PiPower power management system

Introduction

Overview

The PiPower board is a power management system intended to allow a Raspberry Pi (or other electronic device) to be powered at a remote location. Its design allows for the direct connection to a Raspberry Pi via the GPIO header, but can be used with any device requiring a 5v supply. When used with a Raspberry Pi, simple two way communication can be established between the PiPower board, and the Raspberry Pi. This allows for the PiPower to send battery status information to the Raspberry Pi, and for the Raspberry Pi to signal to the board that it needs to be hard-reset.

The PiPower is controlled by an always-on microcontroller. The microcontroller chosen is a PICAXE 08M allowing for easy end-user modification of its firmware.

Why the need for the PiPower?

Inexpensive consumer electronic devices generally make no guarantees as to continuous operational reliability (or “uptime”). For example, a USB 3G modem might work fine for a few hours or days, but require a full reset (i.e. be removed and re-inserted) from time to time. Doing a full reset of a device is no problem when it's sitting in front of you but becomes an issue when access is difficult - whether it be in your roof, or 1000km away. The goal of the PiPower is to allow the use of cheap, consumer grade electronic devices in off-grid projects where regular human intervention is difficult.

Features of the PiPower

1. 5v, 3A power supply
2. Solar regulator for charging lead-acid batteries. (up to 50W solar array)
3. Battery state monitoring (i.e. signaling when batteries are depleted to avoid deep discharge)
4. Switching off and hard power-resetting of connected devices.

Design philosophy

1. All-or-nothing power control. For the sake of simplicity, it was decided that the PiPower will remove power from all attached devices instead of, for example, just USB ports.
2. Tight physical integration with Raspberry Pi. The Raspberry Pi is a powerful Linux platform with very low power requirements (particularly the model A version). It is ideal for use in small solar installations. Despite the design intent for use with a Raspberry Pi, the PiPower can be used with any device – it just requires four hook-up wires for power and signaling.
3. Easily modified by the end-user. The PICAXE is easy to program in BASIC, and can be modified while in-circuit. Source code for the Raspberry Pi heartbeat program is supplied.

Operation

Powering the Raspberry Pi

Power supply issues are common with Raspberry Pi experimenters. Generally this is due to poor quality phone charger type supplies and/or associated cables. For this reason, the PiPower powers the Raspberry Pi directly via the GPIO header with a quality 3A switched-mode power supply.
The choice of a switched-mode supply ensures the highest efficiency possible which is important for battery powered systems. Coupled with a Raspberry Pi model A, a user can expect a system consuming around 2W of power (depending on attached USB peripherals, etc).

Solar battery charging

The PiPower is designed to be used with small, stand-alone 12v solar systems. As such, it incorporates a simple solar regulator. The solar regulator is controlled by the PICAXE microcontroller, and its operation can be easily modified by the user. The use of the solar regulator system is optional.
The solar charging system employs a shunt regulator design. For this reason it is important to install an in-line fuse to protect the MOSFET in the event that a high-current power source is mistakenly applied to the solar input. Solar panels ranging from 5W to 50W are suitable. The MOSFET turns on (i.e. shunts the solar input to ground) when the battery voltage reaches around 13.8-14v.
Note: The shunt turn-on voltage of 13.8v should ensure that a 12v lead-acid battery, if accidentally connected to the PV input, shouldn’t cause the MOSFET to switch on as the no-load voltage of a lead-acid battery is typically below this (when not connected to a charger). This provides a level of protection against damage if a battery is mistakenly connected to the solar input, however a fuse is still recommended.

Battery status monitoring

Deep discharging of certain battery chemistries can shorten their life, or even permanently damage them. For this reason, the PiPower incorporates a battery monitoring system in order for the system to take some action (such as removing power to connected devices) when batteries are getting depleted.

Power supply control

An important feature of the PiPower is its ability to completely remove power from a device connected to its 5v supply, such as the Raspberry Pi itself. This may be required to protect depleted batteries, or to do a “hard” reset of a device that's malfunctioning.

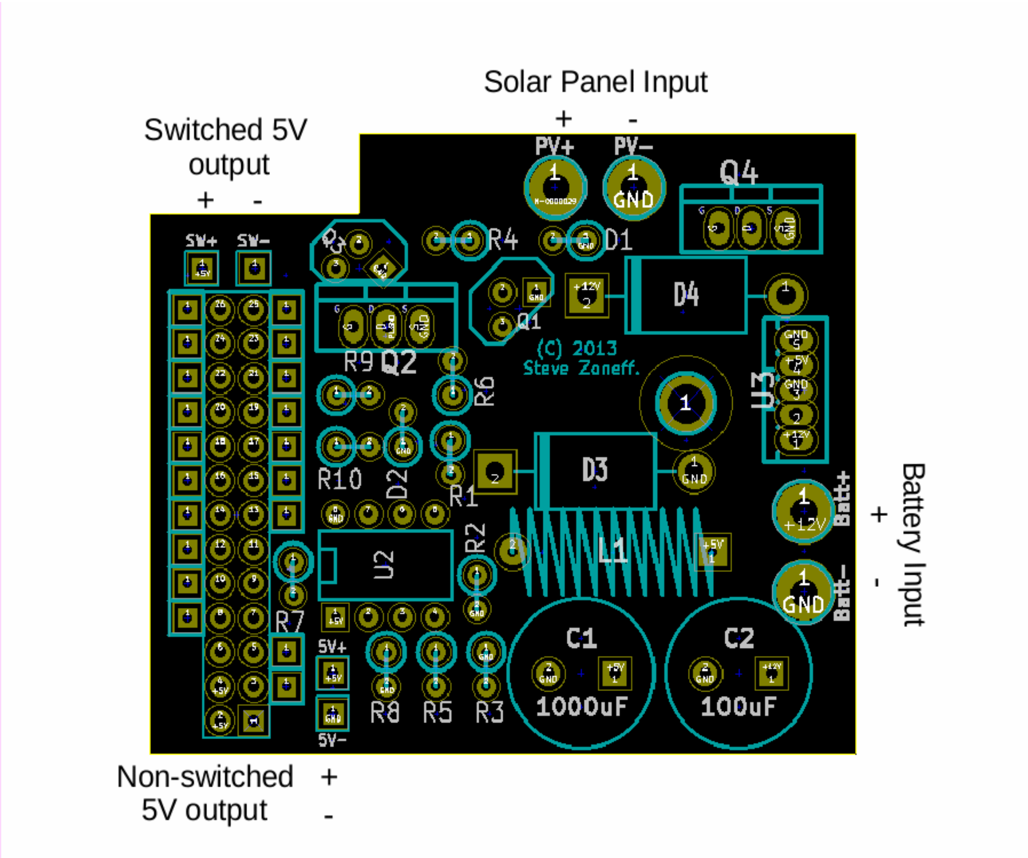
Raspberry Pi to PICAXE signaling (GPIO 11 to pin c.3)

By default, this connection is used to provide a “heart-beat” from the Raspberry Pi to the PICAXE. This allows the PICAXE to determine the state of the Raspberry Pi. While the heartbeat is present, the PICAXE will not remove power from the connected devices.

PICAXE to Raspberry Pi signaling (pin c.2 to GPIO 07)

In its default configuration, this pin carries battery voltage information to the Raspberry Pi. With the default firmware installed into the PICAXE, the information transferred is the 8-bit number representing the voltage on the ADC pin (PICAXE Pin 3). This is stored on the Raspberry Pi in /tmp/voltage. If this number is low (i.e. indicating a low battery voltage), the Raspberry Pi can shut itself down, hence stopping the heartbeat which causes power to be removed. The PICAXE will keep the power removed until the battery voltage has risen to a value set in its firmware (default ~12.8v).

Connection descriptions



Switched 5V output:	Power output for a device to be switched on/off.
Solar Panel Input:	Power from a solar panel (up to 100w in size).
Battery Input:	Power input to the system.
Non-switched 5V output:	Permanently powered 5V supply.

PiPower V1.0 (through-hole version) assembly notes

Generally it's a good idea to install passive components first and semiconductors last. In this case however, it is recommended that the entire power supply section be installed and tested before installing the rest of the board. To do this, install components C1, C2, L1, D3 and U3 first. These components constitute the power supply.
With these components installed, apply power (around 12v) to the Batt+ and Batt- connection. Test that 5v is measured across the points marked 5v+ and 5v- (located between R8 and the 26 pin header).
If the power supply section checks OK, proceed with the rest of the assembly.

The board is connected to the Raspberry Pi with the component side of the board facing up. Therefore, the 26 pin header is installed so that it sits on the underside of the board, with its soldering being done on the component side of the board.

Note that if the solar regulator feature is not required, components R9, R10, Q3, Q4 and D4 can be omitted from the assembly.

Specific component notes

Resistors are not polarity sensitive but should be installed sitting upright with the body of the component sitting on the board where the "ring" is drawn for each resistor. This reduces the risk of short-circuits between components, and allows for the attachment of the PICAXE programming cable to the correct legs of R3 and R4.

Note the polarity of D3, D4 (Cathode ends toward the 26 pin header). These diodes are laid flat along the board.

Capacitors C1 and C2 are polarity sensitive and require that the negative leg is inserted into the hole with the round solder pad (closest to the 26 pin header). The positive leg goes through the hole with the square solder pad.

U2 (PICAXE 08M2) is inserted with the "notched" end facing toward the 26-pin header. Pin 1 of this component is near the R7 and 5v+ board labels.

U3 (SMPS controller) is inserted with the metal tab side facing inward (i.e. toward the 26 pin header), while Q2 and Q4 are inserted with the metal tab side facing outward. The outline on the PCB indicates the side the tab must face for correct orientation.

Q1 and Q3 are inserted as per the PCB outline. Note Q1 sits at an angle when compared with Q3.

Programming the PICAXE 08M2 (loading firmware)

The PICAXE microcontroller can be programmed in-circuit by making connections to the following components:

TX from the PC: Leg of R3 (equates to the junction of the 10k and 22k programming resistors)
RX to the PC: Leg of R4 (equates to pin 7 on the PICAXE)
Ground: Anode of D3 (i.e. the leg facing U3)

Refer to the standard Revolution Education instructions for further details on programming PICAXE devices.

Appendix A: Bill of materials

Reference	Component	Notes
R1, R8	3.0k Ohms	R1 should be 1% tolerance.
R2, R4, R7, R9	1.5k Ohms	R2 should be 1% tolerance.
R3, R6, R10	10k Ohms	R3 is part of the PICAXE programming cct.
R5	22k	R5 is part of the PICAXE programming cct.
C1	1000uF electrolytic	Low ESR.
C2	100uF electrolytic	Low ESR. Note DC working voltage requirements (25v+ for typical 12v solar systems).
D1, D2	~14v Zener	Optional. Use if MOSFET Vgs is likely to be too high for the device. Not required for typical 12v solar systems.
D3, D4	1N5822 Diode	
L1	100uH inductor	
Q1, Q3	BC547 NPN transistor	
Q2, Q4	IRF3205 N-Channel MOSFET	
U2	PICAXE 08M	Programmed with appropriate firmware.
U3	LM2576T-5	5v SMPS regulator.

Appendix B: Modifying the code.

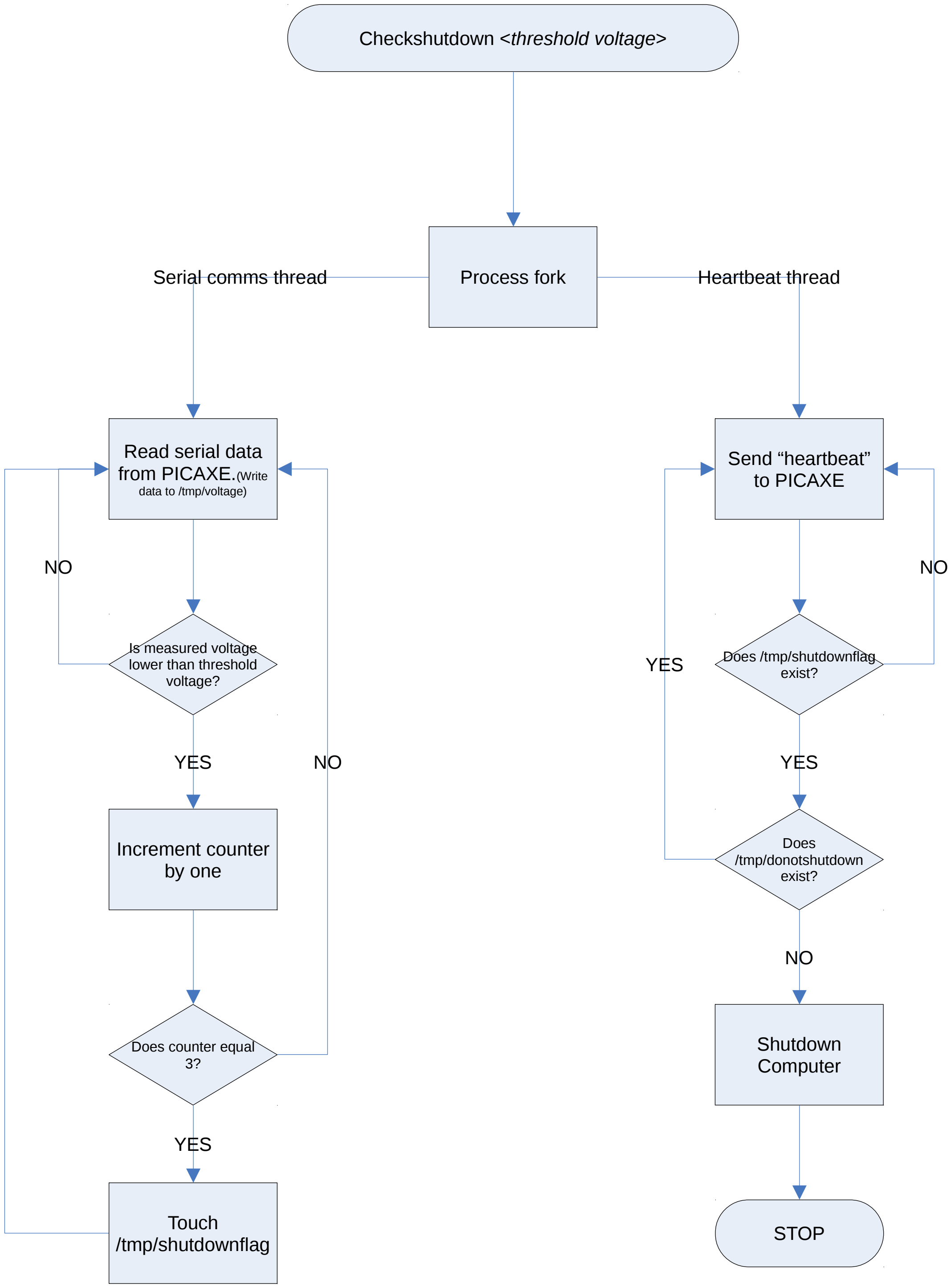
PICAXE code

The board incorporates a PICAXE08M that can be re-programmed using the standard Revolution Education tools using their BASIC language or flow chart method.
Reasons for changing the PICAXE code may include changing the voltage thresholds or removing the heartbeat requirement.

Raspberry Pi code

A basic heartbeat program (checkshutdown) is supplied that monitors the battery status signal on GPIO7, and generates a heartbeat signal on GPIO11. This program looks for the presence of /tmp/shutdownflag, and if present, shuts down the Raspberry Pi. This obviously removes the heartbeat, and if the supplied PICAXE code is in use, will cause a hard power reset of the attached devices.
The /tmp/shutdownflag file can be created for any reason. For example, it could be created by a program monitoring the status of attached USB devices, and if a problem is found, creating this file will cause a hard-reset of all devices.
The checkshutdown process receives voltage data from the PICAXE and stores the raw value (a number between 0 and 255) to /tmp/voltage.
The checkshutdown process can be started automatically by entering its path into /etc/rc.local, or creating an entry into /etc/inittab.
When modifying the code, ensure that any trigger you cause for a hard-reset to occur is accompanied by a graceful shutdown of the Raspberry Pi (unless read-only file-systems are in use).
Compiling this program requires that the BCM2835 libraries are installed.

Appendix C: Checkshutdown process flowchart



Appendix D: Schematic diagram