Omni-EP

Omni-EP User's Manual Version 1.0

Document Part No. 94-0034
Published September 2005

Limited Warranty

JK microsystems, Inc. warrants each Omni-EP to be free from defects in material and workmanship for a period of 90 days from the date of purchase. This warranty shall not apply to any unit which has been subject to misuse, neglect, accident, or abnormal conditions of operation.

JK microsystems' obligation under this warranty is limited to repairing or replacing, at JK microsystems' option, any unit returned to the factory within 90 days of the date of purchase, provided that JK microsystems determines that the unit is defective and has been used in compliance with the terms of this warranty. If the failure has been caused by misuse, neglect, accident, or abnormal conditions of operation, repairs will be billed at a nominal cost.

The foregoing warranty is exclusive and in lieu of all other warranties, expressed or implied, including, but not limited to, any warranty of merchantability or fitness for any particular purpose. JK microsystems shall not be liable for any special, incidental or consequential damages, whether in contract, tort, or otherwise.

Important Notice

Life Support / Mission Critical Applications

This product is not fault-tolerant and is not designed, manufactured or intended for use or resale as on-line control equipment in hazardous environments requiring fail-safe performance, such as in the operation of nuclear facilities, aircraft navigation or communication systems, air traffic control, direct life support machines, or weapons systems, in which the failure of our hardware or software could lead directly to death, personal injury, or severe physical or environmental damage.

Table of Contents

Overview

The Omni-EP single board computer is based on the Cirrus EP9302 microprocessor. The EP9302 is a high performance, 32-bit, single-chip microcomputer with a Memory Management Unit. Onboard Ethernet provides a direct connection to 10/100 Mbps networks. The Linux operating system allows for maximum versatility and control. High endurance flash memory eliminates EPROM programming without worry of damaging the onboard non-volatile memory with repeated program cycles. Applications are uploaded directly into the flash disk. Expansion options provide high capacity flash storage eliminating the size and reliability problems associated with electro-mechanical storage devices.

Software development for the Omni-EP is remarkably simple and quick. Programs are written on a PC compatible computer in the language of your choice. After your application has been compiled or assembled and linked, it is uploaded to the Omni-EP's flash disk with your favorite telecommunications program using the X-Modem protocol. The application can then be tested and debugged through the console serial port. When the application is running to your satisfaction, the startup script file can be modified so that the application will load and execute upon reset or powerup. These features yield a quick and cost effective solution for applications such as networking, embedded web and serial protocol conversion.

Features

200 MHZ EP9302 ARM920T Processor
100 MHZ System Bus
10/100 Mbit/sec Ethernet controller with power saving features.
32 MBytes Ram Memory
16 MBytes Flash Memory capable of hardware write protection
High Speed Serial Ports (16550 Type):
    1 RS232 Port (TxD, RxD, GND)
    1 RS-232 with modem control (RTS, CTS)
16 Digital I/O lines (16 individually configurable for interrupts)
Internal SPI Bus for easy peripheral expansion
AC97 Audio with Line In/Out
    Headphone amplifier on output
2 Port USB 2.0 Full Speed Host (OHCI)
    1 Port external to enclosure
Battery backed clock calendar
Watchdog Timer
2.6 Linux Kernel
Compact size

Operation

When 5V DC is applied to the power terminal, the RedBoot bootloader will begin execution using Serial 1 (J7) for output. The Serial port is set by default to 57600 baud, 8 data bits, 1 stop bit, and no parity. By default it will wait 1 second before running a default script to load the default linux kernel and a ramdisk as the root file system. If a cntrl-c is received in the 1 second interval RedBoot can allow you to change the default script, IP addresses, as well as the kernel and filesystem images.

If the cntrl-c signal is not received RedBoot will execute the default script and begin execution of the kernel code. After the kernel loads, the default startup has been setup to mount the flash memory (on /mnt/FlashMemory) and then execute the startup script /mnt/FlashMemory/startup. The startup script can be modified by the user so that user programs can be run on startup.

> **Warning**
> Although the flash memory devices used have a guaranteed lifetime of over 10,000 write cycles, it is possible for an application to quickly wear them out. The on board flash memory is intended to store programs, and setup data which is normally not changed. Avoid storing data or frequently changed information on the flash disk.
>
> **Warning**
> In order to prevent issues of data loss on flash memory, it is recommended that after every write that the file system is unmounted. Loss of power before the file system is unmounted may result in loss of data.

Getting Started

In order to start development on the Omni-EP you will need a PC compatible computer with an available serial port. Connect the Omni-EP's console header (J7) to the available serial port on the PC with a DB9 pin serial cable. Run the telecommunications program configuring it for 57600 baud, 8 data bits, 1 stop bit, and no parity. Apply power to the Omni-EP using a source of regulated +5V DC power capable of supplying 1 amp.

The OmniFlash-EP will take several seconds to boot, and will display its progress while doing so. It should respond with a welcome message, and ask you to press enter for a shell prompt. Press enter and then type ls, then enter again to see the contents of the root directory.

> **Warning**
> The bootup process will be halted or be incomplete if characters are received on the UART line while booting. This means that in many cases the Omni-EP will need to be brought up before devices send it information over this connection.

If you do not get a welcome message, or an echo of the characters you type, you need to check your serial port setup. To check everything but the Omni-EP, remove the ribbon cable from J7

and short pins 3 and 5 on the cable. If characters sent to the Omni-EP are not echoed, you need to check your serial setup. You must resolve the issue before you can continue.

If you were able to ls successfully, take a few moments to explore the contents of the file system. The default root file system is resident in RAM. This means that any changes to the core files of the system will not remain after a reset. This is done to ensure that files important to the operating system cannot be lost.

The flash memory is mounted on system initialization to the `/mnt/FlashMemory` directory. Files modified here will remain after power loss or system reset, however it is recommended that the file system be unmounted after a write, as with any linux system.

After looking at the files on the Omni-EP, the next step is to try and upload a file. One of the ways of doing this is serially using the `rx` command. On the Omni-EP, type `rx` followed by the name of the file you wish to send. The Omni-EP will begin sending characters to the console, polling for the file.

On your PC, start the file transfer, usually by pressing cntrl-a, and then s. Select the xmodem protocol, and enter the filename. The transfer should start, and when complete you should be returned to the shell prompt. If it does not work, there are a few things to check. First verify that the terminal program does not have handshaking or flow control enabled. Second the Carrier Detect signal (pin 9 on the DB-9 connector) can be sensed as low or false. Make sure that the signal is at least +3 volts into your PC if you are not able to transfer files.

If the transfer terminated without problems, you have a working development environment. There are other methods available for transferring files including tftp and a USB storage disk.

When power is applied to the Omni-EP, one of the first things it does is look for a CNTRL-C character received from the keyboard at 57600 baud. If it receives this signal within one second of startup, boot will halt and you will get a RedBoot prompt. This allows for convenience during development, and a fail-safe break method.

The most reliable method to break the boot process using the CNTRL-C is to connect the Omni-EP to your computer with the ribbon cable and start your communications program. Press and hold CNTRL-C as you apply power to the board. Using this approach, a stream of characters will be sent to the board as it powers up. If the CNTRL-C flag is not found, the board will continue to load the kernel and mount the file systems. It will by default try to execute the file `/mnt/FlashMemory/startup`.

Hardware

Digital I/O Ports

The Omni-EP has a total of 16 digital I/O lines directly from the processor organized as follows:

Port A I/O

Port A has 8 bits of I/O available.

These 8 bits are available for users. By default, they are set as input and the status of the lines can be read on the data register (0x80840000). To set these bits as output, write a 1 in the corresponding bit position of the direction register (0x80840010). Values can then be output on the lines by writing to the data register (0x80840000).

To enable interrupts on these lines, first ensure that the direction of the port A bits desired is "input". Next, set the bit desired in the Interrupt enable register, (0x8084009C).

The type of interrupt can be decided by writing a 1 for edge sensitive or 0 for level sensitive (default) to the Interrupt Type 1 register (0x80840090). A "1" written to the corresponding bit of the Interrupt Type 2 Register (0x80840094) sets the pin to a high level or rising edge trigger. A "0" will set the level as low, or falling edge trigger (default).

Port B I/O

Port B has 8 bits of I/O available.

All 8 bits are available for users. By default, they are set as input and the status of the lines can be read on the data register (0x80840004). To set these bits as output, write a 1 in the corresponding bit position of the direction register (0x80840014). Values can then be output on the lines by writing to the data register (0x80840004).

To enable interrupts on these lines, first ensure that the direction of the port B bits desired is "input". Next, set the bit desired in the Interrupt enable register, (0x808400B8).

The type of interrupt can be decided by writing a 1 for edge sensitive or 0 for level sensitive (default) to the Interrupt Type 1 register (0x808400AC). A "1" written to the corresponding bit of the Interrupt Type 2 Register (0x808400B0) sets the pin to a high level or rising edge trigger. A "0" will set the level as low, or falling edge trigger (default).

LEDS

Additionally, the onboard LEDs can be controlled. The orange LED is set as bit 1 of 0x80840020, and the green as bit 0 of 0x80840020. Writing a 1 to the position turns the LED on, clearing it turns it off.

Programming the I/O Ports

While it may seem tempting to try and access these registers with a pointer to memory, this is not feasible because Linux uses a memory manager for accessing parts of memory. The locations as seen by a pointer are in fact to virtual memory, not physical memory.

For convenience, the program `regedit` has been included with the board. It allows for a user to set and clear bits in a register on the command line, or in a script. To read a value, place the desired register after the call to the program. If a write is desired, a value can be placed afterwards to be written. To set it into quiet mode, place a `-q` before the arguments, and the program will only output a minimal amount of information.

A more in depth approach to modifying the registers can be used by first opening `/dev/mem`, and then using `mmap` to have linux map the values back and forth for you. For a detailed example of this, please refer to the `regedit` source code, as well as the man pages for `mmap`.

Interrupts are only available in kernel mode, as in all Linux machines. This means that if I/O interrupts are needed, then a kernel module will have to service these interrupts. There are some options on the chip for debouncing, and it is recommended to use them or other debouncing techniques for the I/O interrupts. An example of this is included on the Dev CD. Note also that the interrupt for ports A and B are combined into a single interrupt to the OS.

Driving Loads with the I/O Ports

The ports on the Omni-EP are capable of driving small loads or interfacing to TTL logic devices. These ports can only source/sink a few milliamps. In order to interface with many loads, additional circuitry, such as a transistor or relay, will be required. Designing the interface between an output pin and a higher current load can present a challenge, especially if high speed is required or the load is inductive in nature.

Switching inductive loads such as relays, solenoids and motors can generate transient voltages many times larger than the steady-state operating voltage of the load. For example, turning off a 12 volt solenoid can easily create a negative spike of 200 volts. Worst case, these transients can destroy your controller. In milder cases, they can cause program failures and flash memory corruption. In the case of high current, high inductance devices, the spike need not even be directly connected to the controller to cause damage or program failure.

Controllers damaged by inductive spikes are considered to be abused and are not eligible for warranty repair.

A detailed study of dealing with inductive spikes is beyond the scope of this manual. For more information, a good starting point is The Art Of Electronics, 2nd Ed. (Horwitz and Hill, 1989) pages 52-53.

The following items should be considered when driving inductive loads:

A) When driving a DC inductive load, place a diode in parallel with the load. In most cases, the diode can be a general purpose power diode such as a 1N4002. The cathode (banded end) of the diode should connect to the positive side of the load. Locate the diode as physically close to the load as possible. This applies to a small relay driven by a port pin, as well as a larger inductive load connected to the contacts of a relay.

B) If you are using a relay to switch an AC-powered inductive load, put a varistor in parallel with the load. The varistor voltage rating should be about 2 times the RMS (1.5 times the peak-to-peak) steady-state voltage of the load.

C) Relays switching an inductive load may require a capacitor placed across their contacts. 0.1µF to 1.0µF is a good starting point. If the relays are switching an AC load, place a 100 ohm resistor in series with the capacitor.

D) Do not use the controller's ground or power conductors to carry current from switched inductive loads. Isolate these signals and route then directly to and from the power supply and as far away from the controller as possible. A separate power source for large inductive loads is strongly recommended. In the case of very large inductive loads, a separate enclosure for the controller may be required.

Asynchronous Serial Ports

The Omni-EP has 2 serial ports, Serial 0 and Serial 1. Both are 16550 compatible and are internal to the EP9302 processor. The maximum data rate is 115k Baud at RS-232 levels.

Serial 0 is wired as Data Communications Equipment (DCE) for direct connection to a computer or terminal. By default this port is configured as a 3 wire RS-232 port implementing Rx, and Tx, CTS, and RTS on the RJ-11 connector.

Serial 1 is wired as Data Communications equipment (DCE) for direct connection to a computer or terminal. It implements Tx and Rx signals on the DB9 connector

The serial ports can be accessed directly through linux by using the cat command and redirecting output to /dev/ttyAM0 or /dev/ttyAM1. Additionally these can be opened for reading or writing by your programs in order to access the ports. To change baud rates, use the `stty` program (see `/etc/rc.sysinit` for example).

Watchdog Timer

The EP9302 is equipped with a watchdog timer that can be configured to generate a processor reset. When enabled, software must keep the watchdog from timing out, indicating proper operation. If the watchdog timer expires, the Omni-EP will be reset.

To activate the timer, write the value 0xAAAA to the watchdog register at 0x80940000. The timer has a resolution of 250ms. If the register is not rewritten with the value 0x5555 within the time period, the board will reset. To turn off the watchdog at any time, write the value of 0xAA55 to the register. This can be demonstrated with the `regedit` program:

```
./regedit  0x80940000 0xAAAA
```

Ethernet

The Ethernet driver for connecting to 10/100 networks is included in the kernel. The only thing the user must do is to initialize the connection. The following two options are available:

Dynamic IP: Use the `udhcpc` program to automatically find an IP address. Your network must be configured for dhcp.

Example: `udhcpc -i eth0`

This will bring up the on board Ethernet connection. It will display the IP address it obtains, as well as the lease time. For more information on the functionality of the program, please refer to the busybox man pages at www.busybox.net .

Static IP:

To have a fixed IP address, use the ifconfig program to enable your Ethernet interface
Example: `ifconfig eth0 10.10.1.197 netmask 255.255.255.0`

Other options may also need to be set. In order to access outside networks, the gateway must be routed.
Example: `route add default gw 10.10.1.1 eth0`

Additionally a nameserver may be needed. To set this option, create a file named `resolv.conf` in the `/etc` directory with the following line:

```
nameserver 10.10.1.100
```

Where 10.10.1.100 should be replaced with your network's nameserver IP address.

Once you have configured your Ethernet connection with one of the above, you can test it with the ping program to ensure that it works. You should be able to ping a computer on the network off the OmniFlash-EP, as well as ping back to the OmniFlash-EP.

The board has two LEDs that indicated the status of the Ethernet link. The LNK LED indicates the status of the Ethernet. When illuminated, the OmniFlash-EP is receiving the Ethernet 'heartbeat' and is connected to a live network. If this LED is not illuminated, there is a problem with the Ethernet wiring or the network. The ACT LED indicates activity on the network. The LED will flash when a data packet is received or transmitted.

USB

Two USB ports are available on the Omni-EP. Many USB device drivers are included with the Omni-EP, including wireless USB capability. Specific drivers may be needed for many devices, but the most common applications do not require any drivers. When compiling drivers for the Omni-EP, it is best to do so as a mod that can be loaded into the kernel at runtime with the `modprobe` command. For more information on doing this, please stop by the forums on www.jkmicro.com

To access a usb pen disk, first insert it into one of the open ports. The kernel should report fiding the device and giving it an address, for example:

~ # hub.c: new USB device not_pci-3, assigned address 2
scsi0 : SCSI emulation for USB Mass Storage devices
Vendor: LEXAR Model: JUMPDRIVE SPORT Rev: 1000
Type: Direct-Access ANSI SCSI revision: 02
Attached scsi removable disk sda at scsi0, channel 0, id 0, lun 0
SCSI device sda: 506880 512-byte hdwr sectors (260 MB)
sda: Write Protect is off
Partition check:
sda: sda1

The important thing to note is the last line where the device is assigned. In this case, the device is sda1. The first pen drive detected will always be assigned sda1, with the next being assigned sda2, sda3, etc. In order for the drive to be accessed, it must be mounted with the following command:

```
mount /dev/sda1 /media/
```

The first argument lists the device, the second the mount location. Mount will automatically cycle through known file system types until it finds one of them on the drive. It may print out filesystem types that it does not find, but will still mount the drive. You can explicitly define the filesystem type when calling the command. For more information, see the busybox man pages at www.busybox.net .

To check and make sure the device is mounted, simply use the mount command again:

~ # mount

/dev/ramdisk on / type ext2 (rw)
/proc on /proc type proc (rw)
devpts on /dev/pts type devpts (rw)
/dev/mtdblock4 on /mnt/FlashMemory type ext2 (rw)
/dev/sda1 on /mnt/USB type vfat (rw)

The last line indicates that the USB key drive is indeed mounted, and further more that it uses a vfat filesystem, common to DOS and Windows. To see the contents of the drive, merely change directories to the mounted drive:

```
cd /media/
```

Audio

The OmniFlash-EP is capable of both audio input and output. The codec output is buffered by a simple amplifier capable of driving headphones. The audio input level is expected to be line level though, and as such a microphone will need to be amplified. The input signals are NOT brought externally from the enclosure, but are present on the board on J11.

Utilizing the 2.6 kernel ALSA (Advanced Linux Sound Architecture), the Omni-EP can play a wide range of media files using the included madplay program.

Battery Backed Clock Calendar

The Omni-EP can keep track of the time using a battery backed clock calendar chip. The driver for the program should be loaded on startup before any other devices in order to ensure that the necessary timing requirements can be met. The driver works by inserting a small function into the “stime” system call. The system time is updated when the module is loaded. Note that the resolution of the clock chip is seconds. This means that when it is loaded it is possible to lose fractions of a second. This may have user implications.

In order for the time in the clock calendar to be appropriately updated, make certain that this function is called by your user space programs. Most functions that set the time do it through this function call, and thus should be mostly transparent to the user. The clock itself has a resolution of 100 years, and as such will need to be changed in the year 2099. Dates before Jan 1, 2004 are not recognized by the driver, though the source code is included and may be modified as desired.

Jumpers

***JP1* - Flash Write Protect**
This jumper selects whether the flash memory should be write protected. This is useful in applications where security of the system is vital, and should not be changed. In order to write protect the flash, move this jumper to position 2-3.
Default: 1-2 jumpered, Flash memory not write protected.

***JP2* - Serial Boot**
This jumper is used for serially flashing the device and normally should not be changed.
Default: 1-2 Jumpered

Cables and Connectors

The following tables show the signal name (direction) for each connector pin.

To locate pin one of a connector, look for the following identifiers. Pin one has a square PCB pad and the others are round. This should be visible on the bottom of the PCB. Pin one will also be identified on the board silkscreen with a '1' and/or a dot. Dual row headers have ODD numbered pins on one side and EVEN numbered pins on the other.

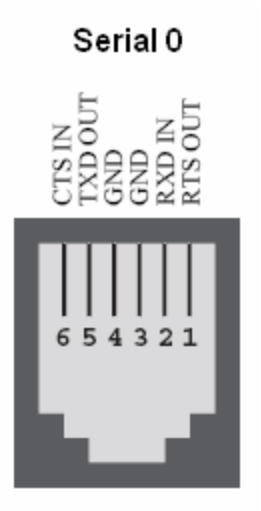
Figure 1 (Pinout)

J10 Line In

1	L_IN
2	R_IN
3	GND

J4 SPI

1	+ 5V
2	SCLK1
3	SFRM1
4	SPI_RX1
5	SPI_TX1
6	GND



J1 Port A

+5V	1	2	PA.0
PA.4	3	4	PA.1
PA.5	5	6	PA.2
PA.6	7	8	PA.3
PA.7	9	10	GND

J2 Port B

+3.3V	1	2	PB.0
PB.4	3	4	PB.1
PB.5	5	6	PB.2
PB.6	7	8	PB.3
PB.7	9	10	GND

J7 Serial 1

NC	1	2	NC
TX1	3	4	NC
RX1	5	6	NC
NC	7	8	NC
GND	9	10	NC

J8 USB

+5V F	1	2	+5V F
D1-	3	4	D2-
D1+	5	6	D2+
GND	7	8	GND
NC	9	10	NC

Software

Operating System

The Omni-EP uses a 2.6 kernel, allowing use of ALSA as well as other newer features in the kernel The kernel on its own is not enough though, the RedBoot bootloader is necessary to set up the environment and the memory manager. The user will see this bootloader on power up and reset, where it is automatically set to load and execute the linux kernel. Modifications to the bootloader should not be necessary, however if you would like more information on doing this or compiling your own kernel, please stop by the forums at http://www.jkmicro.com .

Implemented Linux Commands

Currently defined Busybox functions:

[, [[, ash, awk, basename, bunzip2, busybox, bzcat, cal, cat,
chmod, chroot, chvt, clear, cmp, cp, crond, crontab, cut, date,
dc, dd, deallocvt, df, dirname, dmesg, du, dumpkmap, echo, egrep,
env, expr, false, fbset, fdisk, fgrep, find, fold, free, freeramdisk,
ftpget, ftpput, getopt, grep, gunzip, gzip, halt, hdparm, head,
hexdump, hostid, hostname, httpd, ifconfig, ifdown, ifup, inetd,
init, insmod, install, kill, killall, klogd, last, length, linuxrc,
ln, loadfont, loadkmap, logger, logread, losetup, ls, lsmod, makedevs,
md5sum, mdev, mesg, mkdir, mkfifo, mknod, mkswap, mktemp, modprobe,
more, mount, mv, nc, netstat, nslookup, od, openvt, patch, pidof,
ping, pivot_root, poweroff, printf, ps, pwd, rdate, realpath,
reboot, renice, reset, rm, rmdir, rmmod, route, run-parts, sed,
seq, setkeycodes, sh, sha1sum, sleep, sort, start-stop-daemon,
strings, stty, swapoff, swapon, sync, syslogd, tail, tar, tee,
telnet, telnetd, test, tftp, time, top, touch, tr, traceroute,
true, tty, udhcpc, umount, uname, uncompress, uniq, unzip, uptime,
usleep, vi, watch, wc, wget, which, xargs, yes, zcat

For a more complete description of these including usage, please look at the busybox web site at http://www.busybox.net . Piping commands and redirections of input/output function as per a normal linux environment.

Additional tools:

dosfsck, e2fsck, eraseall, fsck, mkdosfs, mke2fs, mtd_debug, nwepgen, prism2dl, wlancfg, wlanctl-ng, wland, aplay, amixer

For more a complete list of functionality and options, use the --help option on the command line or just enter the executable without arguments.
For Example:

```
~ # dosfsck
usage: dosfsck [-aAflrtvVwy] [-d path -d ...] [-u path -u ...]
               device
  -a       automatically repair the file system
  -A       toggle Atari file system format
  -d path  drop that file
  -f       salvage unused chains to files
  -l       list path names
  -r       interactively repair the file system
  -t       test for bad clusters
  -u path  try to undelete that (non-directory) file
  -v       verbose mode
  -V       perform a verification pass
  -w       write changes to disk immediately
  -y       same as -a, for compat with other *fsck
~ #
```

Programming

The Omni-EP has an ARM9 processor, which means that anything programmed for an x86 machine will not function properly on ARM architecture. It is required to cross compile the code so that it can be interpreted properly. It is also recommended to do development on a Linux machine to avoid potential confusion and files attribute problems. A popular linux cross compiler (gcc) has been included on the development CD. This will allow you to effectively program the device.

To set up your cross compiler development environment, first go to the directory that your mounted CD is on. Go to the `/armgcc4.1.1` directory and copy the contents (a directory named "arm") to the `/armgcc4.1.1` directory on your development system. The cross compiler versions of gcc are in this location, and now we must make an appropriate link to them. If a link is not created, the incorrect version of gcc will be called for compiling, and the program will not work. To make a link, first find the location of gcc and use the `ls` command to link to it. If you followed the above, it should be something like this:

```
# ln /armgcc4.1.1/4.1.1-920t/bin/ my_arm_gcc
```

This will create a link to the gcc compiler called `my_arm_gcc` . This can be copied to your bin directory, or your development folder to be called at compile time. Links to g++, the assembler, and linker can also be made in a similar fashion.

Specifications

Power Supply: 5 VDC +/- 5% regulated, 2W (nominal)
Operating Temperature: -40 to +70 °C

Port A, B:

Symbol	Parameter	MIN	MAX	Units	Condition
V_{IL}	Input Low	-0.3	0.8	V	
V_{IH}	Input High	2.1	3.6	V	
V_{OL}	Output Low		1.2	V	I_{OL}= 10mA
V_{OH}	Output High	2.8		V	I_{OH}= -4mA

Mating Connectors:

Connector	Mfg	MFG P/N	Mfg	MFG P/N	JK micro P/N
2x5 Housing (J1, J2, J8)	Molex	22-55-2101	Oupiin	4072-2X05H	28-0030
Pins	Molex	16-02-0096	Oupiin	404-PIN-10K	28-0033
Power Jack (J5)	CUI	PP3-002A			88-0018
Pins, Friction Lock Housings	Molex	08-50-0114	Oupiin	4071-PIN-T	28-0013

Mechanical:

Dimensions 4.16” x 3.44" x 0.63”
105.66mm x 87.37mm x 16mm

Weight 63.96 grams

Contact Information

JK microsystems, Inc.
1403 Fifth Street, Suite D
Davis, CA 95616

Telephone: (530) 297-6073
Fax: (530) 297-6074
Email: sales@jkmicro.com (sales inquiries)
help@jkmicro.com (technical support)
Web: http://www.jkmicro.com

Change log:
1.0 First release Dec 17 2008 - AY