ICARUS

Power Monitor

Software Reference Manual

This document describes the functionalities of the board, based on Aria G25 SMD Module. On the board is installed an embedded Linux system, generated with Buildroot tool. It needs about 16 seconds to boot.

To prevent micro SD corruption, the files system is mounted in read only mode.

To communicate with the board the power monitor server use the WebSocket protocol (RFC 6455) on port 4444. A WebSocket library for LabView is provided with a simple example to communicate with the board over Ethernet, at link: http://www.pd.infn.it/~caste/pub/WebSockets.zip. On the board is running a Web server (lighttpd) with a simply web page to control and monitor the power supply, also a Secure Shell server (sshd) is running on the system.

Getting Started

Ethernet Configuration:

By default, the Ethernet is configured with DHCP enabled, configuration can be changed by touchscreen display or using console debug connector (mini USB connector, USB to RS232. FTDI FT230X) and edit file /etc/network/interfaces. The files system is in read only mode, use command /usr/sbin/rw to switch from read only to read write mode and /usr/sbin/ro to switch back.

WARNING the Ethernet MAC address is generated with microSD serial number.

USB device connector:

The board can be used directly connected to PC via USB device connector (USB typeB), it provides a USB Ethernet Gadget configured with static IP 10.42.0.10, configure the PC side with the same subnet address, for example 10.42.0.1. . Configuration can be changed by touchscreen display or can be changed by edit file /etc/network/interfaces, logging using console debug connector. Warning the Ethernet Gadget at boot generate random MAC address, so on the linux PC (for example Ubuntu with Network Manager) the network must be reconfigured or you must create a special UDEV rule to automatically configure it.

Below a simple script to create udev rule for usb0 on linux PC, run the scrip only one time and with root privilege.

```
#!/bin/sh
MAC=02:11:22:33:44:55
echo "[keyfile]" >>/etc/NetworkManager/NetworkManager.conf
echo "unmanaged-devices=mac:$MAC" >>/etc/NetworkManager/NetworkManager.conf
echo "#!/bin/sh" >/etc/udev/usb0.sh
echo "/sbin/ifconfig usb0 down" >>/etc/udev/usb0.sh
echo "/sbin/ifconfig usb0 hw ether $MAC" >>/etc/udev/usb0.sh
```

```
echo "/sbin/ifconfig usb0 10.42.0.1" >>/etc/udev/usb0.sh
echo "/sbin/ifconfig usb0 up" >>/etc/udev/usb0.sh
echo 'SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*",
ATTRS{product}=="RNDIS/Ethernet Gadget",  KERNEL=="usb*", RUN+="/bin/sh
/etc/udev/usb0.sh"' >>/etc/udev/rules.d/70-persistent-net.rules
```

The commands accepted by server are:

Status command: return the status.

Syntax: **Status?**

Returns: the status in JSON format, example:

```
{ "arr": [{ "volt": 11.548000, "curr": 0.012023, "temp": 27.937000 }, {
"volt": 9.456476, "curr": 0.014617, "temp": 27.750000 }, { "volt": 5.028000,
"curr": 0.012695, "temp": 27.750000 }, { "volt": 3.495564, "curr": 0.023248,
"temp": 27.062000 }], "local": 0, "power": 1, "fan": 0}
```

Fan value: 0=high speed, 1=middle speed, 2=low speed. Value -1.00 on voltage and current or value -300.00 on temperature indicate a read error on I2C bus

Version command: return the software version.

Syntax: **Version?**

Returns: **V2.0**

Password command: enable the connection to accept SET commands. The password is not encrypted, and can be set on file /etc/IcaPwrMon.conf.

Syntax: **Password:<password>**

Returns: **OK**

Set Power command: set power supply on/off

Syntax: **Set:Power <0/1>**

Returns: **OK**

Set Fan command: set fan velocity

Syntax: **Set:Fan <0/1/2>**

Argument: Fan value: 0=high speed, 1=middle speed, 2=low speed.

Returns: **OK**

Records command: return the history of the monitoring data. The board store data every second for a total of 1024 points.

Syntax: **Records:Range? <time>,<maxsize>**

Arguments: <time> specified the time in seconds. First time use time=0 to obtain the first data available, then use the last time+0.1, received from a previous call, to obtain successive data.
<maxsize> is the maximum number of record to return and must be less or equals than 64

Returns: return the stored data in JSON format, example:

```
{"T0" : [15.5,24.31],[16.5,24.31],[17.5,24.31],[18.5,24.31],[19.5,24.31]],
"T1" : [[15.5,23.94],[16.5,23.94],[17.5,23.94],[18.5,23.94],[19.5,23.94]],
"T2" : [[15.5,23.94],[16.5,23.94],[17.5,24.00],[18.5,24.00],[19.5,24.00]],
"T3" : [[15.5,24.12],[16.5,24.12],[17.5,24.19],[18.5,24.19],[19.5,24.19]],
"V0" : [[15.5,12.00],[16.5,12.00],[17.5,12.00],[18.5,12.00],[19.5,12.00]],
"V1" : [[15.5,9.00],[16.5,9.00],[17.5,9.00],[18.5,9.00],[19.5,9.00]],
"V2" : [[15.5,-5.00],[16.5,-5.00],[17.5,-5.00],[18.5,-5.00],[19.5,-5.00]],
"V3" : [[15.5,3.30],[16.5,3.30],[17.5,3.30],[18.5,3.30],[19.5,3.30]],
"I0" : [[15.5,1.00],[16.5,1.00],[17.5,1.00],[18.5,1.00],[19.5,1.00]],
"I1" : [[15.5,1.00],[16.5,1.00],[17.5,1.00],[18.5,1.00],[19.5,1.00]],
"I2" : [[15.5,1.00],[16.5,1.00],[17.5,1.00],[18.5,1.00],[19.5,1.00]],
"I3" : [[15.5,1.00],[16.5,1.00],[17.5,1.00],[18.5,1.00],[19.5,1.00]],
"maxrecord": 1024}
```

Value -1.00 on voltage and current or value -300.00 on temperature indicate a read error on I2C bus.

Returns Error message Syntax: **ERROR:<number>,<message>**

Configuration file /etc/IcaPwrMon.conf:

```
#
# Temperature sensors assignment
# syntax:
# alias <temperature ID> <Power Supply ID>
#

alias 10-0008006e80d5   1
alias 10-00080048a21e   3
alias 10-0008006ea336   0
alias 10-0008006ebc6e   2

#
# Temperature unit
# C = Celsius
# F = Fahrenheit
T-Unit C

#
# Power On/Off password
#
password icarus

#
# Current Calibration
# syntax:
# Imon<Power Supply ID> <m> <q>
#

Imon0 0.99586 -0.02582
Imon1 0.988377 -0.010092
Imon2 0.98981 -0.01502
Imon3 0.93496 0.002679

#
# Voltage Calibration
# syntax:
# Vmon<Power Supply ID> <m> <q>
#
# V=Vread - (I*m+q)
# (I*m+q) is shunt partition (about shunt/3)
#

Vmon1 0.008839 0.007395
Vmon3 0.01766 0.004025
```