ZSCircuits Electronic Systems
ZS-1100-A
IOT Power Meter: Interface Control Document Ver 0.1

Author : **Prajay Madhavan**

Date : **07-Aug-2020**

Email : **support@zscircuits.in**

Contents

1. Outline

This document details the USB interface control of the ZS-1100-A. The information contained in this document could be used to develop a graphical user interface program or a data acquisition program for the ZS-1100-A. This document is provided as is without any explicit warranty whatsoever. ZSCircuits assumes no responsibility to the damage to the equipment or loss of data resulting from the software made using this document.

2. Introduction

The ZS1100A is a programmable power supply with high speed and precision current measurement capability. The tool uses precision analog techniques on the hardware and DSP algorithms on the PC software to enhance the dynamic range of the measurement from few uA to about 1.5A. The Hardware acts only as a data capture device which simplifies the overall design substantially. The FT2232H device acts as a USB to FIFO bridge to send the data captured by the ADC to the PC in real time. The GUI running on the PC captures this data and processes them to display the current and voltage waveforms.

3. Hardware Architecture

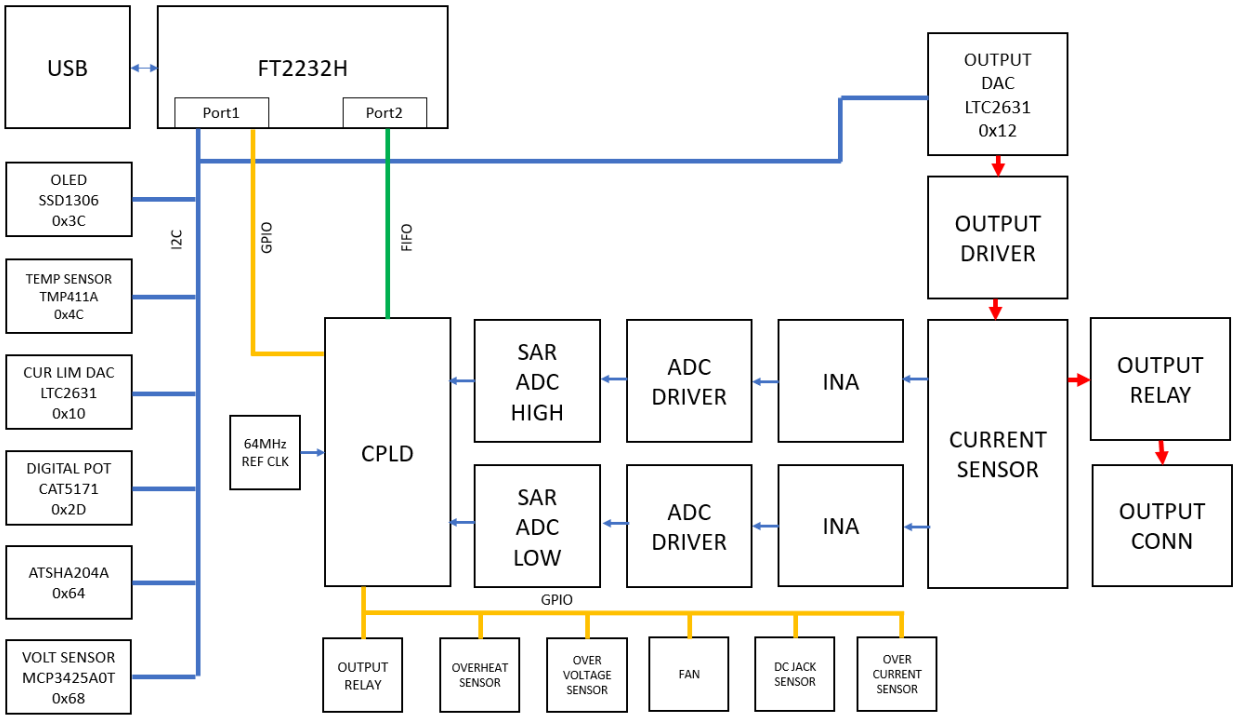


Figure 1 : ZS1100A Hardware Architecture

The above figure details the hardware architecture of the ZS1100A which is relevant to the interface control. Apart from this, there are other hardware features on the board including Power supplies (LDO & DCDC) which are not detailed here as they are not programmable. Please refer to the board schematics for detailed explanation.

4. Functional description

4.1. Principle of operation

The simplified output stage of the hardware is shown below. Essentially it consists of an opamp used with an push pull stage with a 1 Ohm precision resistor in series with the load. Since the input bias current of the opamp is well below 1nA, the entire load current passes through the sense resistor. The voltage measured across the sense resistor is a direct indication of the load current in this case.

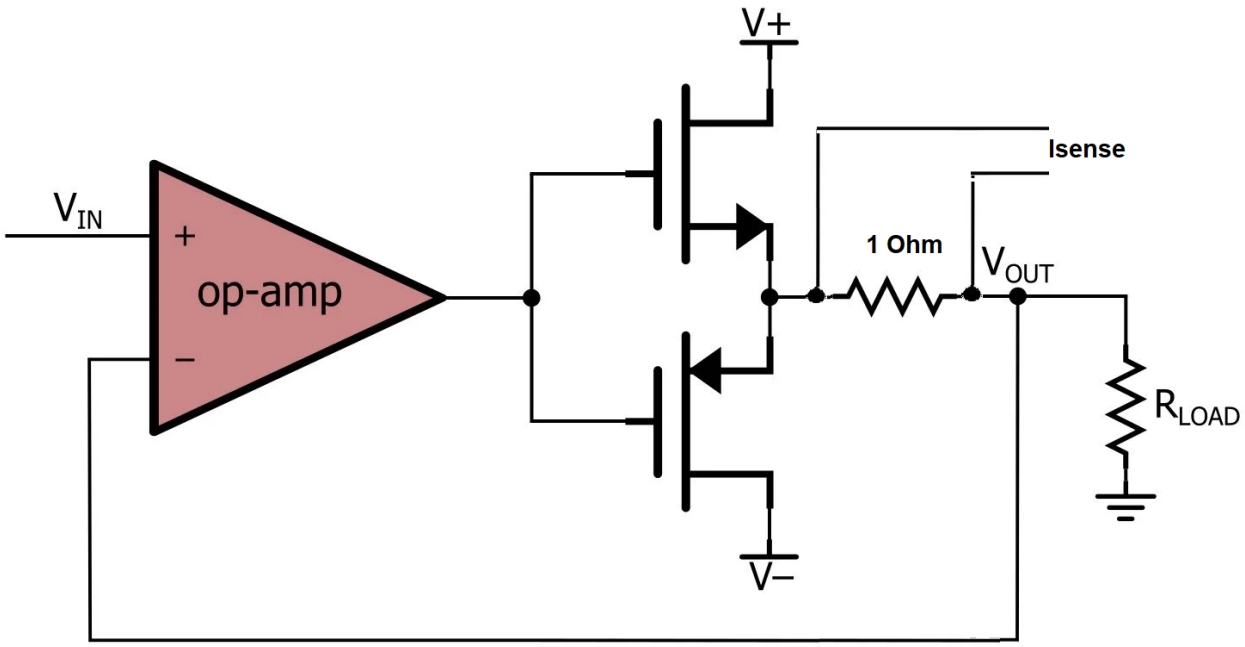


Figure 2 : Output driver and current sensing (Simplified)

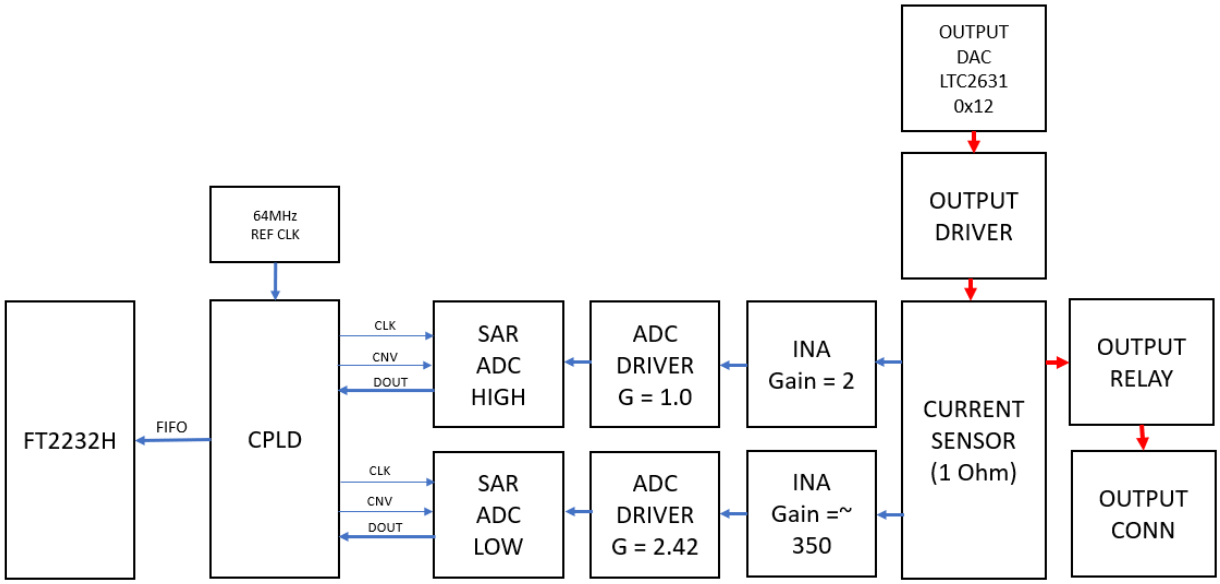


Figure 3 : Front end block diagram

The INA is chosen such that the bias current is negligible compared to the load current. The voltage across the sense resistor is measured by 2 paths. A LOW path and a HIGH path shown by the different gains. At low currents (< 5mA), the current measured by the LOW path is more accurate as it has a higher gain. At higher currents, the LOW path saturates out and the measurements are more accurate in the HIGH path. The measurements from both the paths are sampled simultaneously and used to estimate the actual current. Note that the exact gains and resistor values has to be calculated from the final schematics and BOM. Please refer to the schematics for the exact connection details.

4.2. Digital interface

The CPLD on the board, generates the timing waveform required by the ADC and for the FIFO interface of the FT2232H. A 64MHz clock is used as a reference to generate these waveforms. The CPLD also samples a 6 bit digital port (not shown above) which is sent to the FIFO. The data from the ADC1, ADC2 and the Digital port is sampled simultaneously and sent out in a round robin method on the FIFO bus. The ADC data is sent in 2s complement format with MSB byte sent first. Please refer to the ADCs data sheet for further details.

The CPLD starts to send the data after the RESET line is asserted. The FTDI's BCBUS3 pin is used as a RESET to the CPLD and this signal is active LOW. (The data is sent out only when the BCBUS3 signal is made HIGH). The first data pushed on the FIFO is a dummy data and must be ignored. The subsequent bytes are all required and should be collected in the same sequence without any breaks. The software must ensure that there is no loss of data at any point of time. In case of a break in the data, the sync characters can be used to re-establish the link. See further sections for details on the sync bytes.

The timing of the data is shown below. After the BCBUS3 line goes high, the data is pushed on the FIFO, which is automatically read by the USB driver. The application should query the driver periodically to collect this data in order to prevent any buffer overflows and subsequent data loss.

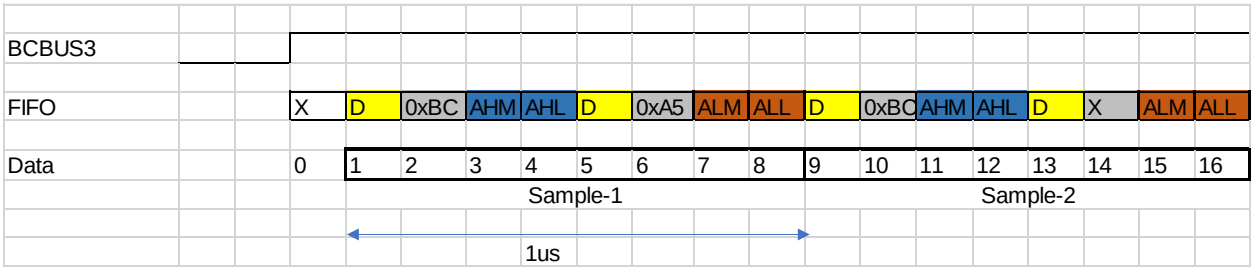


Figure 4 : Data flow on the USB bus

Legend

D => Digital data

0xBC , 0xA5 are sync characters

AHM, AHL are ADC High's MSB and LSB respectively

ALM, ALL are ADC Low's MSB and LSB respectively

The sample flow on the USB FIFO is shown again below in detail

CLK	Data	Sample No
0	DUMMY	
1	Digital	Sample-1
2	SYNC (0xBC)	
3	ADC_HIGH_MSB(AHM)	
4	ADC_HIGH_LSB(AHL)	
5	Digital	
6	SYNC (0xA5)	
7	ADC_LOW_MSB (ALM)	
8	ADC_LOW_LSB (ALL)	
9	Digital	Sample-2
10	SYNC (0xBC)	
11	ADC_HIGH_MSB(AHM)	
12	ADC_HIGH_LSB(AHL)	
13	Digital	
14	SYNC (0xA5)	
15	ADC_LOW_MSB (ALM)	
16	ADC_LOW_LSB (ALL)	
17	Digital	Sample-3
18	SYNC (0xBC)	
19	ADC_HIGH_MSB(AHM)	
20	ADC_HIGH_LSB(AHL)	
21	Digital	
22	SYNC (0xA5)	
23	ADC_LOW_MSB (ALM)	
24	ADC_LOW_LSB (ALL)	

The samples are sent out at the rate of 1MHz. For e.g. Sample-1 contains 8 bytes of data corresponding to 2 bytes of Digital data and 2 bytes of ADC HIGH and 2 bytes of ADC LOW along with 2 sync data. The effective data rate to the PC is 8 x 1Mbyte = 8Mbyte/sec (64 Mbits/sec)

Note that the Digital port gets truncated to only 6 bits as the 2 bits on the MSB are always zero. Please refer to the schematics for further details.

Note : The dummy data is un-known and can be any 8 bit value.

The sync bytes cab be used by the software to establish a synchronization scheme.

4.3. Current estimation

The current measured by the HIGH path can be estimated as per the code given below

```
int Sample = 0;

 int SignedSample = 0;

 Sample = Sample + (int)((ADC_HIGH_MSB << 8) & 0x00FF00);

 Sample = Sample + (int)((ADC_HIGH_LSB << 0) & 0x0000FF);

if (Sample >= Math.Pow(2, DevModel.AdcNoOfBits - 1))

    SignedSample = Sample - Convert.ToInt32(Math.Pow(2, DevModel.AdcNoOfBits));

else

    SignedSample = Sample;

//Calculate voltage in V

Double VoltageSample = (Double)SignedSample / Math.Pow(2, DevModel.AdcNoOfBits - 1) * DevModel.Vref;

VoltageSample = VoltageSample - DevModel.VoltageOffsetMain;

Double CurrentHigh = -1 * VoltageSample / (DevModel.Rs * DevModel.MainCsaGain);

//Note : The Vref = 3.0V, AdcNoOfBits = 16, MainCsaGain = Total Gain of the High Path, Rs = 1 Ohm, VoltageOffsetMain is
the offset in volts which is estimated with no load.
```

Similarly, the current measured by the low path can be estimated.

Having the data from both high and low current measurement, the optimum value of the load current has to be estimated to minimize the error.

4.4. Voltage measurement

The output voltage is measured using a resistor divider network followed by an ADC. The divider ratio for the ZS-1100-A is 1/6. The SD ADC MCP3425A0T is used to estimate this scaled down voltage and the PC software logs this data with time. A sampling time of 20ms to 50ms is adequate for the voltage sampling as the voltage is not expected to drop at a fast rate. An exception would be in the case of high output resistance setting, where the drop can be computed by knowing the output current and resistance.

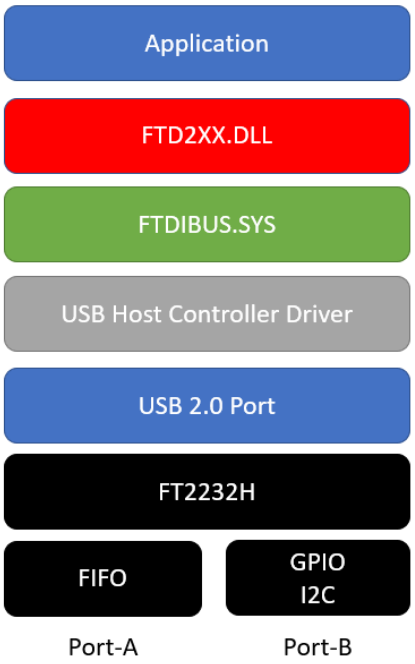
4.5. GPIO Allocation

The Port-B of the FT2232H is used to control various digital input and outputs. The table below summarizes the usage of the GPIO port

Signal	Usage	Direction (w.r.t. PC)
BCBUS0	DC Jack Detection	Input
BCBUS1	Short Circuit Detect	Input
BCBUS2	Over Heat Detect	Input
BCBUS3	Reset	Output
BCBUS4	Warmup Detect	Input
BCBUS5	Relay EN	Output
BCBUS6	Driver EN	Output
BCBUS7	Over Voltage	Input

4.6. Software Overview

The software stack for the application running on the PC is shown below. The IOT Power Profiler GUI provided by ZSCircuits is written in C# using DOT Net Framework 5.0.



The PC software essentially functions can be described as below

1. Set the DACs to set the output voltage and current limiting
2. Turn ON the output relay.
3. Once **START** is clicked, raise the BCBUS3 High
4. Collect the data from the USB end point (Using D2XX dll) continuously without break
5. Process the data in real time to extract the current and digital data
6. Compress the current signal and digital data signal to save space
7. Write the current and digital data to the disk in separate files
8. Capture the voltage waveform at a periodic interval (20ms to 50ms) using the I2C bus (2^{nd} port of the FT2232H)
9. Monitor the state of the different status lines from the CPLD (Overheat, Over current etc) and provide indications on the GUI. Take appropriate actions based on the status.
10. Lower the BCBUS3 when the data collection is stopped.

4.7. Warning

The signals on the CPLD are hard coded to be inputs or outputs. These signals are connected to the FT2232H directly. Incorrect programming of the FT2232H could potentially damage the IO ports of the CPLD and/or the FT2232H, in case of a clash (e.g. output shorted to output with different logic levels). Hence care should be taken to ensure that the IOs are programmed correctly by the software. Since the port type may be set by the contents of the EEPROM, ensure that the EEPROM for the FTDI matches with the EEPROM template provided by ZSCircuits.