	V	$2.7V \leq V_{DD} \leq 5.5V$
Leakage Current		-1	—	+1	μA	
SDA/SCL (open-drain output):						
Data Coding Format		Straight Binary				
High-level input voltage	V_{IH}	$0.7 V_{DD}$	—	—	V	
Low-level input voltage	V_{IL}	—	—	$0.3 V_{DD}$	V	
Low-level output voltage	V_{OL}	—	—	0.4	V	$I_{OL} = 3 mA$, $R_{PU} = 1.53 k\Omega$
Hysteresis of Schmitt trigger inputs	V_{HYST}	—	$0.05 V_{DD}$	—	V	$f_{SCL} = 400 kHz$ only
Input leakage current	I_{LI}	-1	—	+1	μA	$V_{IN} = 0.1 V_{DD}$ and $0.9 V_{DD}$
Output leakage current	I_{LO}	-1	—	+1	μA	$V_{OUT} = 0.1 V_{SS}$ and $0.9 V_{DD}$
Pin capacitance (all inputs/outputs)	C_{IN} , C_{OUT}	—	—	10	pF	$T_{AMB} = 25^{\circ}C$, $f = 1 MHz$; (Note 2)
Bus Capacitance	C_B	—	—	400	pF	SDA drive low, 0.4V

Note 1: "Sample time" is the time between conversions once the address byte has been sent to the converter. Refer to Figure 5-6.

2: This parameter is periodically sampled and not 100% tested.

3: R_{PU} = Pull-up resistor on SDA and SCL.

4: SDA and SCL = V_{SS} to V_{DD} at 400 kHz.

5: t_{ACQ} and t_{CONV} are dependent on internal oscillator timing. See Figure 5-5 and Figure 5-6 for relation to SCL.

MCP3221

DC ELECTRICAL SPECIFICATIONS (CONTINUED)

Electrical Characteristics: Unless otherwise noted, all parameters apply at $V_{DD} = 5.0V$, $V_{SS} = GND$, $R_{PU} = 2\text{ k}\Omega$, $T_{AMB} = -40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$, I ² C Fast Mode Timing: $f_{SCL} = 400\text{ kHz}$ (Note 3).						
Parameters	Sym	Min	Typ	Max	Units	Conditions
Power Requirements						
Operating Voltage	V_{DD}	2.7	—	5.5	V	
Conversion Current	I_{DD}	—	175	250	μA	
Standby Current	$I_{DD(S)}$	—	0.005	1	μA	SDA, SCL = V_{DD}
Active bus current	I_{DDA}	—	—	120	μA	Note 4
Conversion Rate						
Conversion Time	t_{CONV}	—	8.96	—	μs	Note 5
Analog Input Acquisition Time	t_{ACQ}	—	1.12	—	μs	Note 6
Sample Rate	f_{SAMP}	—	—	22.3	ksp/s	$f_{SCL} = 400\text{ kHz}$ (Note 1)

Note 1: "Sample time" is the time between conversions once the address byte has been sent to the converter. Refer to Figure 5-6.

2: This parameter is periodically sampled and not 100% tested.

3: R_{PU} = Pull-up resistor on SDA and SCL.

4: SDA and SCL = V_{SS} to V_{DD} at 400 kHz.

5: t_{ACQ} and t_{CONV} are dependent on internal oscillator timing. See Figure 5-5 and Figure 5-6 for relation to SCL.

TEMPERATURE SPECIFICATIONS

Electrical Characteristics: Unless otherwise noted, all parameters apply at $V_{DD} = 5.0V$, $V_{SS} = GND$.						
Parameters	Symbol	Min	Typ	Max	Units	Conditions
Temperature Ranges						
Industrial Temperature Range	T_A	-40	—	+85	$^{\circ}\text{C}$	
Extended Temperature Range	T_A	-40	—	+125	$^{\circ}\text{C}$	
Operating Temperature Range	T_A	-40	—	+125	$^{\circ}\text{C}$	
Storage Temperature Range	T_A	-65	—	+150	$^{\circ}\text{C}$	
Thermal Package Resistances						
Thermal Resistance, 5L-SOT23	θ_{JA}	—	256	—	$^{\circ}\text{C/W}$	

MCP3221

TIMING SPECIFICATIONS

Electrical Characteristics: All parameters apply at $V_{DD} = 2.7V - 5.5V$, $V_{SS} = GND$, $T_{AMB} = -40^{\circ}C$ to $+85^{\circ}C$.

Parameters	Sym	Min	Typ	Max	Units	Conditions
PC Standard Mode						
Clock frequency	f_{SCL}	0	—	100	KHz	
Clock high time	T_{HIGH}	4000	—	—	ns	
Clock low time	T_{LOW}	4700	—	—	ns	
SDA and SCL rise time	T_R	—	—	1000	ns	From V_L to V_{IH} (Note 1)
SDA and SCL fall time	T_F	—	—	300	ns	From V_H to V_{IL} (Note 1)
START condition hold time	T_{HD-STA}	4000	—	—	ns	
START condition setup time	T_{SU-STA}	4700	—	—	ns	
Data input setup time	T_{SU-DAT}	250	—	—	ns	
STOP condition setup time	T_{SU-STD}	4000	—	—	ns	
STOP condition hold time	T_{HD-STD}	4000	—	—	ns	
Output valid from clock	T_{AA}	—	—	3500	ns	
Bus free time	T_{BUF}	4700	—	—	ns	Note 2
Input filter spike suppression	T_{SP}	—	—	50	ns	SDA and SCL pins (Note 1)
PC Fast Mode						
Clock frequency	f_{SCL}	0	—	400	KHz	
Clock high time	T_{HIGH}	600	—	—	ns	
Clock low time	T_{LOW}	1300	—	—	ns	
SDA and SCL rise time	T_R	$20 + 0.1C_B$	—	300	ns	From V_L to V_{IH} (Note 1)
SDA and SCL fall time	T_F	$20 + 0.1C_B$	—	300	ns	From V_H to V_{IL} (Note 1)
START condition hold time	T_{HD-STA}	600	—	—	ns	
START condition setup time	T_{SU-STA}	600	—	—	ns	
Data input hold time	T_{HD-DAT}	0	—	0.5	ns	
Data input setup time	T_{SU-DAT}	100	—	—	ns	
STOP condition setup time	T_{SU-STD}	600	—	—	ns	
STOP condition hold time	T_{HD-STD}	600	—	—	ns	
Output valid from clock	T_{AA}	—	—	900	ns	
Bus free time	T_{BUF}	1300	—	—	ns	Note 2
Input filter spike suppression	T_{SP}	—	—	50	ns	SDA and SCL pins (Note 1)

Note 1: This parameter is periodically sampled and not 100% tested.

Note 2: Time the bus must be free before a new transmission can start.

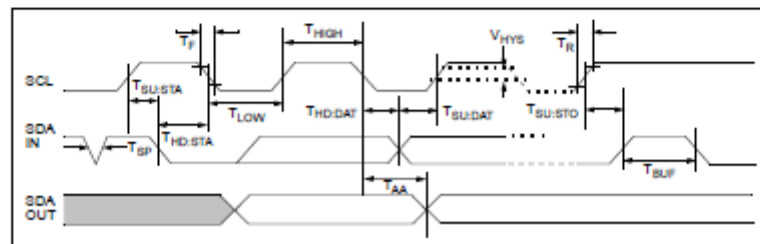


FIGURE 1-1: Standard and Fast Mode Bus Timing Data.

2.0 TYPICAL PERFORMANCE CURVES

Note: The graphs and tables provided following this note are a statistical summary based on a limited number of samples and are provided for informational purposes only. The performance characteristics listed herein are not tested or guaranteed. In some graphs or tables, the data presented may be outside the specified operating range (e.g., outside specified power supply range) and therefore outside the warranted range.

Note: Unless otherwise indicated, $V_{DD} = 5V$, $V_{SS} = 0V$, I^2C Fast Mode Timing ($SCL = 400$ kHz), Continuous Conversion Mode ($f_{SAMP} = 22.3$ ksp/s), $T_A = +25^\circ C$.

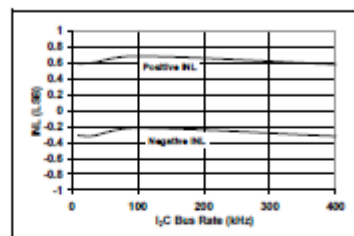
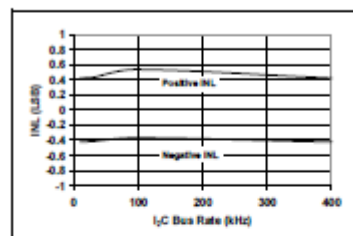
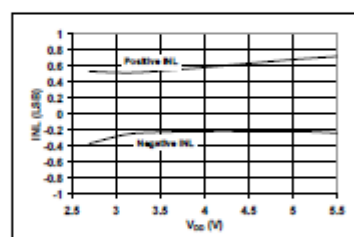
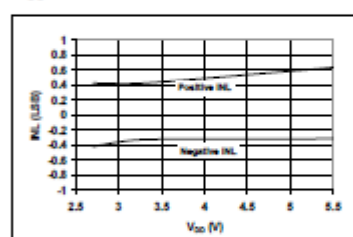
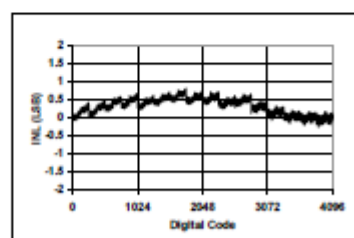
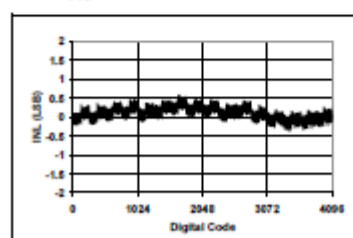


FIGURE 2-1: INL vs. Clock Rate.

FIGURE 2-4: INL vs. Clock Rate
($V_{DD} = 2.7V$).FIGURE 2-2: INL vs. V_{DD} - I^2C^{TM}
Standard Mode ($f_{SCL} = 100$ kHz).FIGURE 2-5: INL vs. V_{DD} - I^2C^{TM} Fast
Mode ($f_{SCL} = 400$ kHz).FIGURE 2-3: INL vs. Code
(Representative Part).FIGURE 2-8: INL vs. Code
(Representative Part, $V_{DD} = 2.7V$).

MCP3221

Note: Unless otherwise indicated, $V_{DD} = 5V$, $V_{SS} = 0V$, I^2C Fast Mode Timing (SCL = 400 kHz), Continuous Conversion Mode ($f_{SAMPL} = 22.3$ ksp/s), $T_A = +25^\circ C$.

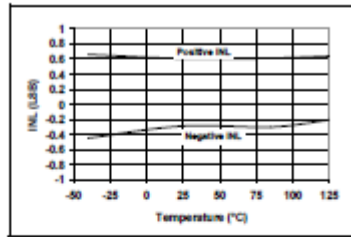


FIGURE 2-7: INL vs. Temperature.

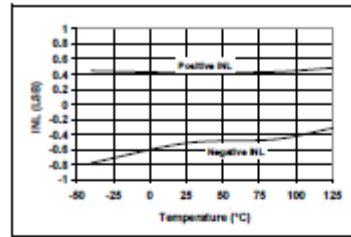


FIGURE 2-10: INL vs. Temperature ($V_{DD} = 2.7V$).

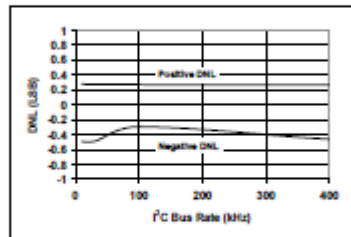


FIGURE 2-8: DNL vs. Clock Rate.

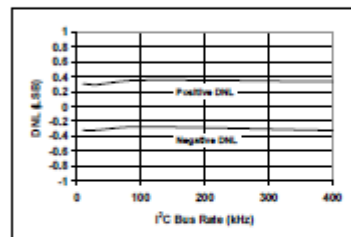


FIGURE 2-11: DNL vs. Clock Rate ($V_{DD} = 2.7V$).

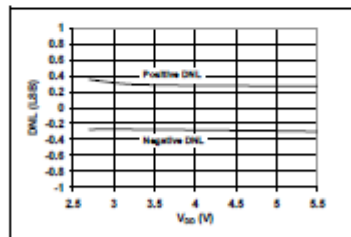


FIGURE 2-9: DNL vs. $V_{DD} - I^2C^{\text{TM}}$ Standard Mode ($f_{SCL} = 100$ kHz).

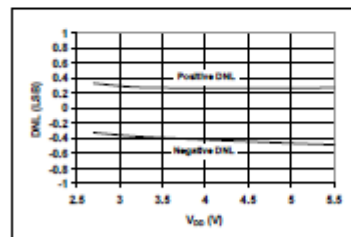


FIGURE 2-12: DNL vs. $V_{DD} - I^2C^{\text{TM}}$ Fast Mode ($f_{SCL} = 400$ kHz).

Note: Unless otherwise indicated, $V_{DD} = 5V$, $V_{SS} = 0V$, I²C Fast Mode Timing (SCL = 400 kHz), Continuous Conversion Mode ($f_{SAMP} = 22.3$ ksp/s), $T_A = +25^{\circ}C$.

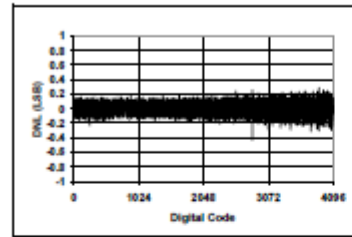


FIGURE 2-13: DNL vs. Code (Representative Part).

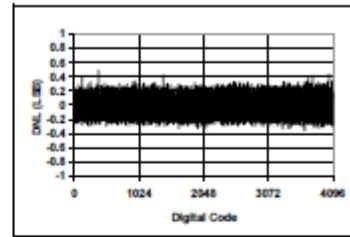


FIGURE 2-16: DNL vs. Code (Representative Part, $V_{DD} = 2.7V$).

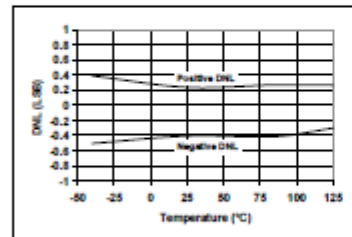


FIGURE 2-14: DNL vs. Temperature.

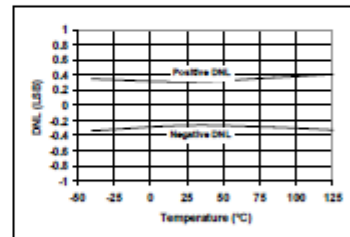


FIGURE 2-17: DNL vs. Temperature ($V_{DD} = 2.7V$).

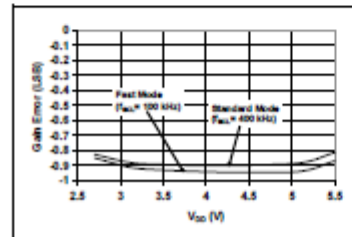


FIGURE 2-15: Gain Error vs. V_{DD} .

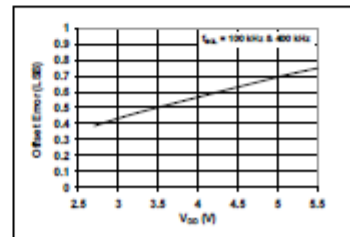


FIGURE 2-18: Offset Error vs. V_{DD} .

Note: Unless otherwise indicated, $V_{DD} = 5V$, $V_{SS} = 0V$, I^2C Fast Mode Timing (SCL = 400 kHz), Continuous Conversion Mode ($f_{SAMP} = 22.3$ kbps), $T_A = +25^\circ C$.

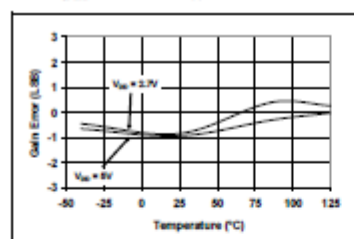


FIGURE 2-19: Gain Error vs. Temperature.

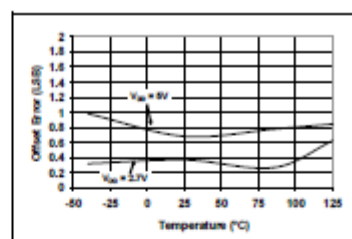


FIGURE 2-22: Offset Error vs. Temperature.

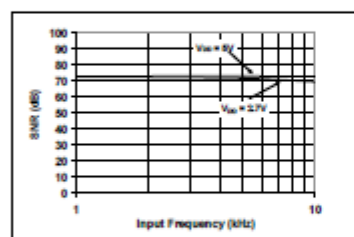


FIGURE 2-20: SNR vs. Input Frequency.

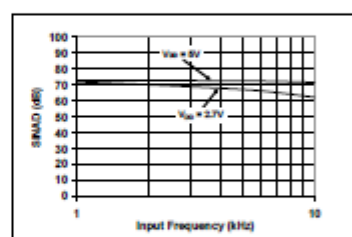


FIGURE 2-23: SINAD vs. Input Frequency.

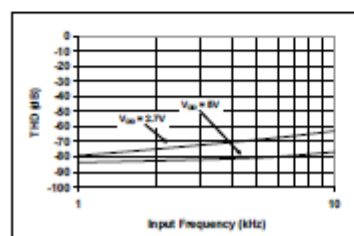


FIGURE 2-21: THD vs. Input Frequency.

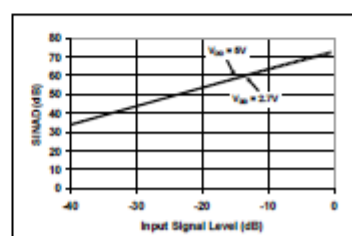


FIGURE 2-24: SINAD vs. Input Signal Level.

Note: Unless otherwise indicated, $V_{DD} = 5V$, $V_{SS} = 0V$, \dot{I}^2C Fast Mode Timing (SCL = 400 kHz), Continuous Conversion Mode ($f_{SAMP} = 22.3$ ksp/s), $T_A = +25^\circ C$.

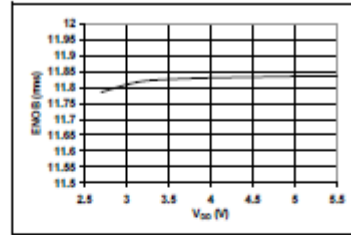


FIGURE 2-25: ENOB vs. V_{DD} .

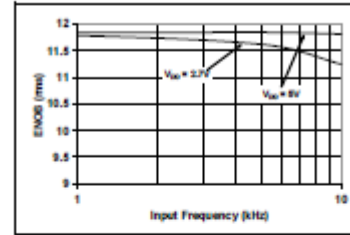


FIGURE 2-26: ENOB vs. Input Frequency.

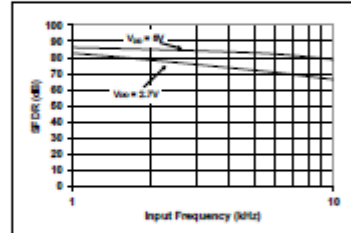


FIGURE 2-27: SFDR vs. Input Frequency.

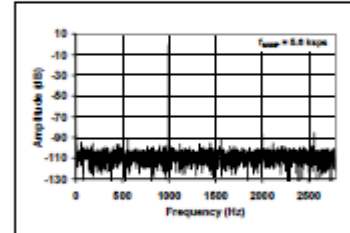


FIGURE 2-28: Spectrum Using \dot{I}^2C Standard Mode (Representative Part, 1 kHz Input Frequency).

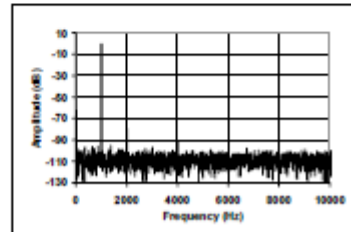


FIGURE 2-29: Spectrum Using \dot{I}^2C Fast Mode (Representative Part, 1 kHz Input Frequency).

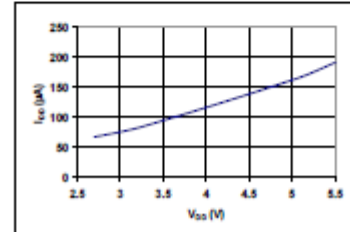


FIGURE 2-30: I_{DD} (Conversion) vs. V_{DD} .

MCP3221

Note: Unless otherwise indicated, $V_{DD} = 5V$, $V_{SS} = 0V$, I^2C Fast Mode Timing (SCL = 400 kHz), Continuous Conversion Mode ($f_{SAMP} = 22.3$ ksp/s), $T_A = +25^\circ C$.

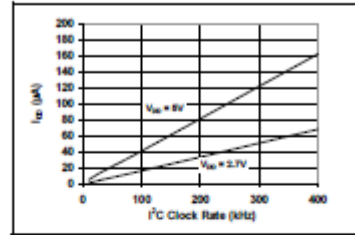


FIGURE 2-31: I_{DD} (Conversion) vs. Clock Rate.

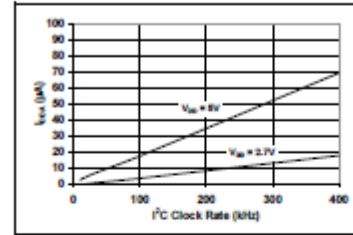


FIGURE 2-34: I_{DDA} (Active Bus) vs. Clock Rate.

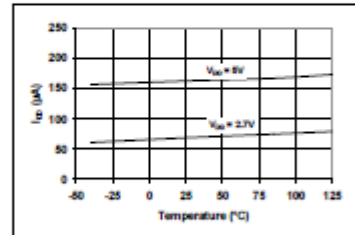


FIGURE 2-32: I_{DD} (Conversion) vs. Temperature.

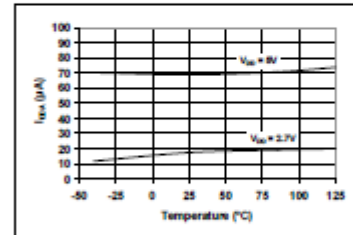


FIGURE 2-35: I_{DDA} (Active Bus) vs. Temperature.

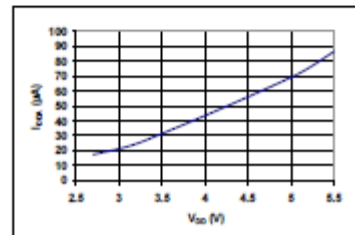


FIGURE 2-33: I_{DDA} (Active Bus) vs. V_{DD} .

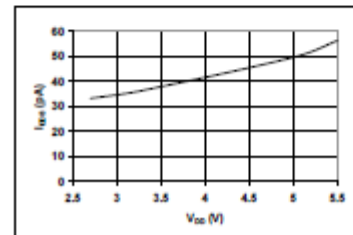


FIGURE 2-36: I_{DDS} (Standby) vs. V_{DD} .

MCP3221

Note: Unless otherwise indicated, $V_{DD} = 5V$, $V_{SS} = 0V$, I^2C Fast Mode Timing (SCL = 400 kHz), Continuous Conversion Mode ($f_{SAMP} = 22.3$ ksp/s), $T_A = +25^\circ C$.

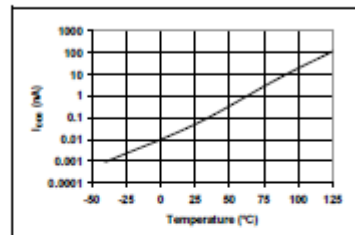


FIGURE 2-37: I_{DDS} (Standby) vs. Temperature.

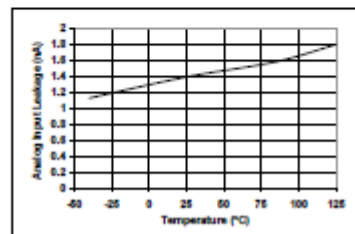


FIGURE 2-38: Analog Input Leakage vs. Temperature.

2.1 Test Circuits

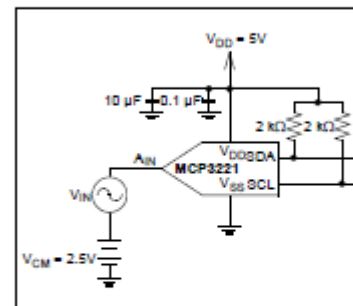


FIGURE 2-39: Typical Test Configuration.

MCP3221

3.0 PIN FUNCTIONS

TABLE 3-1: PIN FUNCTION TABLE

Name	Function
V _{DD}	+2.7V to 5.5V Power Supply
V _{SS}	Ground
A _{IN}	Analog Input
SDA	Serial Data In/Out
SCL	Serial Clock In

3.1 V_{DD} and V_{SS}

The V_{DD} pin, with respect to V_{SS}, provides power to the device as well as a voltage reference for the conversion process. Refer to Section 8.4 "Device Power and Layout Considerations", "Device Power and Layout Considerations", for tips on power and grounding.

3.2 Analog Input (A_{IN})

A_{IN} is the input pin to the sample-and-hold circuitry of the Successive Approximation Register (SAR) converter. Care should be taken in driving this pin. Refer to Section 8.1 "Driving the Analog Input", "Driving the Analog Input". For proper conversions, the voltage on this pin can vary from V_{SS} to V_{DD}.

3.3 Serial Data (SDA)

SDA is a bidirectional pin used to transfer addresses and data into and out of the device. Since it is an open-drain terminal, the SDA bus requires a pull-up resistor to V_{DD} (typically 10 kΩ for 100 kHz and 2 kΩ for 400 kHz SCL clock speeds). Refer to Section 8.2 "Connecting to the I²C Bus", "Connecting to the I²C Bus", for more information.


For normal data transfer, SDA is allowed to change only during SCL low. Changes during SCL high are reserved for indicating the START and STOP conditions. Refer to Section 5.1 "I²C Bus Characteristics", "I²C Bus Characteristics".

3.4 Serial Clock (SCL)

SCL is an input pin used to synchronize the data transfer to and from the device on the SDA pin and is an open-drain terminal. Therefore, the SCL bus requires a pull-up resistor to V_{DD} (typically 10 kΩ for 100 kHz and 2 kΩ for 400 kHz SCL clock speeds). Refer to Section 8.2 "Connecting to the I²C Bus", "Connecting to the I²C Bus".


For normal data transfer, SDA is allowed to change only during SCL low. Changes during SCL high are reserved for indicating the START and STOP conditions. Refer to Section 8.1 "Driving the Analog Input", "Driving the Analog Input".

Appendix E –MB1S - MB8S Bridge Rectifiers datasheet




MB1S - MB8S
Bridge Rectifiers

May 2009

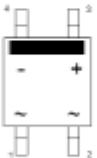


Features

- Low leakage
- Surge overload rating : 95 amperes peak.
- Ideal for printed circuit board.
- UL certified, UL #E111753 and E326243.



SOIC-4
Polarity symbols molded
or marking on body



Absolute Maximum Ratings * $T_A = 25^\circ\text{C}$ unless otherwise noted

Symbol	Parameter	Value					Units
		1S	2S	4S	6S	8S	
V_{RRM}	Maximum Repetitive Reverse Voltage	100	200	400	600	800	V
V_{RMS}	Maximum RMS Bridge Input Voltage	70	140	280	420	560	V
V_R	DC Reverse Voltage (Rated V_R)	100	200	400	600	800	V
I_{AVG}	Average Rectified Forward Current @ $T_A = 50^\circ\text{C}$	0.5					A
I_{FSM}	Non-Repetitive Peak Forward Surge Current 8.3 ms Single Half-Sine-Wave	95					A
T_{STG}	Storage Temperature Range	-55 to +150					$^\circ\text{C}$
T_J	Operating Junction Temperature	-55 to +150					$^\circ\text{C}$

* These ratings are limiting values above which the serviceability of any semiconductor device may be impaired.

Thermal Characteristics

Symbol	Parameter	Value	Units
P_D	Power Dissipation	1.4	W
$R_{\theta JA}$	Thermal Resistance, Junction to Ambient,* per leg	85	$^\circ\text{C/W}$
$R_{\theta JL}$	Thermal Resistance, Junction to Lead,* per leg	20	$^\circ\text{C/W}$

* Device mounted on PCB with 0.5-0.5" (13x13 mm) lead length.

Electrical Characteristics $T_A = 25^\circ\text{C}$ unless otherwise noted

Symbol	Parameter	Value	Units
V_F	Forward Voltage, per bridge @ 0.5 A	1.0	V
I_R	Reverse Current, per leg @ Rated V_R	5.0 0.5	μA mA
	I^2t rating for fusing $t < 8.3$ ms	5.0	A^2s
C_T	Total Capacitance, per leg $V_R = 4.0\text{V}$, $f = 1.0\text{MHz}$	13	pF

© 2009 Fairchild Semiconductor Corporation
MB1S - MB8S Rev. D1
1
www.fairchildsemi.com

MB1S - MB8S — Bridge Rectifiers

Appendix F – Arduino code

```
// include the library code:
#include <LiquidCrystal.h>
```

```
// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(7, 6, 5, 4, 3, 2);
```

```
int AN0 = 0;
int AN1 = 0;
unsigned long time = 0;
```

```

float Voltage = 0;
float Current = 0;
float Power = 0;
int Unit = 0;

void setup() {

  lcd.begin(16, 2);
  Serial.begin(115200);

}

void loop() {

  AN0 = analogRead(A0);
  AN1 = analogRead(A1);
  time++;

  Voltage = (((((float)AN0 * 5.0)/1023.0)*90.0)/1.414) ;
  Current = ((2.50 - (((float)AN1 * 5.0)/1023.0)) * 2.0);

  if(Current <= 0.03){
    Current = 0.00;
  }

  else{
    Current = Current;
  }

  //Power = ((Power + ((Voltage * Current) * (1.0/3600.0)))/1000.0);
  Power = (Power + ((Voltage * Current) * (1.0/3600.0)));

  Unit = Power/1000;

  Serial.print(Voltage);
  Serial.print(",");
  Serial.print(Current);
  Serial.print(",");
  Serial.print(time);
  Serial.print(",");
  Serial.print(Power/1000);
  Serial.print(",");
  Serial.println(Unit);

  lcd.setCursor(0, 0);
  lcd.print(Voltage,2);

  lcd.setCursor(7, 0);

```



```

lcd.print("V");

lcd.setCursor(10, 0);
lcd.print(Current,2);

lcd.setCursor(15, 0);
lcd.print("A");

lcd.setCursor(0, 1);
lcd.print(Power/1000,2);

lcd.setCursor(5, 1);
lcd.print("KWh");

lcd.setCursor(10, 1);
lcd.print(time);

lcd.setCursor(15, 1);
lcd.print("S");

delay(1000);
}

```

Appendix G – Android code

```

package com.sensor.module;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.UUID;
import java.util.Vector;
import android.os.Bundle;
import android.os.Handler;
//import android.os.SystemClock;
import android.util.Log;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;

```

```

import android.widget.Spinner;

import android.widget.TextView;

import android.widget.Toast;

import android.app.Activity;

import android.bluetooth.BluetoothAdapter;

import android.bluetooth.BluetoothDevice;

import android.bluetooth.BluetoothSocket;

import android.content.BroadcastReceiver;

import android.content.Context;

import android.content.Intent;

import android.content.IntentFilter;


public class MainActivity extends Activity {


    // --    Log

    private static final String TAG = "Bluetooth-Main";

    private static final boolean D = true;

    // --    DataClass

    private IncommingData data = new IncommingData();

    // --    Blue_tooth

    private BluetoothAdapter mBluetoothAdapter;

    private BluetoothDevice btDevice;

    private BluetoothSocket clientSock;

    // --    Integers

    private static final int REQUEST_ENABLE_BT = 3;

    // --    ArrayAdapter

    private ArrayAdapter<CharSequence> mArrayAdapter;

```

```

// -- OS Handler

private Handler spinner_handler;

private Handler gui_handler;

// -- Vector

private Vector<String> dev_id;

// -- Widgets

private static Spinner spinner;

private TextView[] textViewArray;

// -- Boolean

public boolean isDevConnected = false;

public boolean isDevDisconnected = false;

public boolean dataReady = false;

// -- BroadcastReceiver

private BroadcastReceiver mReceiver;

@Override

public void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);

    setContentView(R.layout.activity_main);

if(D) Log.d(TAG, "+++ ON CREATE +++");

        mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();

        mArrayAdapter = new ArrayAdapter<CharSequence>(this,
android.R.layout.simple_spinner_item);

        mArrayAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropd
own_item);

spinner_handler = new Handler();

        gui_handler = new Handler();

        textViewArray = new TextView[5];

        textViewArray[0] = (TextView) findViewById(R.id.textView1);

```

```

textViewArray[1] = (TextView) findViewById(R.id.textView2);
textViewArray[2] = (TextView) findViewById(R.id.textView3);
textViewArray[3] = (TextView) findViewById(R.id.textView4);
textViewArray[4] = (TextView) findViewById(R.id.textView5);

dev_id = new Vector<String>();

if(mBluetoothAdapter == null) {

    Toast.makeText(this, "Bluetooth is not available on your device",
Toast.LENGTH_LONG).show();

}

mReceiver = new BroadcastReceiver()    {

    @Override

    public void onReceive(Context context, Intent intent) {

        String action = intent.getAction();

        if(BluetoothDevice.ACTION_FOUND.equals(action)){

            BluetoothDevice device =
intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);

            mArrayAdapter.add(device.getName());

            dev_id.add(device.getAddress());

            Toast.makeText(context, "Found Device
"+device.getName(), Toast.LENGTH_LONG).show();

            Log.v(TAG, "*** Found Device "+device.getName());

            spinner_handler.post(new Runnable() {

                Spinner spinner = (Spinner)
findViewById(R.id.spinner1);

                public void run() {

                    spinner.setAdapter(mArrayAdapter);

                }
            });
        }
    }
}

```

```

        });

    }

}

};

IntentFilter filter = new IntentFilter(BluetoothDevice.ACTION_FOUND);
registerReceiver(mReceiver, filter);

}

@Override

protected void onStart() {

    super.onStart();

    if(D) Log.d(TAG, "+++ ON START +++");

    if(!mBluetoothAdapter.isEnabled()){

        Intent enableIntent = new
Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);

        startActivityForResult(enableIntent, REQUEST_ENABLE_BT);

    }

    setSearchButtonClickActionListener();

    setSpinnerListener();

    setConnectButtonClickActionListener();

}

@Override

protected void onPause() {

    super.onPause();

    if(D) Log.d(TAG, "+++ ON PAUSE +++");

}

@Override

protected void onStop() {

```

```

        super.onStop();

        if(D) Log.d(TAG, "+++ ON STOP +++");

        unregisterReceiver(mReceiver);
    }

    //    ** Action Listeners **

    private void setSearchButtonClickActionListener(){

        if(D) Log.d(TAG, "--- SEARCH_BUTTON_LISTNER ---");

        Button searchButton = (Button) findViewById(R.id.button1);

        searchButton.setOnClickListener(new View.OnClickListener() {

            public void onClick(View arg0) {

                Log.v(TAG, "**** Search Button Clicked.");

                mAdapter.clear();

                dev_id.clear();

                mBluetoothAdapter.startDiscovery();

            }

        });
    }

    private void setConnectButtonClickActionListener(){

        if(D) Log.d(TAG, "--- CONNECT_BUTTON_LISTNER ---");

        final Button connectButton = (Button) findViewById(R.id.button2);

        connectButton.setOnClickListener(new View.OnClickListener() {

            public void onClick(View v) {

                if(isDevConnected){

                    BTDisconnect(connectButton);

                    connectButton.setText("Connect");

                } else{

```

```

        if(BTConnect(connectButton)){

            connectButton.setText("Disconnect");

            Updator();

            dataGrabber();

        }

    }

});

}

private void setSpinnerListener(){

    if(D) Log.d(TAG, "--- SPINNER_LISTENER ---");

    spinner = (Spinner) findViewById(R.id.spinner1);

    spinner.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {

public void onItemSelected(AdapterView<?> parent, View view, int pos, long id) {

        String name = (String) parent.getSelectedItem();

        int location = mAdapterAdapter.getPosition(name);

        String address = dev_id.elementAt(location);

        Log.v(TAG, "Address Selected "+address);

        btDevice = mBluetoothAdapter.getRemoteDevice(address);

    }

    public void onNothingSelected(AdapterView<?> arg0) {

    }

});

}

// ** BLUETOOTH **

private boolean BTConnect(Button b){

```

```

        Log.d(TAG, "+++ Bluetooth Connecting +++");

        mBluetoothAdapter.cancelDiscovery();

        try {

            clientSock =
btDevice.createRfcommSocketToServiceRecord(UUID.fromString("00001101-0000-1000-
8000-00805F9B34FB"));

        } catch (IOException e) {

            Log.e(TAG, "Error during connection",e);

            clientSock = null;

        }

        if(clientSock == null){

            Log.e(TAG, "Error while establishing connection: no socket");

            isDevConnected = false;

            return false;

        }
    else{

        Log.d(TAG, " --- Connecting to the socket ---");

        try {

            clientSock.connect();

            Log.d(TAG, " --- Connected to the socket ---");

            Toast.makeText(this, "Connecting to the "+btDevice.getName(),
Toast.LENGTH_LONG).show();

            isDevConnected = true;

            return true;

        } catch(IOException e){

            b.setText("Connect");

            Log.e(TAG, "Connecting failed",e);

            isDevConnected = false;

```



```

        return false;
    }
}

private void BTDisconnect(Button b){
    Log.d(TAG, "+++ Bluetooth Disconnecting +++");

    if(clientSock != null){
        try {
            clientSock.close();

            isDevConnected = false;

            Toast.makeText(this, "Disconnecting from the
            "+btDevice.getName(), Toast.LENGTH_LONG).show();

            Log.v(TAG, "+++ Bluetooth Disconnecting Succesfull +++");
        } catch (IOException e) {
            e.printStackTrace();

            isDevConnected = true;

            b.setText("Disconnect");

            Log.v(TAG, "+++ Bluetooth Disconnecting Unsuccesfull
            +++");
        }
    }
}

private void listenForMessage(){
    String result = "";

    Log.d(TAG, " --- Listen For The Message --- ");

    try {

```

```

        InputStreamReader streamReader = new
InputStreamReader(clientSock.getInputStream(),"US-ASCII");

        BufferedReader reader = new BufferedReader(streamReader);


        //long now = SystemClock.uptimeMillis();

        //long lastRead = now;

        while(isDevConnected){

            if(reader.ready()){

                result = reader.readLine();

                Log.d(TAG, "reading successfull data : "+result);

                data.run(result);

                dataReady = true;

                //lastRead = SystemClock.uptimeMillis();

            }

            //else{

                //Log.d(TAG, "reader not ready

"+System.currentTimeMillis());

                //SystemClock.sleep(500);

            //    }

            //    now = SystemClock.uptimeMillis();

            /*

            try {

                Thread.sleep(1000);

            } catch (InterruptedException e) {

                e.printStackTrace();

            }*/

        }

```

```

        } catch (IOException e) {

            e.printStackTrace();

            Log.e(TAG, " *** reading failed ***",e);

        }

    }

    //    ** THREADS **

    private void dataGrabber(){

        new Thread(new Runnable(){

            public void run(){

                listenForMessage();

            }

        }).start();

    }

    private void Updator(){

        new Thread(new Runnable(){

            public void run(){

                Log.d(TAG, "--- UPDATOR THREAD ---");

                while(isDevConnected){

                    gui_handler.post(new Runnable(){

                        public void run() {

                            if(dataReady){

                                for(int j=0; j<textViewArray.length;

j++){

                                    textViewArray[j].setText(data.getChunk(j));

                                }

                            }

                        }

                    });

                }

            }

        });

    }

```

```

        }

    });

}

}).start();

}

}

// -- Data Class

class IncommingData{

    String[] chunks = new String[5];

    public IncommingData(){

        for(int i=0; i<chunks.length; i++){

            chunks[i] = " ";

        }

    }

    public void run(String rawData){

        String[] temp = null;

        try{

            temp = rawData.split(",");

        } catch(ArrayIndexOutOfBoundsException e) {

            e.printStackTrace();

        }

        if(temp.length ==5){

            for(int i=0; i<chunks.length; i++){

                chunks[i] = temp[i];

            }

        }

    }

}

```

```
        }  
    }  
  
    public String getChunk(int index){  
        return chunks[index];  
    }  
}
```