Table of Contents

1 Introduction

1.1 Electrum Family of Controllers

The Electrum series of single board computers are based on an ARM9 microcontroller with a vast array of peripherals including 10/100 Ethernet, analog and digital I/O?s. Through example programs and project files this family of single board computers can be developed from concept to production quickly. The Electrum family is available in standard configurations that include the more popular options used in embedded applications. Custom configurations and design services are also available for applications with special requirements.

1.2 Electrum 100 Overview

The Electrum 100 is a single board computer designed for cost-sensitive control applications that require high performance, networking and reliable multitasking capabilities. Powered by an Atmel ARM926EJ-S core capable of running at 400MHz, it can fulfill the most demanding requirements in monitoring, instrumentation, data acquisition, process control, factory automation and many other applications. An extensive array of peripherals is built-into a small Pico-ITX form factor.

1.3 Electrum 100 Features

- 400 MHz 32-bit ARM926EJ-S core
- 512 MB NAND Flash
- Micro SD Socket
- 64 MB SDRAM
- 10/100 Ethernet
- Two RS232 Serial Ports
- I2S, I2C, SSI (SPI) Ports
- Dual USB Host ports (12 Mbits per second)
- One USB Device port (12 Mbits per second)
- 6 timers, 1 watchdog timer, 1 Real-time timer, 1 Periodic interval timer
- 47 GPIO
- Debian Linux (armel)
- +5V power supply required
- Dimensions: 3.93 inches x 2.83 inches (100mm x 72mm)

1.4 Electrum 100 Options

- I/O Plus Option
    - 8-Channel 12-bit ADC
    - 4-Channel 12-bit DAC
    - 1 ? RS485 port
    - LCD Port
    - Keypad Port
- Battery circuit
    - Real time clock
    - Shutdown
    - Requires SR 44 battery

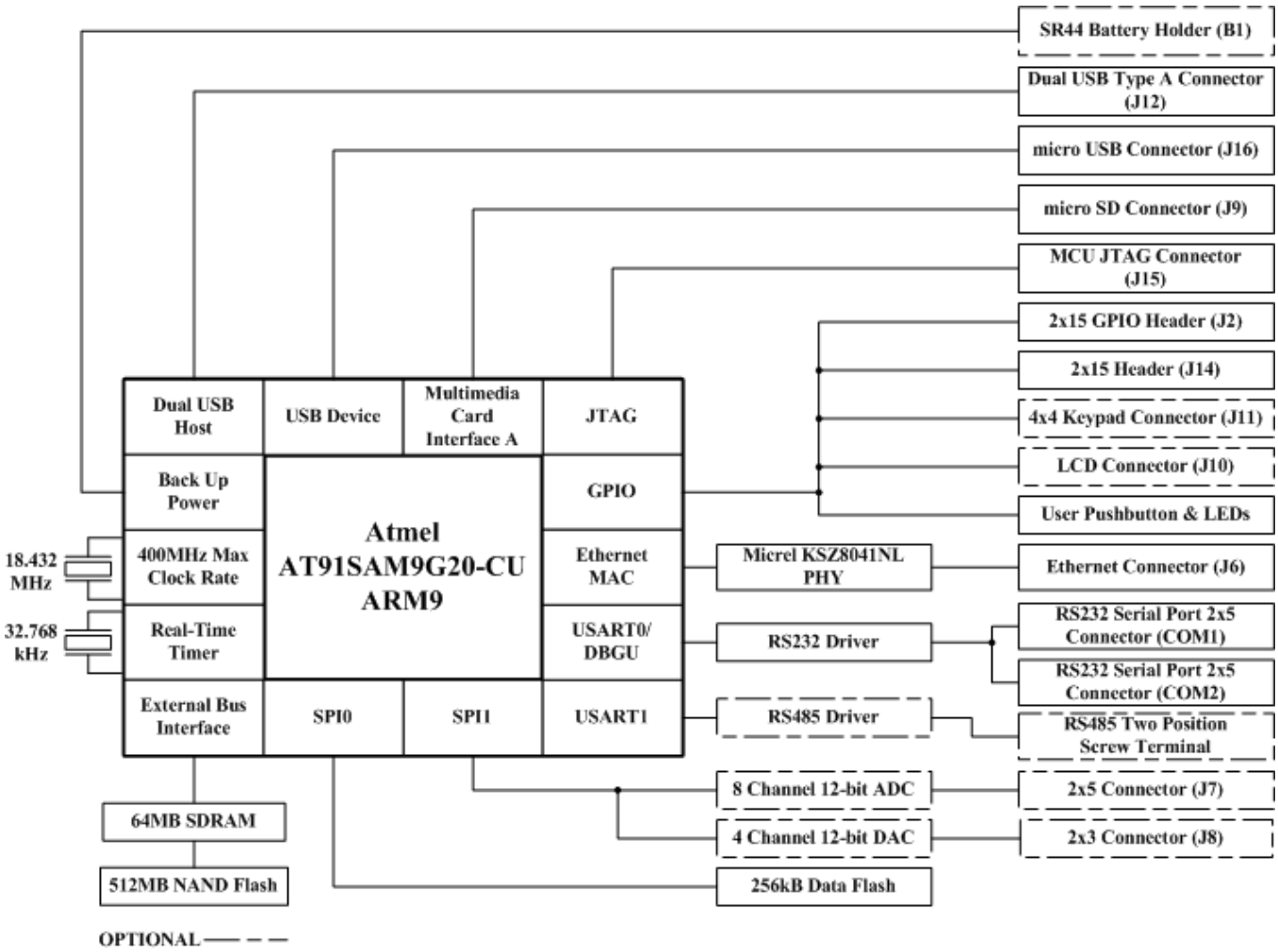


Figure 1.3: Electrum 100 Hardware Block Diagram

1.5 Software and Support

Debian Linux (armel) on the Electrum 100 provides a stable and reliable operating system base that can be easily extended using readily available libraries and applications. Linux can be loaded from NAND Flash or microSD card or an USB device, simplifying software development and distribution. Secure remote access can be implemented via web or command line interfaces, providing off-site monitoring and maintenance capabilities. Many popular Linux applications are available or can be easily ported using the GNU compiler collection as well as popular IDEs including Eclipse and Code::Blocks. x86 Linux is the recommended development environment with ARM cross compilers and other development tools readily available. If you are restricted to using a Windows PC for development, you may consider VMWare or other virtualization software that allows you to run Linux from Windows.

Thousands of popular Linux applications are available for Debian Linux to reduce application development time and simplify integration with code libraries developed for industrial and scientific environments. Using these tools, you can achieve significant functionality in a very short time. These open source tools can be easily extended, allowing a virtually unlimited number of possibilities.

Micromint USA provides free technical support by phone, email, or fax. Technical support emails are usually answered within one business day. Software and documentation updates are available on our website at www.micromint.com. Each product comes with a one year warranty.

NEXT: Getting Started

PREVIOUS: Table of Contents

2 Getting Started

This chapter covers the initial installation of the hardware and software for the Electrum 100.

2.1 Installation

To setup board parameters, a terminal or a PC running a terminal emulator should be connected to the first serial port (COM1) using a null modem cable. Connect a 5 VDC power supply and wait for a login prompt. Upon power up, an Electrum 100 with its default factory configuration will boot from Dataflash and load the kernel and filesystem from NAND. To perform the initial board setup, please login as root and enter "password" when prompted for a password. Change the password using the 'passwd' command. The date can be changed using 'date' or via NTP using 'ntpdate'.

```
Debian GNU/Linux 5.0 electrum100 ttyS0

electrum100 login: root
Password:
Last login: Wed Jun 30 09:00:23 UTC 2010 on ttyS0
Linux electrum100 2.6.33.5-at91 #1 Tue Jun 29 11:37:56 EST 2010 armv5tejl

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last was Wed Jun 30 09:10:46 2010 on ttyS0.

electrum100:~# passwd root
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully

electrum100:~# date 063012222010
Wed Jun 30 12:22:00 UTC 2010

electrum100:~# hwclock --systohc
```

The board default Ethernet configuration uses DHCP to obtain a dynamic address. To change to a static address, edit the following lines in /etc/network/interfaces .

```
# The primary network interface
auto eth0
allow-hotplug eth0
iface eth0 inet static
address 192.168.1.221
netmask 255.255.255.0
gateway 192.168.1.253
```

Before powering down, please execute the "shutdown" command from the root account to insure a proper shutdown.

```
electrum100:/# shutdown -h now

Broadcast message from root@electrum100 (ttyS0) (Wed Jun 30 12:38:26 2010):
The system is going down for system halt NOW!
...
Will now halt.
Power down.
```

Files can be transferred from the PC to the Electrum 100 using an Ethernet network, a USB flash drive or the COM1 serial port. Ethernet uploads can be made via SCP, NFS, FTP (ftpget) or TFTP. Serial uploads can be made via zmodem (rz).

2.2 Toolchain installation

To develop software for the Electrum 100, the recommended environment is a PC running x86 Linux. The Linux kernel should be 2.6.26 or above and the GCC compiler 4.3.2 or above. Debian or derivatives (including Ubuntu) are the Linux distributions directly supported by Micromint. Board software is available for download from the Electrum Wiki.

http://wiki.micromint.com/index.php/Electrum_Documentation

The recommended toolchain is the Embedian ARM cross toolchain. To install this cross toolchain under Debian or Ubuntu distributions, follow these steps:

1. Add the native compiler, make and ncurses library if they are not already in your development system.

```
# apt-get install gcc g++ make libncurses5-dev
```

2 Add the following line to /etc/apt/sources.list

```
deb http://www.emdebian.org/debian/ squeeze main
```

and update the signatures

```
# apt-get install emdebian-archive-keyring
# apt-get update
```

3. Install the following packages

```
# apt-get install linux-libc-dev-armel-cross libc6-armel-cross libc6-dev-armel-cross binutils-arm-linux-gnueabi \
  gcc-4.3-arm-linux-gnueabi g++-4.3-arm-linux-gnueabi gdb-arm-linux-gnueabi uboot-mkimage

# apt-get install apt-cross dpkg-cross
```

More information on the Embedian toolchain is available from these references:

http://wiki.debian.org/EmdebianToolchain
http://www.emdebian.org/crush/keys.html

Another alternative is the Sourcery G++ Lite for GNU/Linux from CodeSourcery. The current version of this toolchain can be can be retrieved from the URL below. Select the "IA32 GNU/Linux TAR" distribution when updating.

http://www.codesourcery.com/downloads/public/gnu_toolchain/arm-none-linux-gnueabi

The PATH change will be effective on your next login. To confirm that the toolchain is on your PATH and is working properly, try compiling a simple 'Hello' program.

```
$ arm-linux-gnueabi-gcc --version
arm-linux-gnueabi-gcc (Debian 4.3.2-1.1) 4.3.2
Copyright (C) 2008 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

$ arm-linux-gnueabi-gcc -g -march=armv5te -Os -Wall hello.c -o hello

$ file hello
hello: ELF 32-bit LSB executable, ARM, version 1 (SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.26,
 not stripped
```

To set the toolchain prefix when compiling user space applications with Sourcery G++ Lite for GNU/Linux use

```
$ export CROSS_COMPILE=arm-linux-gnueabi-
```

Some applications may also require changing other Makefile variables to use the toolchain prefix. To test your applications in the Electrum 100, you can export one of your directories via NFS and mount it from the board. Once the application is complete, you can copy the binaries to the flash filesystem. Other alternatives to copy applications to the board include SCP, FTP, TFTP or zmodem.

Other toolchains can be used with the Electrum 100 to fulfill specific application requirements. Purposes of alternate toolchains include (but are not limited to) using GPL2 licensing (gcc 4.2.1 or lower) or using uClibc as the system 'C' library. Note that using an alternate toolchain to regenerate a kernel and its operating system utilities requires significant expertise with embedded Linux.

Although x86 Linux is the preferred development environment, it is feasible to develop user space applications on Windows PCs. The recommended toolchain would be the "IA32 Windows Installer" of Sourcery G++ Lite for GNU/Linux. Installation of the Cygwin shell and utilities is also advisable if you intend to use 'make' and other tools normally available under Linux. This will allow you to compile Linux user space executables to use on the Electrum 100. One important limitation when using a Windows PC for development is that many Makefiles for Linux applications assume packages or characteristics of a Linux installation that may not be implemented in Cygwin under Windows. For example, it could be very time consuming to rebuild the Linux kernel or the board bootloader under Windows. If you are restricted to using a Windows PC for development, you may consider VMWare or other virtualization software that allows you to run Linux from Windows.

2.3 Using Remote NFS Filesystems

The kernel in the Electrum 100 can mount its root filesystem via NFS. This allows setup of a directory in a development workstation or server to act as a repository for the target files, saving time during development since files are instantly accessible from the board

without a transfer process. Once the development is released, the directory can be converted to a JFFS2 filesystem using 'mkfs.jffs2'. These are the main steps to use an NFS root filesystem with the Electrum 100: 1. Install the NFS service in your development workstation, if it is not installed already.

```
$ apt-get install nfs-kernel-server nfs-common portmap
```

2. Add the ARM root filesystem directory to your /etc/exports file and allow access to your local subnet.

```
/home/electrum100/rootfs 192.168.1.0/24(rw,sync,no_subtree_check,no_root_squash)
```

After changing the exports file, perform an update by executing ?exportfs ?a?.

3. Allow access in /etc/hosts.allow to portmap and NFS daemons to your local subnet.

```
# Allow NFS mounts from local network
portmap: 127.0.0.1 192.168.1.0/255.255.255.0
mountd: 127.0.0.1 192.168.1.0/255.255.255.0
lockd: 127.0.0.1 192.168.1.0/255.255.255.0
rquotad: 127.0.0.1 192.168.1.0/255.255.255.0
statd: 127.0.0.1 192.168.1.0/255.255.255.0
```

4. Change the boot arguments on the board to use NFS. When booting, press any key to go to the U-Boot prompt. These are the arguments to boot the board with 192.168.1.221/24 and mount the root filesystem via NFS from 192.168.1.249.

```
Electrum> setenv bootargs console=ttyS0,115200 root=/dev/nfs nfsroot=192.168.1.249:/home/electrum100/rootfs
ip=192.168.1.221:192.168.1.249::255.255.255.0::eth0:
Electrum> saveenv
Electrum> reset
```

The parameters are as follows:

```
nfsroot=<server_ip>:<mount_directory>
ip = <board_ip>:<server_ip>:<gateway_ip>:<netmask>:<hostname>:<device>:<proto>
```

The IP on the boot arguments is used until the Linux TCP/IP service is started. If the IP configuration in your bootstring and the /etc/network/interfaces on your root filesystem are not consistent, your connection will be terminated when TCP/IP starts and you will see a message of the form "nfs: server x.x.x.x not responding". To avoid this, modify /etc/network/interfaces on your root filesystem consistent with the IP config on the boot arguments.

You can export multiple directories for different purposes, e.g. a development filesystem, a production filesystem, an application-specific filesystem, etc. This can save time during the development process.

For more details on configuring NFS, please visit the following URL or contact Micromint support at support@micromint.com .

http://tldp.org/HOWTO/NFS-HOWTO/server.html

2.4 Updating the Linux kernel and root filesystem

Released versions of the the Electrum 100 Linux kernel and filesystem are included in Released versions of the the Electrum 100 Linux kernel and filesystem are included in the "boot" directory of the Tools CD. Full sources are included in the board support package (BSP) to allow development of customized kernels and/or filesystems. They can be rebuilt with the Makefile provided.

```
$ cd ~/electrum100
$ make kernel
$ make filesystem
```

The resulting kernel and filesystem will be in the "boot" subdirectory. These files can be uploaded to the Electrum 100 NAND flash using the U-Boot bootloader. Updating the NAND will require the addresses of the NAND flash partitions shown in Table 2-1.

Start	End	Description	Max Size	Device
0x00000000	0x0003FFFF	Bootstrap	256 KB	/dev/mtdblock0
0x00040000	0x0007FFFF	U-Boot executable	256 KB	/dev/mtdblock1
0x00080000	0x000BFFFF	U-Boot environment	256 KB	/dev/mtdblock2
0x000c0000	0x000FFFFF	U-Boot redundant	256 KB	/dev/mtdblock3
0x00100000	0x001FFFFF	User area	1 MB	/dev/mtdblock4
0x00200000	0x003FFFFF	Linux kernel	2 MB	/dev/mtdblock5

0x00400000	0x1FFFFFFF	Root filesystem	508 MB	/dev/mtdblock6

Table 2-1: NAND Flash Memory Map ? 512 MB (Base = 0x40000000)

NOTE: Only the kernel and filesystem partitions are used when booting from Dataflash

To access the bootloader console, connect a terminal or a PC running a terminal emulator should be connected to the first serial port (COM1) using a null modem cable. Upon power up, press any key to prevent the boot process from loading the kernel. Files can be uploaded using a USB flash drive, MMC/SD card, TFTP, NFS or a serial upload. The following would be representative commands to use a USB flash drive for NAND updates.

	Byte Count
Linux 2.6.33.20-1 Kernel	0x1E0000
Debian 5.0.9	0x3720000
Debian 6.0.4	0x3D40000

Table 2-2: Byte Counts for Kernel and File Systems

NOTE: Byte counts below correspond to the 2.6.33.20-1 kernel and 6.0.4 filesystem on the Wiki. If you use others you will need to adjust the byte counts.

```
Electrum> nand erase

NAND erase: device 0 whole chip
Skipping bad block at  0x280000000000000
Erasing at 0x1ffe000000000000 --   0% complete.
OK

Electrum> usb start
(Re)start USB...
USB:   scanning bus for devices... 2 USB Device(s) found
       scanning bus for storage devices... 1 Storage Device(s) found

Electrum> fatload usb 0 0x20100000 uImage-2.6.33.20-1-at91
reading uimage-2.6.33.20-1-at91

1939092 bytes read

Electrum> nand write 0x20100000 0x200000 0x1E0000

NAND write: device 0 offset 0x200000, size 0x1e0000
1966080 bytes written: OK

Electrum> fatload usb 0 0x20100000 rootfs.armel.jffs2
reading rootfs.armel.jffs2

64225280 bytes read

Electrum> nand write.jffs2 0x20100000 0x400000 0x3D40000

NAND write: device 0 offset 0x400000, size 0x3d40000
 64225280 bytes written: OK

Electrum> usb stop
stopping USB..

Electrum> reset
```

The NAND flash erase procedure can report some bad blocks. That is normal. They are mapped during the process and not used for storage. Note that the filesystem size (last parameter in the 'nand write' command) should match the size of the file you generated to insure the filesystem is created properly.

To copy the files from an FTP server, check that the following parameters are setup in the bootloader environment.

```
Electrum> printenv
bootargs=console=ttyS0,115200 root=/dev/mtdblock6 mtdparts=atmel_nand:256k(bootstrap)ro,256k(uboot)ro,256k(env1),
256k(env2),1024k(user),2M(linux),-(root) rw rootfstype=jffs2
bootcmd=nand read 0x22000000 0x200000 0x200000; bootm
bootdelay=1
baudrate=115200
ethact=macb0
ethaddr=00:21:a3:00:00:12
ipaddr=192.168.1.221
netmask=255.255.255.0
serverip=
bootargs=console=ttyS0,115200
```

```
stdin=serial
stdout=serial
stderr=serial

Electrum> setenv ethaddr 00:21:a3:00:00:00
Electrum> setenv ipaddr 192.168.1.221
Electrum> setenv netmask 255.255.255.0
Electrum> setenv serverip 192.168.1.249
Electrum> saveenv
```

The ethaddr should match the board serial number of the board and the serverip should be the TFTP server address. With this setup, the NAND flash can be updated via TFTP using the following commands.

```
Electrum> nand erase

NAND erase: device 0 whole chip
Skipping bad block at  0x02dc0000
Skipping bad block at  0x33b80000
Erasing at 0x3ffc0000 -- 100% complete.
OK

Electrum> tftp 0x20100000 uImage-2.6.33.20-1-at91
...

Electrum> nand write 0x20100000 0x200000 0x1E0000

NAND write: device 0 offset 0x200000, size 0x1e0000
1966080 bytes written: OK

Electrum> tftp 0x20100000 rootfs.armel.jffs2
...

Electrum> nand write.jffs2 0x20100000 0x400000 0x3720000

NAND write: device 0 offset 0x400000, size 0x3720000
57802752 bytes written: OK

Electrum> reset
```

To use NFS or serial upload (loady) please consult the U-Boot documentation at the URL below or contact Micromint support (support@micromint.com) . http://www.denx.de/wiki/U-Boot

2.5 Updating the Bootstrap and Bootloader

Released versions of the the Electrum 100 botstrap and bootloader are included in the "boot" directory of the Tools CD. Full sources are included in the board support package (BSP) to allow development of customized boot features. They can be rebuilt with the Makefile provided.

```
$ cd ~/electrum100
$ make bootstrap
$ make bootloader
```

The resulting bootstrap and bootloader will be in the "boot" subdirectory. These files can be uploaded to the Electrum 100 Dataflash using the Atmel AT91 In-System Programmer (ISP) utility included in the "tools" directory on the CD. The AT91 ISP is currently only released for Windows PCs. A test version of the AT91 ISP for Linux is available on the Atmel web site. Please check the README file in the "tools" directory of the CD. Updating the Dataflash will require the addresses of the Dataflash partitions shown in Table 2-3.

Start	End	Description	Max Size
0x00000000	0x000020FF	Bootstrap	8,448 (>8 KB)
0x00002100	0x000041FF	U-Boot environment	8,448 (>8 KB)
0x00004200	0x0003DDFF	U-Boot executable	236,544 (231 KB)
0x0003DE00	0x00041FFF	User area	16,896 (>16 KB)

Table 2-3": DataFlash Memory Map ? 256 KB (CS1 Base = 0xD0000000)

The SAM-BA utility of the AT91 ISP can upload the boot files using a Segger J-Link (JTAG) or a serial port connected to COM2 (DBGU). When updating the Dataflash via the serial port, jumper JP2 must be removed prior to powering up the board. When the AT91 ISP is loaded, a window similar to that of Figure 2-1 will be displayed. Select the desired interface and click the "Connect" button.

Figure 2-1: Atmel AT91 ISP connection window

For serial port uploads, after the connect process please replace jumper JP2 to allow access to the Dataflash. Once the connection to the board is established, a window similar to that of Figure 2-2 will be displayed. Select the "Dataflash AT45 DB" tab.

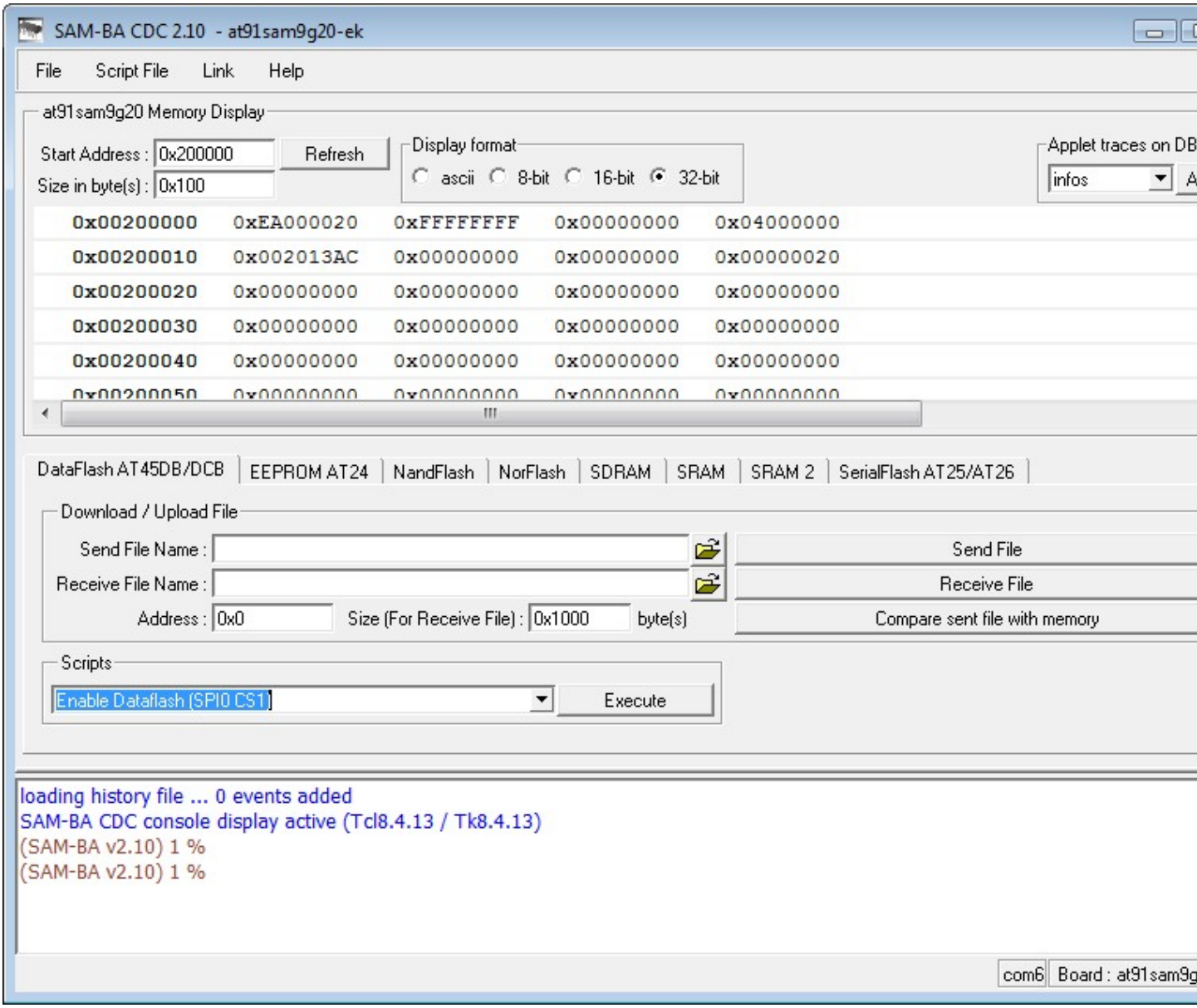


Figure 2-2: Atmel AT91 ISP Dataflash utilities

To perform the upload follow these steps:

1. Select "Enable Dataflash (SPIO CS1)" from the scripts listbox and click on "Execute".
2. Select "Erase All" from the scripts listbox and click on "Execute".

3. Select "Send boot file" from the scripts listbox and click on "Execute". Select the location and filename of the bootstrap file bootstrap-df-1.11.bin .
4. Select the location and filename of the bootloader file u-boot-df-2010.09.bin . Enter 0x4200 on the Address field and click on "Send File".
5. Exit SAM-BA and reset the board.

This procedure does not erase the bootloader data saved on the environment partition. That is erased only if you select to completely erase the Dataflash with the SAM-BA "Erase Dataflash" script. If the Dataflash is erased, after resetting the board, press a key when booting to enter a minimal bootloader environment as indicated in section 2.3 of this manual.

2.6 Memory Maps

The processor memory map and the Linux device names of internal peripherals are shown in Tables 2-4 and 2-5 and Figure 2-3. Majors and minors follow the Linux device name conventions listed on the devices-2.6+.txt file included in the "docs" directory of the Tools CD. IOCTLs available to user space applications vary according to the driver. Consult Linux device driver references for more details.

Start	End	Description
0x00000000	0x0FFFFFFF	Internal CPU memories
0x20000000	0x2FFFFFFF	SDRAM (32 ? 256 MB)
0x40000000	0x4FFFFFFF	NAND (128 MB ? 2 GB)
0xD0000000	0xDFFFFFFF	DataFlash at CS1
0xF0000000	0xFFFFFFFF	Internal CPU peripherals

Table 2-4: Main Memory Map

Start	Description	Device
0xFFFB0000	COM1 (UART0)	/dev/ttyS0
0xFFFB4000	UART1	
0xFFFB8000	UART2	
0xFFFC4000	Ethernet MAC (EMAC)	eth0
0xFFFC8000	SPI0	
0xFFFCC000	SPI1	
0xFFFFF200	COM2 (DBGU)	/dev/ttyS1
0xFFFFF400	PIOA	
0xFFFFF600	PIOB	
0xFFFFF800	PIOC	

Table 2-5: Internal Peripheral Memory Map

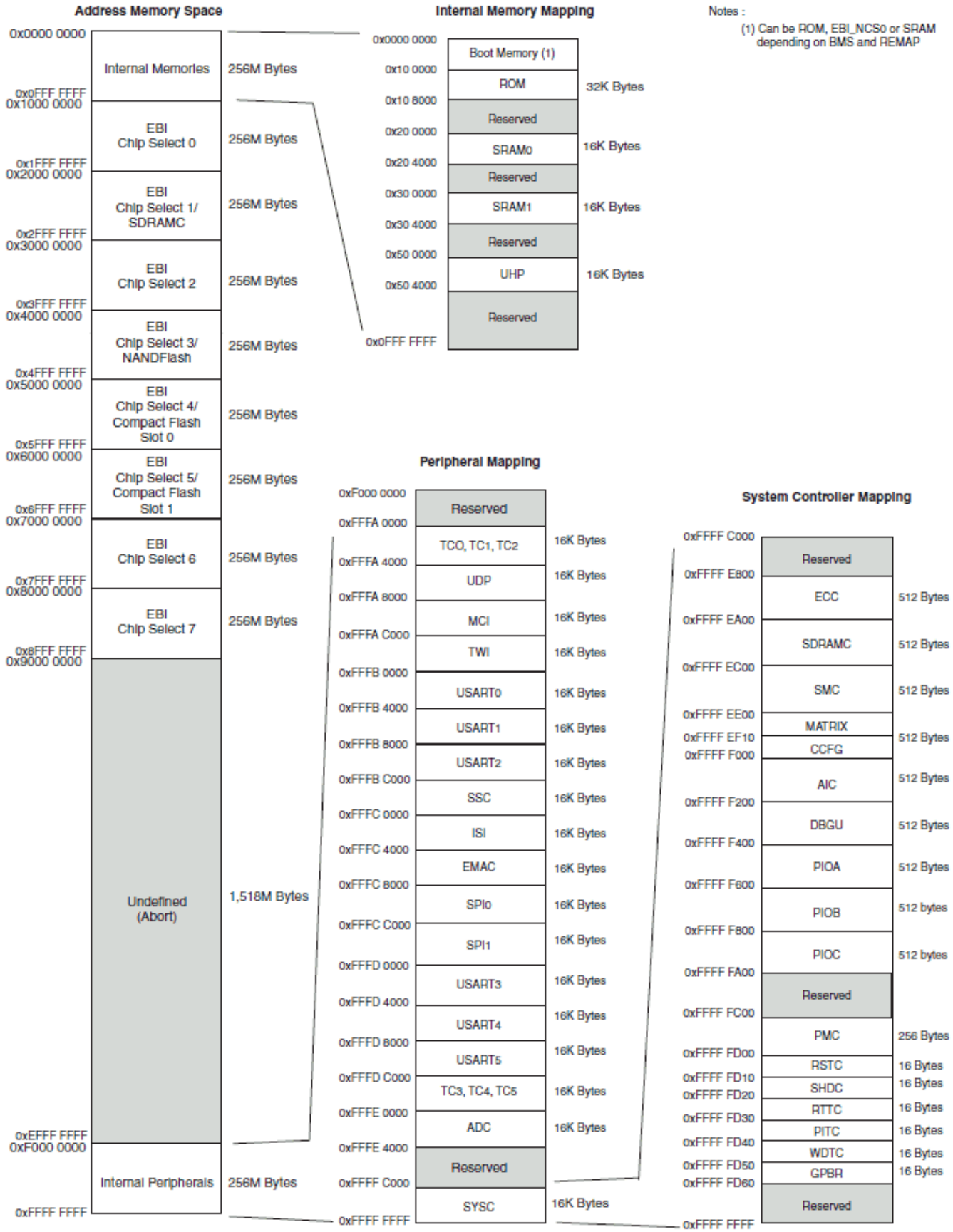

Address Memory Space
0x0000 0000
Internal Memories
256M Bytes
0x0FFF FFFF
0x1000 0000
EBI Chip Select 0
256M Bytes
0x1FFF FFFF
0x2000 0000
EBI Chip Select 1/ SDRAMC
256M Bytes
0x2FFF FFFF
0x3000 0000
EBI Chip Select 2
256M Bytes
0x3FFF FFFF
0x4000 0000
EBI Chip Select 3/ NANDFlash
256M Bytes
0x4FFF FFFF
0x5000 0000
EBI Chip Select 4/ Compact Flash Slot 0
256M Bytes
0x5FFF FFFF
0x6000 0000
EBI Chip Select 5/ Compact Flash Slot 1
256M Bytes
0x6FFF FFFF
0x7000 0000
EBI Chip Select 6
256M Bytes
0x7FFF FFFF
0x8000 0000
EBI Chip Select 7
256M Bytes
0x8FFF FFFF
0x9000 0000
Undefined (Abort)
1,518M Bytes
0xEFFF FFFF
0xF000 0000
Internal Peripherals
256M Bytes
0xFFFF FFFF
Internal Memory Mapping
0x0000 0000
Boot Memory (1)
0x10 0000
ROM
32K Bytes
0x10 8000
Reserved
0x20 0000
SRAM0
16K Bytes
0x20 4000
Reserved
0x30 0000
SRAM1
16K Bytes
0x30 4000
Reserved
0x50 0000
UHP
16K Bytes
0x50 4000
Reserved
0x0FFF FFFF
Notes :
(1) Can be ROM, EBI_NCS0 or SRAM depending on BMS and REMAP
Peripheral Mapping
0xF000 0000
Reserved
0xFFFA 0000
TC0, TC1, TC2
16K Bytes
0xFFFA 4000
UDP
16K Bytes
0xFFFA 8000
MCI
16K Bytes
0xFFFA C000
TWI
16K Bytes
0xFFFB 0000
USART0
16K Bytes
0xFFFB 4000
USART1
16K Bytes
0xFFFB 8000
USART2
16K Bytes
0xFFFB C000
SSC
16K Bytes
0xFFFC 0000
ISI
16K Bytes
0xFFFC 4000
EMAC
16K Bytes
0xFFFC 8000
SPI0
16K Bytes
0xFFFC C000
SPI1
16K Bytes
0xFFFD 0000
USART3
16K Bytes
0xFFFD 4000
USART4
16K Bytes
0xFFFD 8000
USART5
16K Bytes
0xFFFD C000
TC3, TC4, TC5
16K Bytes
0xFFFE 0000
ADC
16K Bytes
0xFFFE 4000
Reserved
0xFFFF C000
SYSC
16K Bytes
0xFFFF FFFF
System Controller Mapping
0xFFFF C000
Reserved
0xFFFF E800
ECC
512 Bytes
0xFFFF EA00
SDRAMC
512 Bytes
0xFFFF EC00
SMC
512 Bytes
0xFFFF EE00
MATRIX
0xFFFF EF10
CCFG
512 Bytes
0xFFFF F000
AIC
512 Bytes
0xFFFF F200
DBGU
512 Bytes
0xFFFF F400
PIOA
512 Bytes
0xFFFF F600
PIOB
512 bytes
0xFFFF F800
PIOC
512 bytes
0xFFFF FA00
Reserved
0xFFFF FC00
PMC
256 Bytes
0xFFFF FD00
RSTC
16 Bytes
0xFFFF FD10
SHDC
16 Bytes
0xFFFF FD20
RTTC
16 Bytes
0xFFFF FD30
PITC
16 Bytes
0xFFFF FD40
WDTC
16 Bytes
0xFFFF FD50
GPBR
16 Bytes
0xFFFF FD60
Reserved
0xFFFF FFFF

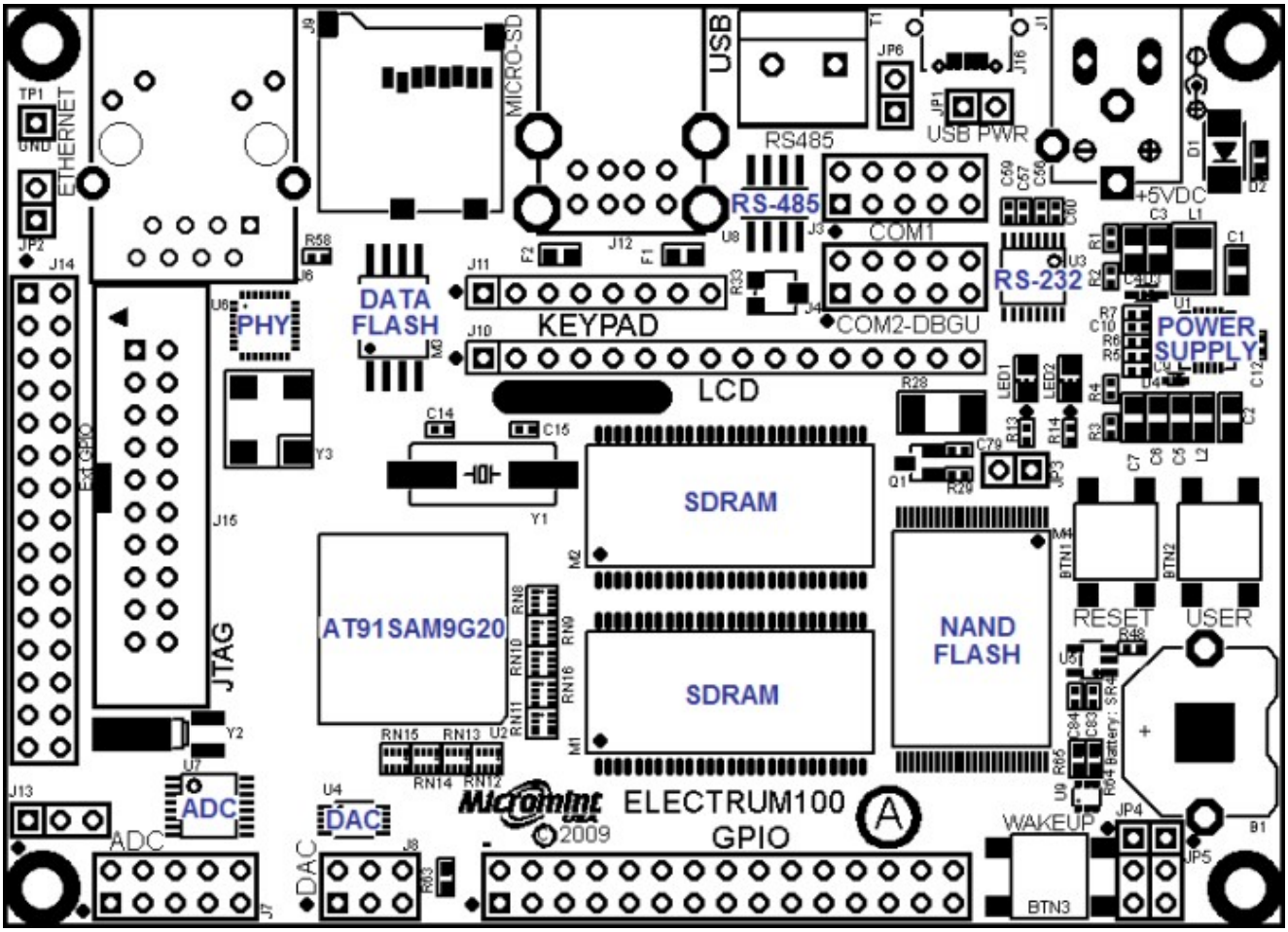
Figure 2-3: AT91SAM9G20 Memory Map

NEXT: Hardware

PREVIOUS: Introduction

3 Hardware

The following image shows where some of the hardware components are located.



Electrum 100 Hardware

3.1 Microprocessor

The Electrum 100 includes an AT91SAM9G20 microprocessor which is based on the integration of an ARM926EJ-S processor with fast ROM and RAM memories. This 32-bit ARM9 microprocessor is capable of 400-MHz operation. It has a wide range of peripherals including an Ethernet MAC, a microSD port, a USB Device Port, and dual USB Host controllers. Several standard peripherals are also included, such as USARTs, SPI bus, TWI bus (I2C), I2S, timers, and counters. Please see the Atmel Semiconductors? AT91SAM9G20 Microprocessor Data Sheet for more information and register definitions. *AT91SAM9G20 key features*

- Internal Memory
  - ♦ 64 kilo-bytes internal ROM
  - ♦ Two 16-kB internal SRAM
- Timers
  - ♦ Two Three-channel 16-bit Timer/Counters

Three External Clock Inputs, Two Multi-purpose I/O pins per channel
Double PWM Generation, Capture/Waveform Mode, Up/Down Capability
High-Drive Capability on Outputs TIOA0, TIOA1, TIOA2

- ♦ Periodic Interval Timer

20-bit interval Timer plus 12-bit Counter

- ♦ Watchdog Timer

Key protected, Programmable Only Once
Windowed 16-bit counter running on slow clock

- ♦ Real-time Timer

32-bit Free-running Backup Counter with 16-bit Prescaler

- 10/100 Ethernet MAC
  - ♦ 28-byte FIFOs
  - ♦ Dedicated DMA Channels for Receive and Transmit
- Four Universal Synchronous/Asynchronous Receiver Transmitters (USART)
  - ♦ Individual baud rate generator
  - ♦ IrDA Infrared Modulation/Demodulation, Manchester Encoding
- Two Master/Slave Serial Peripheral Interface (SPI)
  - ♦ 8-to 16-bit programmable data lengths
  - ♦ Synchronous Communications
- One Two-wire Interface (TWI) (I2C)
  - ♦ Master, multi-master or slave operation
- USB2.0 Full Speed (12 Mbits per second) Device Port
  - ♦ On-chip transceiver
  - ♦ 2,432-byte configurable DPRAM
- USB2.0 Full Speed (12 Mbits per second) Host and Dual Port
  - ♦ Integrated FIFOs
  - ♦ Dedicated DMA channels
- Three 32-bit Parallel Input/Output Controllers (PIOA, PIOB, PIOC)
  - ♦ Input change interrupt on each I/O
  - ♦ Individually programmable open-drain and pull-up resistor
  - ♦ High current drive I/O lines, up to 16mA each
- Reset Controller
  - ♦ Based on power-on reset cell
  - ♦ Reset source identification
  - ♦ Reset output control
- Additional Features
  - ♦ IEEE 1149.1 JTAG Boundry Scan on all digital pins
  - ♦ Programmable PLL for system clock

3.2 Data Flash

The Electrum 100 includes a 256k Byte Data Flash. The communication to the Data Flash is through a Serial Peripheral Interface bus (SPI). It has a maximum clock frequency of 66MHz. The Data Flash is used to store the boot-loader for the AT91SAM9G20 microprocessor. For further information please see Atmel Semiconductors AT45DB021D-SU Data Sheet.

3.3 SDRAM

The Electrum 100 includes positions for two 32MB SD RAMs for a total of 64MB. Read and write accesses to the SDRAM are burst oriented. Each access starts at a selected location and continue for a programmed number of locations. They have a self refresh mode and a 64mS, 8,192-cycle refresh. For further information please see Micron?s MT48LC16M16A2 Data Sheet.

3.4 NAND Flash

The Electrum 100 comes standard with a 512 MB NAND flash. The NAND Flash is capable of sequential reads in 25nS and can program a page in 220µS. An on-chip control logic automates program and erase operations to maximize cycle endurance. The program/erase endurance is specified at 100,000 cycles. For further information please see Micron Semiconductor?s MT29F4G08AACWC Data Sheet.

3.5 PHY

The Electrum 100 includes Micrel KSZ8041NL 10 Base-T/100 Base-TX Physical Layer Transceiver (PHY). The PHY provides MII/RMII interface to transmit and receive data. It has HP Auto MDI/MDI-X to eliminate the need to differentiate between crossover and straight-through cables. For further information please see Micrel?s KSZ8041NL Data Sheet.

3.6 DAC (Optional)

A National Semiconductor?s DAC124S085 general purpose digital-to-analog converter (DAC) is an optional feature for the Electrum 100. The DAC has four channels with a resolution of 12-bit. The output amplifiers allow for a rail-to-rail output swing from 0 to 3.3V. Communication to the DAC is done through a three wire synchronous serial interface that operates up to 40 MHz. The DAC?s outputs have a settling time of 8.5µs. It allows for simultaneous output updating. For further information please see National Semiconductor?s DAC124S085 Data Sheet.

3.7 ADC (Optional)

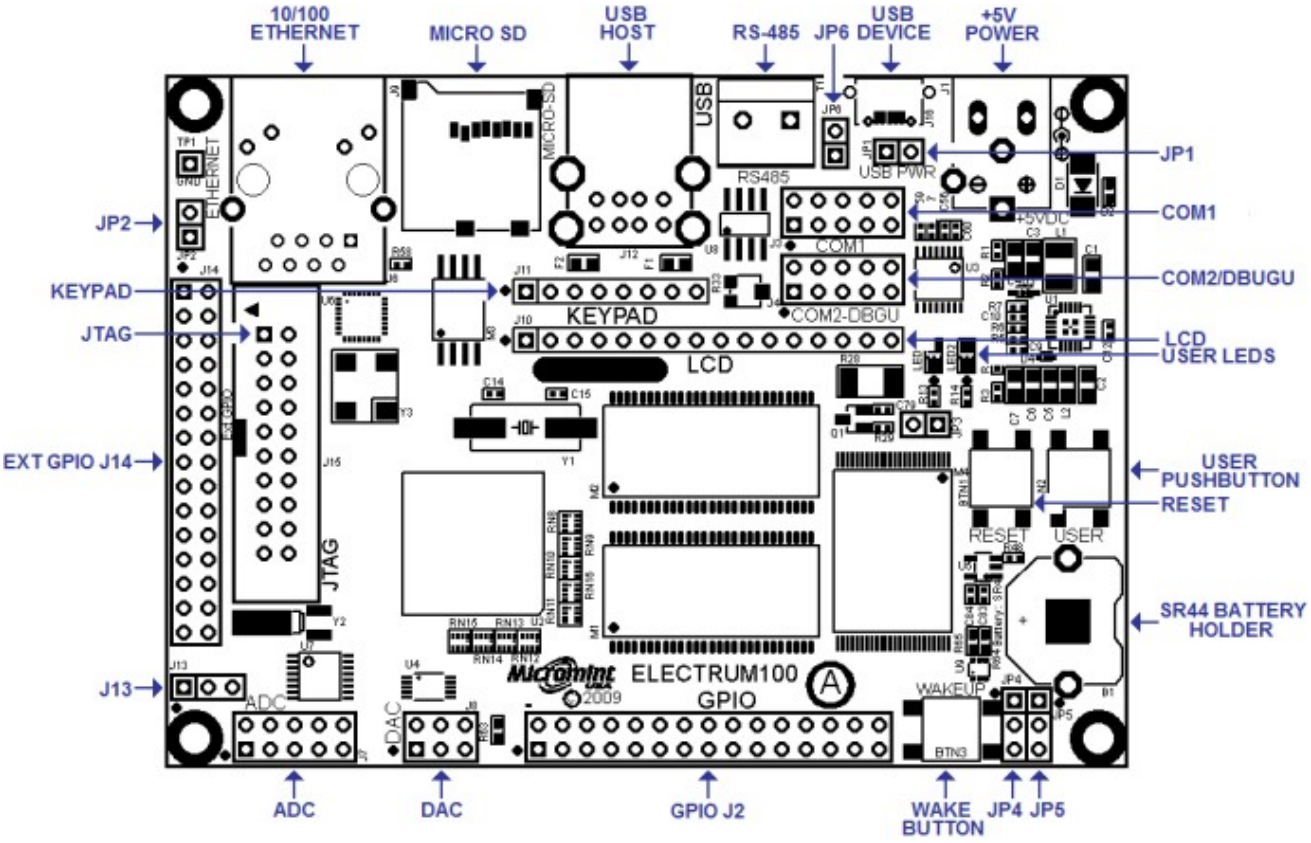
The Electrum 100 has an option to include a National Semiconductor?s ADC128S052 general purpose analog-to-digital converter (ADC). The ADC has eight channels with a resolution of 12-bit. The inputs can range from 0 to 3.3V. Communication to the ADC is done through a three wire synchronous serial interface that operates up to 8 MHz. The ADC?s inputs have a conversion rate up to 500 kSPS. For further information please see National Semiconductor?s ADC128S052 Data Sheet.

NEXT: User Interfaces, Connectors, and Jumpers

PREVIOUS: Getting Started

4 User Interfaces, Connectors, and Jumpers

The following image shows where the connectors, headers, and jumpers are located on the Electrum 100.



Electrum 100 User Interfaces, Connectors, and Jumpers

4.1 Power Supply

The Electrum 100 SBC requires a +5 VDC power supply on connector J1. Typical current requirements are 300 mA with all common peripherals enabled. When the LCD port is in use, the power supply must deliver 0.3V more than the voltage required for the LCD. This is to account for the voltage drop across the protection diode (D1). The LCD current requirements must be considered in sizing the power supply current capacity. J1 comes standard with a 2.5 mm positive center tapped female power supply jack. It can be populated with a 2 position screw terminal upon request. A diode (D1) will protect the Electrum 100 should polarity of the power supply be reversed. The board is also capable of being powered through the USB device port by putting a jumper on JP1.

WARNING: Power should not be connected to J1 and the user should make sure the board is not drawing too much power from the computers USB port.

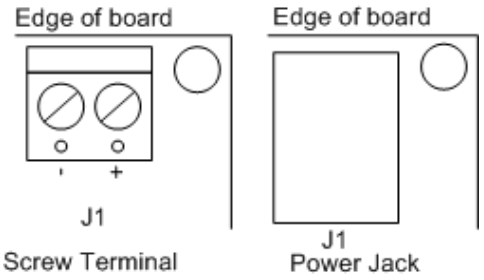


Figure 4.1: Power supply connector configurations

4.2 10/100 Ethernet

The Electrum 100 is equipped with a fully-integrated 10/100 Mbps Ethernet port. The Media Access Control (MAC) is implemented in the AT91SAM9G20 and the Physical (PHY) layer is implemented with Micrel?s KSZ8041NL. J6 is the RJ-45 connector and it has integrated magnetics and LEDs completes the Ethernet sub-system. Please see the KSZ8041NL data sheet for further information on the PHY and the AT91SAM9G20 data sheet for the MAC.

4.3 USB Host

The AT91SAM9G20 has a fully-integrated USB v2.0 Dual Host port. The host port handles full-speed and low-speed protocols. The port reaches the outside world through J12. J12 is a Dual USB Type A connector. The USB host port controller is fully compliant with the OpenHCI specification. Please see the AT91SAM9G20 data sheet for further details on the USB Host port.

4.4 Serial (COM) Ports

A Universal Synchronous Asynchronous Receivers/Transmitters (USART) is level shifted to RS-232 levels. USART0 (COM1) reaches the external world through a 2x5 pin berg header J3. The Universal Asynchronous Receiver/Transmitter (UART) Debug Unit (DBGU) is also level shifted to RS-232 levels. The DBGU (COM2/DBGU) reaches the external world through a 2x5 pin berg header J4. Please see figure 4.2 for the pin outs of COM1 and COM2/DBGU connectors. The two serial ports support software handshaking (XON/XOFF) and are considered to be Data Terminal Equipment(DTE). In order to communicate to a Personal Computer a null modem cable is required. To simplify interfacing to devices using hardware handshaking, a loopback is implemented on the modem control signals, from RTS to CTS and from DTR to CD and DSR. Note that the loopbacks do not provide flow control so software handshaking should be used when proper flow control is desired. The Electrum 100 has the option to be built with an RS-485 driver (U8). USART1 (COM3) reaches the external world through a two position screw terminal. Please see figure 4.2 for the pin out of COM3. The transmitter of the RS-485 driver is enabled by making port C bit 15 a logic 1. The RS-485 network can be terminated with a 120 ohm resistor by placing a jumper on JP3.

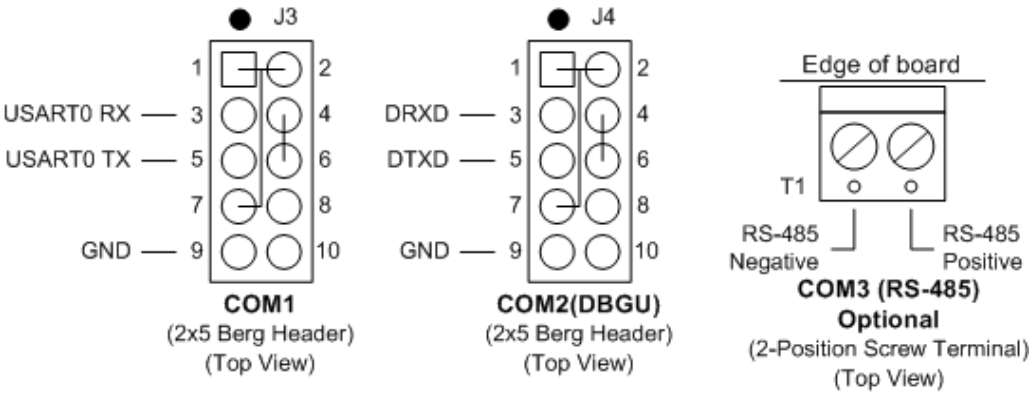


Figure 4.2: COM Ports Connector Pin Outs

4.5 Micro-SD

The micro-SD socket (J9) enables micro-secure-digital memory cards to be plugged into the Electrum 100 microcontroller board. The micro-SD card allows the user the ability of a standard removable media for transferring data to and from the Electrum 100.

4.6 General Purpose Digital Inputs and Outputs

There are fifty-three total bits of GPIO available to the Electrum 100 user. Twenty-three are available on the J2 connector, twenty-four are available on J14 and 5 are available on J11 (optional keypad connector). Please see the pin out for J2 and J14 in Figure 4.3. For the pin out of J11 please see figure 4.4 in section 4.7.

Some of the GPIO on J2 and J14 have alternate functions other than digital inputs and outputs and are shared with some of the hardware on the board. Table 4.1 lists all of the alternate functions that are available to the Electrum 100 user and what connector they are available on. Table 4.2 lists the alternate functions, the hardware it is shared with, and a brief description of the alternate function for connector J2. Table 4.3 lists the alternate functions, the hardware it is shared with, and a brief description of the alternate function for connector J14. For further information on the alternate functions please refer to the AT91SAM9G20 data sheet.

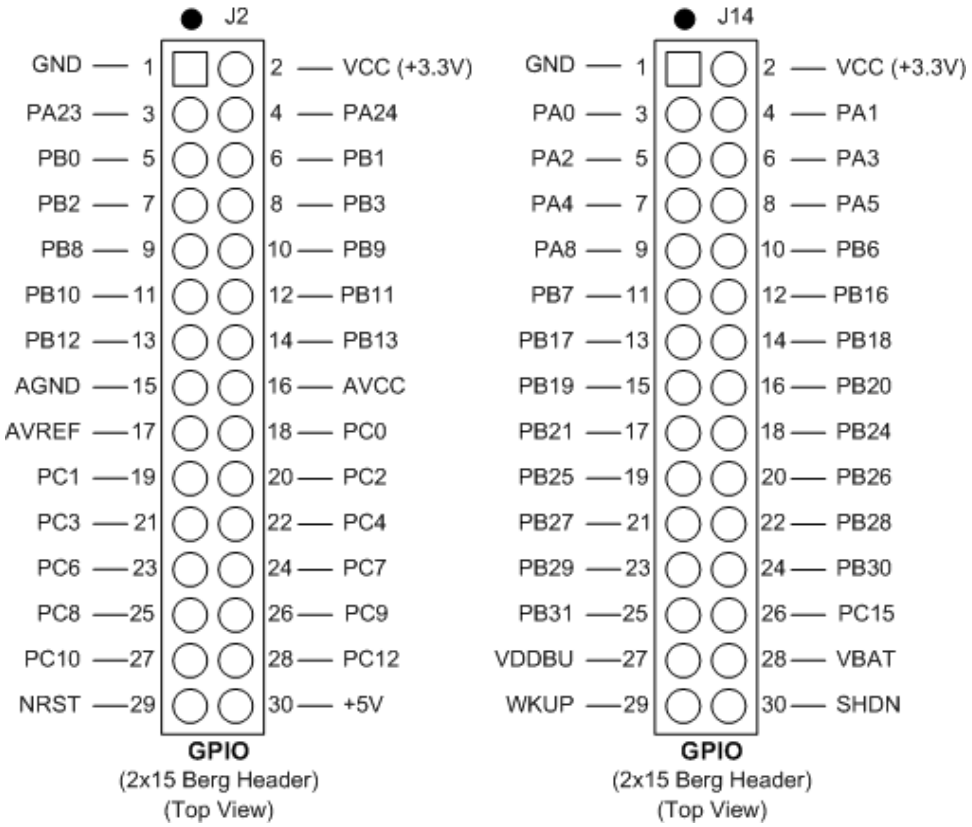


Figure 4.3: GPIO Connector Pin Outs

Alternate Function	Available on Connector
Programmable Clocks	J14
External Interrupt 0 (IRQ0)	J2
External Interrupt 1 (IRQ0)	J14
Multimedia Card Slot B	J14
USART0 Control Lines	J14, J11
USART1	J14
USART2	J2, J14
USART3	J2
USART5	J2
Synchronous Serial Controller	J14
Timer Counter 0	J2, J11
Timer Counter 1, 2	J2, J14, J11,
Timer Counter 3, 4, and 5	J2, J14
Two-Wire Interface	J2
Image Sensor	J2, J11, J14

Analog to Digital Converter	J2

Table 4.1: Alternate Function Availability

J2 Pin#	Signal	Alternate Function	Notes	Shared Hardware
1	GND		Digital Ground	
2	VCC		3.3Volts DC	
3	PA23	TWD	Two-wire Data	
4	PA24	TWCK	Two-wire Clock	
5	PB0	SPI1_MISO/TIOA3	SPI Channel 1 Master Input, Slave Output/Counter 3 I/O line A	Optional ADC, DAC
6	PB1	SPI1_MOSI/TIOB3	SPI Channel 1 Master Output, Slave Input/Counter 3 I/O line B	Optional ADC, DAC
7	PB2	SPI1_SPCK/TIOA4	SPI Channel 1 Clock/Timer Counter 4 I/O line A	Optional ADC, DAC
8	PB3	SPI1_NPCS0/TIOA5	SPI1 - Slave Select 0/Timer Counter 5 I/O line A	
9	PB8	TXD2	COM4 (USART2) TX	
10	PB9	RXD2	COM4 (USART2) RX	
11	PB10	TXD3/ISI_D8	COM5 (USART3) TX/Image Sensor DATA8	
12	PB11	RXD3/ISI_D9	COM5 (USART3) RX/Image Sensor DATA9	
13	PB12	TXD5/ISI_D10	COM6 (USART5) TX/Image Sensor DATA10	Keypad (J11)
14	PB13	RXD5/ISI_D11	COM6 (USART5) RX/Image Sensor DATA11	Keypad (J11)
15	AGND		Analog ground	
16	AVCC		Analog power (3.3V)	
17	AVREF		Analog Reference Voltage	
18	PC0	AD0/SCK3	Analog Input 0/USART3 Serial Clock	
19	PC1	AD1/PCK0	Analog Input 1/Programmable Clock Output 0	
20	PC2	AD2/PCK1	Analog Input 2/Programmable Clock Output 1	
21	PC3	AD3/SPI1_NPCS3	Analog Input 3/ SPI1 - Slave Select 3	DAC SYNC
22	PC4	SPI1_NPCS2	SPI1 - Slave Select 2	ADC Chip Select
23	PC6	TIOB2	Timer Counter 2 I/O line B	LCD (J10)
24	PC7	TIOB1	Timer Counter 1 I/O line B	LCD (J10)
25	PC8	RTS3	USART3 Request to Send	LCD (J10)
26	PC9	TIOB0	Timer Counter 0 I/O line B	
27	PC10	CTS3	USART3 Clear to Send	LCD (J10)
28	PC12	IRQ0	External Interrupt 0	
29	NRST		Microcontroller Reset	
30	+5V		+5 Volts DC	

Table 4.2: GPIO Alternate Functions for J2

J14 Pin#	Signal	Alternate Functions	Notes	Shared Hardware
1	GND		Digital Ground	
2	VCC		3.3Volts DC	
3	PA0	SPI0_MISO/ MCDB0	SPI0 - Master In Slave Out/ Multimedia Card Slot B DATA0	Data Flash
4	PA1	SPI0_MOSI/ MCCDB	SPI0 - Master Out Slave In/ Multimedia Card Slot B Command	Data Flash
5	PA2	SPI0_SPCK	SPI0 - Serial Clock	Data Flash
6	PA3	SPI0_NPCS0/ MCDB3	SPI0 - Slave Select 0/ Multimedia Card Slot B DATA3	
7	PA4	RTS2/ MCDB2	USART2 Request to Send/ Multimedia Card Slot B DATA2	
8	PA5	CTS2/ MCDB1	USART2 Clear to Send/ Multimedia Card Slot B DATA1	
9	PA8	MCCK	Multimedia Card Clock	microSD
10	PB6	TXD1/TCLK1	COM3 (USART1) TX/Timer 1 External Clock Input	Optional RS-485
11	PB7	RXD1/TCLK2	COM3 (USART1) RX/Timer 2 External Clock Input	Optional RS-485
12	PB16	TK0/TCLK3	SSC0 Transmitter Clock/Timer 3 External Clock Input	
13	PB17	TF0/TCLK4	SSC0 Transmitter Frame/Timer 4 External Clock Input	
14	PB18	TD0/TIOB4	SSC0 Transmitter Data/Timer Counter 4 I/O line B	
15	PB19	RD0/TIOB5	SSC0 Receiver Data/Timer Counter 5 I/O line B	
16	PB20	RK0/ISI_D0	SSC0 Receiver Clock/Image Sensor DATA0	
17	PB21	RF0/ISI_D1	SSC0 Receiver Frame/Image Sensor DATA1	
18	PB24	DTR0/ ISI_D4	USART0 Data Transmit Ready/Image Sensor DATA4	LCD (J10)
19	PB25	RI0/ ISI_D5	USART0 Ring Indicator/Image Sensor DATA5	LCD (J10)
20	PB26	RTS0/ISI_D6	USART0 Request to Send/Image Sensor DATA6	LCD (J10)
21	PB27	CTS0/ISI_D7	USART0 Clear to Send/Image Sensor DATA7	LCD (J10)
22	PB28	RTS1/ISI_PCK	USART1 Request to Send/Image Sensor Data Clock	LCD (J10)
23	PB29	CTS1/ISI_VSYNC	USART1 Clear to Send/Image Sensor Vertical Synchro	LCD (J10)
24	PB30	PCK0/ISI_HSYNC	Programmable Clock Output 0/Image Sensor Horizontal Synchro	LCD (J10)
25	PB31	PCK1/ISI_MCK	Programmable Clock Output 0/Image Sensor Reference Clock	LCD (J10)
26	PC15	NWAIT/IRQ1	External Wait Signal/External Interrupt Input	Optional RS-485
27	VDDBU		Powers the Slow Clock oscillator	
28	VBAT		Powers the Slow Clock oscillator through a 1.0V LDO	Battery Option
29	WKUP	-	Wake-up Input	
30	SHDN	-	Shutdown Control	

Table 4.3: GPIO Alternate Functions for J14

4.7 Keypad

A 4x4 matrix keypad using a 16-pin (2x8) ribbon cable can be connected to the microprocessor through J11. Please see Figure 4.4 for the pin out of the keypad connector. If the ports are not used for a keypad they may be used for their alternate functions. Table 4.4 lists the alternate functions for the 8-bits of I/O connected to J11.

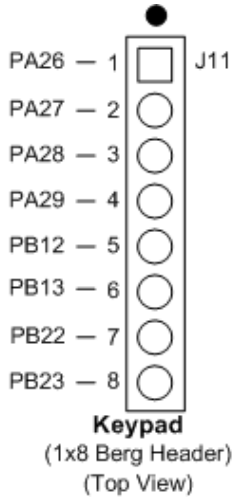


Figure 4.4: Keypad connector pin out

J11 Pin#	Signal	Alternate Functions	Notes	Shared Hardware
1	PA26	TIOA0	Timer Counter 0 I/O line A	
2	PA27	TIOA1	Timer Counter 1 I/O line A	
3	PA28	TIOA2	Timer Counter 2 I/O line A	
4	PA29	SCK1	USART1 Serial Clock	
5	PB12	TXD5/ISI_D10	COM6 (USART5) TX/Image Sensor DATA10	J2
6	PB13	RXD5/ISI_D11	COM6 (USART5) RX/Image Sensor DATA11	J2
7	PB22	DSR0 /ISI_D2	COM1 (USART0) Data Set Ready/ Image Sensor DATA2	
8	PB23	DCD0 /ISI_D3	COM1 (USART0) Data Carrier Detect/ Image Sensor DATA3	

Table 4.4: Keypad GPIO Alternate Functions

4.8 Liquid Crystal Display (LCD)

A standard alphanumeric LCD may be connected to J10 through a 32-pin (2x16) ribbon cable. The contrast for the LCD may be adjusted by turning potentiometer R33 located between J11 (Keypad) and J4 (COM2/DBGU). Please see figure 4.5 for the LCD?s connector pin out. If the LCD is not used then the GPIO may be used for their alternate functions and can be connected to via J2 and J14. Table 4.5 lists the alternate functions for the 8-bits of I/O connected to J11. AZ Displays ACM2004D series is recommended for use with the Electrum 100. http://www.azdisplays.com/index.php?id=Character_Modules&product=c2004d

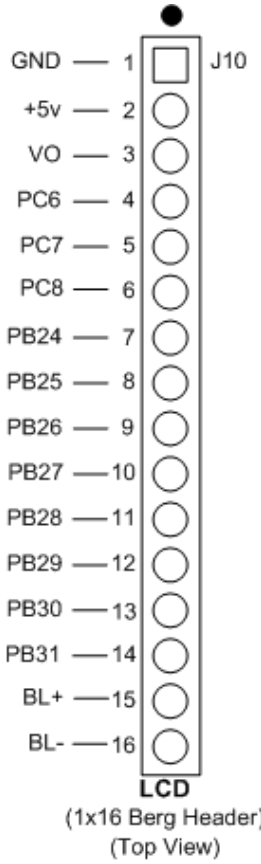


Figure 4.5: LCD connector pin out

J10 Pin#	Signal	Alternate Functions	Notes	Shared Hardware
1	GND		Digital Ground	
2	+5V		+5 Volts DC	
3	VO		LCD Contrast	
4	PC6	TIOB2	Timer Counter 2 I/O line B	J2
5	PC7	TIOB1	Timer Counter 1 I/O line B	J2
6	PC8	RTS3	USART3 Request to Send	J2
7	PB24	DTR0/ ISI_D4	USART0 Data Transmit Ready/Image Sensor DATA4	J14
8	PB25	RI0/ ISI_D5	USART0 Ring Indicator/Image Sensor DATA5	J14
9	PB26	RTS0/ISI_D6	USART0 Request to Send/Image Sensor DATA6	J14
10	PB27	CTS0/ISI_D7	USART0 Clear to Send/Image Sensor DATA7	J14
11	PB28	RTS1/ISI_PCK	USART1 Request to Send/Image Sensor Data Clock	J14
12	PB29	CTS1/ISI_VSYNC	USART1 Clear to Send/Image Sensor Vertical Synchro	J14
13	PB30	PCK0/ISI_HSYNC	Programmable Clock Output 0/Image Sensor Horizontal Synchro	J14
14	PB31	PCK1/ISI_MCK	Programmable Clock Output 0/Image Sensor Reference Clock	J14

15	BL+	LCD Backlight Anode
16	BL-	LCD Backlight Cathode

Table 4.5: LCD GPIO Alternate Functions

4.9 JTAG

The JTAG port can be used for software download and debugging, reducing the need for an in-circuit emulator. For detailed information on the operation of the JTAG port with boundary scan, please refer to IEEE Standard 1149.1-Test Access Port and Boundary-Scan Architecture.

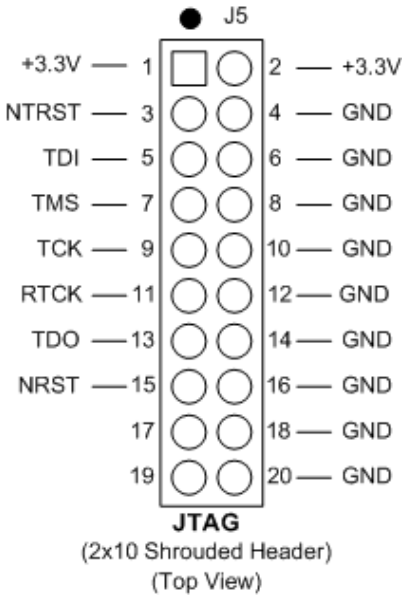


Figure 4.6: JTAG connector pin out

4.10 Pushbuttons and LED

The Electrum 100 comes standard with a user pushbutton, a reset push button, and two user LEDs. The user push button is connected to port A bit 31 with a 10kO pull-up resistor connected to it. User LED 1 is a yellow LED and is connected to port A bit 25. User LED 2 is a green LED and is connected to port A bit 30. Both LEDs can be illuminated by making the corresponding bit a logic low.

4.11 Optional Analog to Digital Converter (ADC)

The Electrum 100?s optional eight channels of 12-bit ADC can be connected to through J7. Please see figure 4.7 for the pin out of the ADC connector. The ADC is accessed through the AT91SAM9G20 SPI1 port. SPI1 slave select 2, port C bit 4, is the ADC?s chip select input for reading the conversion counts of the ADC.

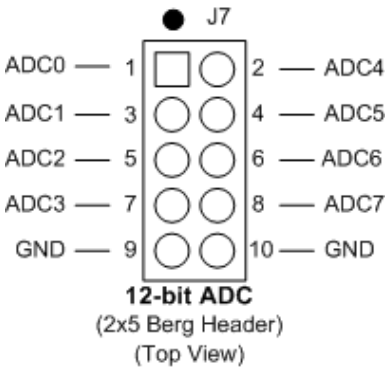


Figure 4.7: Analog to Digital connector pin out

4.12 Optional Digital to Analog Converter (DAC)

The optional four channels of 12-bit DAC can be connected to through J8. Please refer to figure 4.8 for the pin out of the DAC connector. The DAC is accessed through the AT91SAM9G20 SPI1 port. SPI1 slave select 3, port C bit 3, is the DAC?s sync input for loading the conversion count into the DAC.

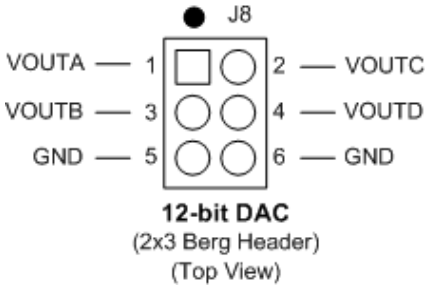


Figure 4.8: Digital to Analog connector pin out

4.13 USB Device

The Electrum 100 comes equipped with a USB Device Port. The Device port is compliant with the USB V2.0 full-speed device specification. It reaches the outside world through J16. J16 is a micro USB Type AB connector. The USB device port has six endpoints that can be configured in one of several USB transfer types. Port C bit 5 is connected to the device ports VBUS power through a voltage divider. This allows the processor to detect when it is being used as a USB device. Please see the AT91SAM9G20 data sheet for further details on the USB Device port.

4.14 Option Jumpers

The Electrum 100?s option jumpers are used to enable memories and set-up power options. JP1 is used to power the Electrum 100 through the USB Device Port. WARNING: Power should not be connected to J1 and the user should make sure the board is not drawing too much power from the computers USB port. Figure 4.8 is the pin out for JP1.

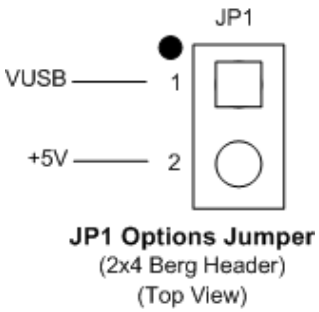


Figure 4.8: USB Device Power Option Jumper

JP4 is used to determine how VDDBU is generated and is only populated if the battery option is present. When a jumper is installed on pins 1 and 2 of JP4, VDDBU is generated from VBAT and if the system loses power the shutdown controller will remain powered from the battery. VBAT can be applied on J14 pin 28 or from the SR44 battery holder. When JP4 is set to pins 2 and 3 VDDBU is generated by the VCORE switching regulator. Figure 4.9 contains the pin out for JP4.

JP5 is used to implement the SHDN pin for shutdown controller. When a jumper is installed on pins 1 and 2 of JP5, the SHDN pin controls the enable pin for the switching regulator. When JP4 is set to pins 2 and 3 of JP5 the outputs for the switching regulator is always enabled. Figure 4.9 contains the pon out for JP5.

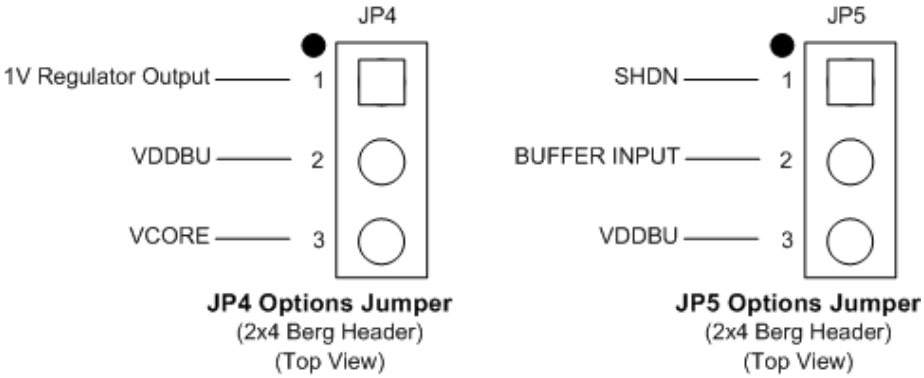


Figure 4.9: VDDBU Option Jumper

JP6 is used to terminate the optional RS-485 network through a 120 resistor. Figure 4.10 is the pin out for JP6.

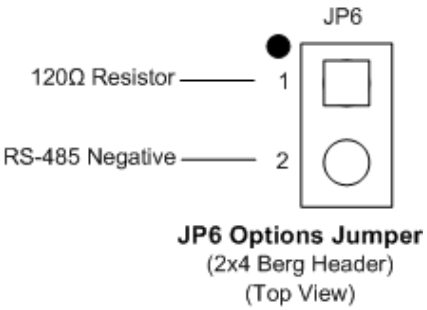


Figure 4.10: RS-485 Network Termination Option Jumper

JP2 is used to connect the chip select signal to the serial dataflash. JP3 is used to connect the chip enable to the NAND Flash. Figure 4.11 contains the pin out of JP2 and JP3.

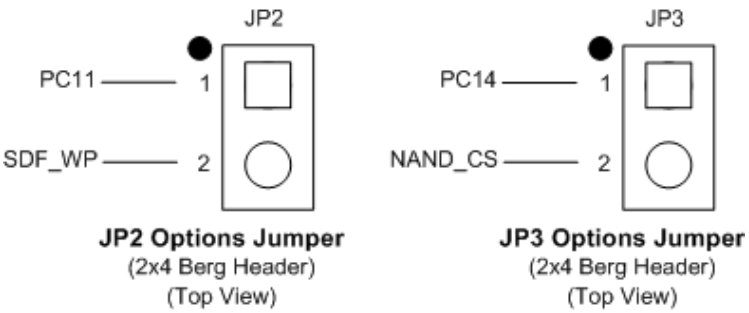


Figure 4.11: Memory Enable Option Jumpers

The analog reference voltages can be easily measured on J13. Figure 4.12 is the pin out for J13.

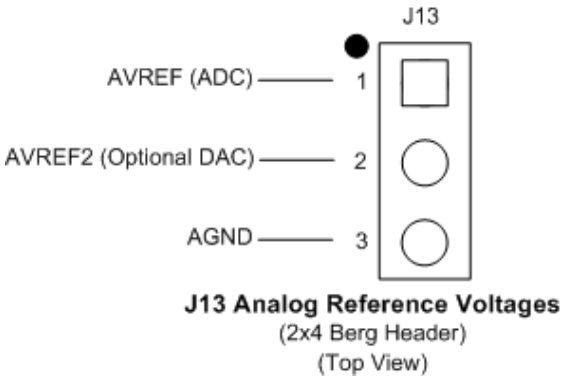


Figure 4.12: Analog Reference Voltage

NEXT: References

PREVIOUS: Mechanical and Electrical Characteristics

5 Mechanical and Electrical Characteristics

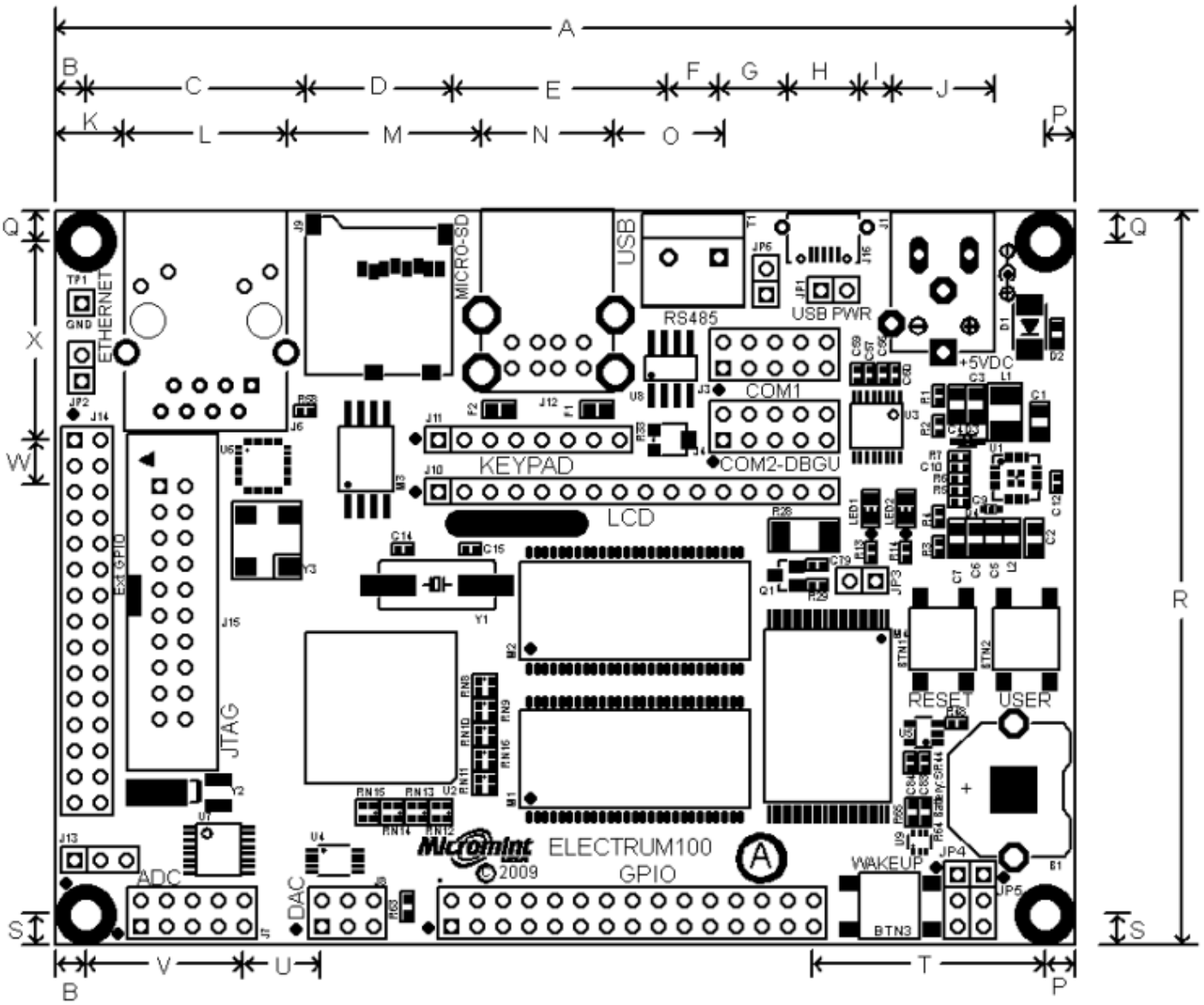
5.1 Absolute Minimum and Maximum Ratings

Characteristic	Minimum	Maximum	Unit
Voltage on J1	4.8	5.8	VDC
Voltage on VBAT(J14)		6.5	VDC
Voltage on ADC	0.0	3.6	VDC
Voltage on Digital Input	0.0	3.6	VDC
Operating Temperature	0	70	ºC
Storage Temperature	-50	125	ºC

The Electrum 100 SBC is currently available for commercial temperature ranges. Contact the Micromint sales department for industrial temperature range availability.

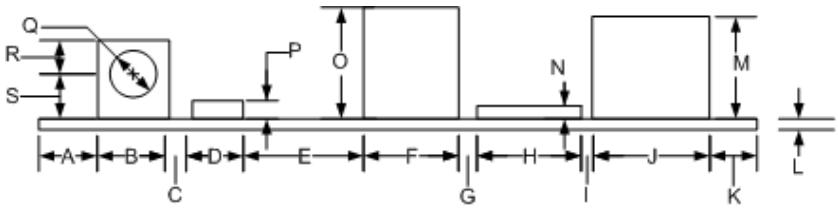
5.2 Mechanical Dimensions

Below is the physical dimensions for the Electrum 100. The mounting holes will accept a #4 size screw.



DIM	Inches	Millimeters	DIM	Inches	Millimeters	DIM	Inches	Millimeters	DIM	Inches	Millimeters
A	3.94	100.0	**G**	0.266	6.75	**M**	0.75	19.05	**S**	0.115	2.92
B	0.116	2.95	**H**	0.275	6.98	**N**	0.511	12.97	**T**	0.895	22.73
C	0.851	21.61	**I**	0.124	3.14	**O**	0.406	10.31	**U**	0.3	7.62
D	0.567	14.4	**J**	0.4	10.16	**P**	0.117	2.97	**V**	0.608	15.44
E	0.828	21.03	**K**	0.265	6.74	**Q**	0.12	3.04	**W**	0.178	4.52
F	0.2	5.08	**L**	0.63	16.0	**R**	2.835	72.0	**X**	0.76	19.3

Electrum 100 Mechanical Dimensions



SIDE VIEW

DIM	Inches	Millimeters	DIM	Inches	Millimeters	DIM	Inches	Millimeters	DIM	Inches	Millimeters
A	0.33234	8.4415	**F**	0.511	12.98	**K**	0.261	6.63	**P**	0.136	3.45
B	0.3543	9.0	**G**	0.110	2.794	**L**	0.06	1.524	**Q**	0.269	6.83
C	0.117	2.97	**H**	0.56693	14.40	**M**	0.545	13.843	**R**	0.177	4.5
D	0.345	8.763	**I**	0.072	1.83	**N**	0.1	2.54	**S**	0.256	6.5
E	0.638	16.21	**J**	0.63	16.00	**O**	0.633	16.068			

Electrum 100 Suggested Openings

NEXT: References

PREVIOUS: User Interfaces, Connectors, and Jumpers

6 References

This section outlines material that may be useful for further reading.

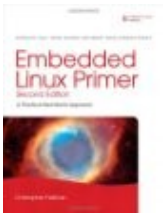
6.1 Documents

AT91SAM9G20 Microcontroller Data Sheet
http://www.atmel.com/dyn/products/product_card.asp?part_id=4337
This data sheet provides reference information for the AT91SAM9G20 microcontroller, describing the functional blocks of the system-on-chip (SoC) device designed around the ARM926EJ-S processor. All MCU registers are described in the data sheet.

at91lib Software Library
http://www.atmel.com/dyn/products/tools_card.asp?tool_id=4343
AT91 peripheral drivers and code examples. Shows which processor registers are commonly used with the microcontroller peripherals. Some of the logic can be used under Linux user space applications by invoking mmap() to map the peripheral base address to a virtual address.

Writing device drivers in Linux: A brief tutorial
http://www.scss.tcd.ie/~simonn/ON3/device_driver_tutorial.pdf
http://www.freesoftwaremagazine.com/articles/drivers_linux
This article by Xavier Calbet published on Free Software Magazine is a good introduction to Linux device drivers.

6.2 Books

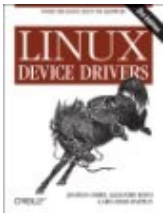


Embedded Linux Primer: A Practical Real-World Approach (2nd Edition)

by Christopher Hallinan

ISBN: 0137017839 Publisher: Prentice Hall (November, 2010)

Excellent reference on embedded Linux including processor selection, bootloaders, kernel building, device drivers, file systems and remote debugging.

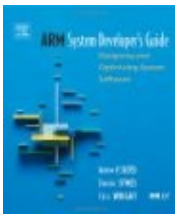


Linux Device Drivers, 3rd Edition

by Jonathan Corbet, Alessandro Rubini, Greg Kroah-Hartman

ISBN: 0131679848 Publisher: O'Reilly Media (February, 2005)

Covers the Linux interfaces to hardware and techniques for writing device drivers, including ports, memory management, timers and interrupts. Free online versions are available in HTML and PDF formats.



ARM System Developer's Guide: Designing and Optimizing System Software

ISBN: 0596005903 Publisher: Morgan Kaufmann; (March, 2004)

In-depth overview of the ARM architecture with examples that outline impact of programming practices on performance, power and cost.

6.3 Useful Web Links

Micromint Web Site
http://www.micromint.com/
Product information and software updates for the Electrum SBCs.

Atmel ARM Solutions
http://www.atmel.com/products/AT91/default.asp?family_id=605
Manuals and application notes for the Atmel ARM microcontrollers.

AT91 Community
http://www.at91.com/

Community site for users and developers of AT91 applications.

Meld Embedded Linux Community
http://meld.org/
Meld is a community for embedded Linux developers sponsored by MontaVista Software.

PREVIOUS: Mechanical and Electrical Characteristics