Wireless-Tag ARM926-DK

ARM926-CB Development Board

DATASHEET

Description

ARM926-DK is a peripheral hardware interface expansion boardfor the use of company's ARM926-CB core application development board , expanded USB , UART, Ethernet and other common interface of the ATMEL AT91SAM9G25-CU chip, convenient for customers to quickly develop a AT91SAM9G25-CU chip application, to provide direct hardware platform for customers all kinds of scheme designs and system applications.

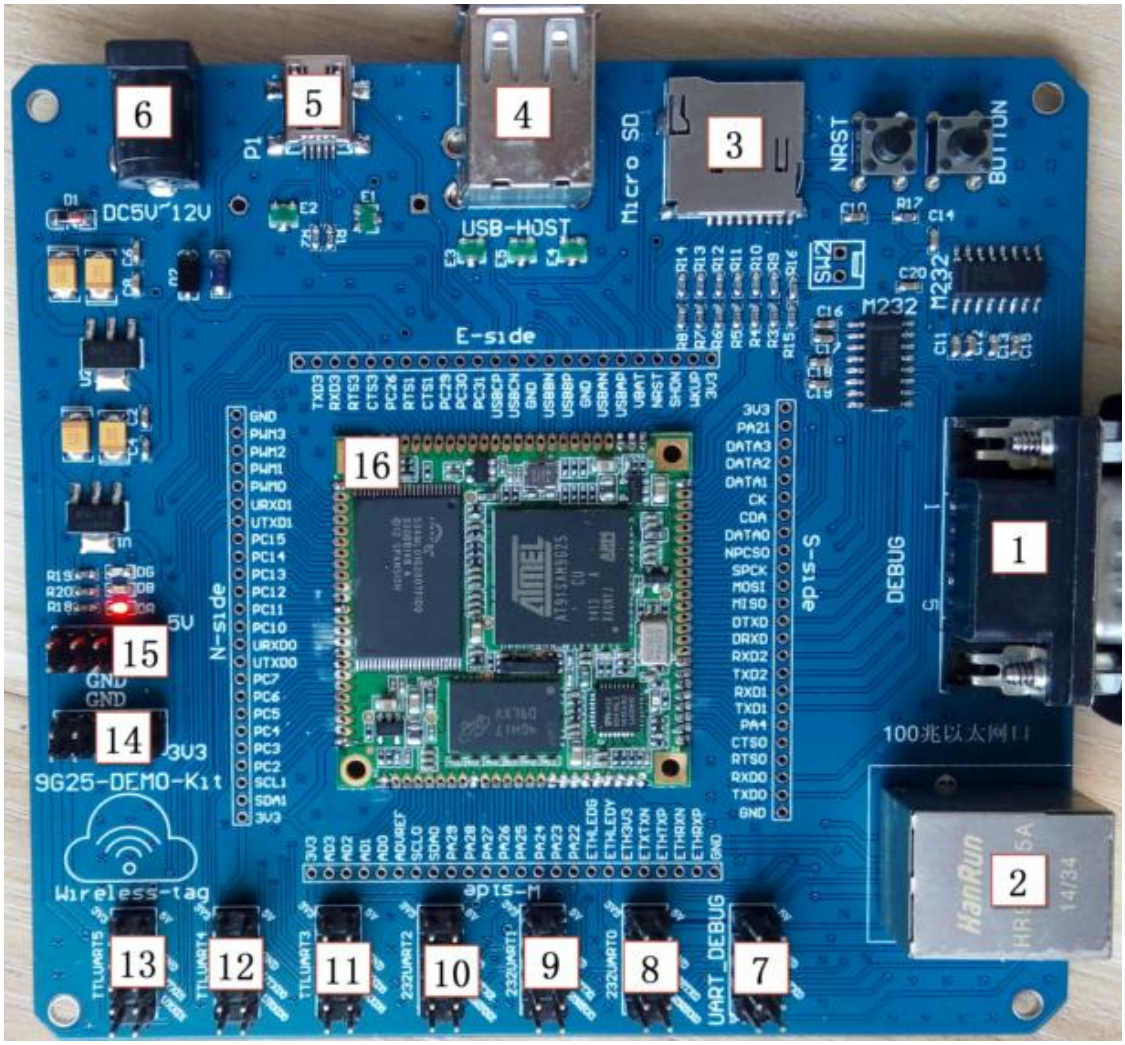
Core configuration：

CPU: ARM9 @ 400Mhz on Atmel AT91SAM9G25-CU

RAM: 128MByte DDR2

Flash: 128MByte Nand Flash

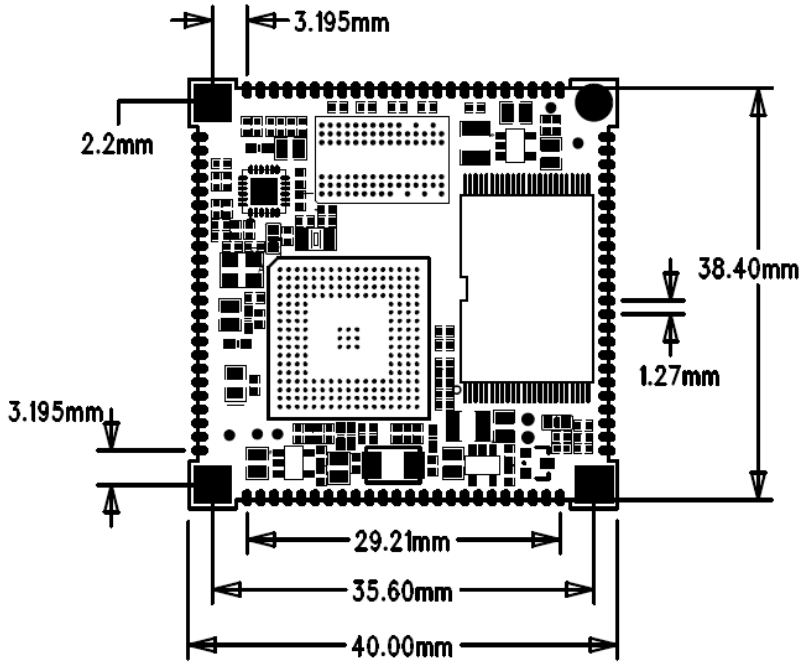
ARM926-DKHardware Interface Overview



NO.	Interface	Note
1.	DEBUG Interface	RS232Interface level，can connect directly with the COM port of the circuit
2.	RJ45 Interface	RJ45Interface, 10M/100Madaptive
3.	Micro SD Card Interface	Self clip SD card slot, press down and stuck the SD card, then repress and bounce
4.	USB Host Interface	Double USB interface
5.	Micro USB	The USB interface can also be used as the USB for the bottom plate to supply power.
6.	DC POWER IN	6V~9VPower input interface
7.	DEBUG_UART	LVTTL-3.3V interface level，DEBUGDebug port, can be connected with the USB to serial circuit, terminal definition, such as floor icon marking.
8.	UART0	RS232Interface level，Port definitions as remarked
9.	UART1	RS232Interface level，Port definitions as remarked
10.	UART2	RS232Interface level，Port definitions as remarked
11.	UART3	TTLInterface level，Port definitions as remarked
12.	UART4	TTLInterface level，Port definitions as remarked
13.	UART5	TTLInterface level，Port definitions as remarked
14.	3.3V Interface	3.3VInterface level，Port definitions as remarked
15.	5.0V Interface	5.0VInterface level，Port definitions as remarked
16.	ARM926-CBCore plate	Terminal definition, please see table

1. ARM926-CBHardware characteristic：

1-1 ARM926-CBPpackage chart:



1-2 ARM926-CB Pin Definition:

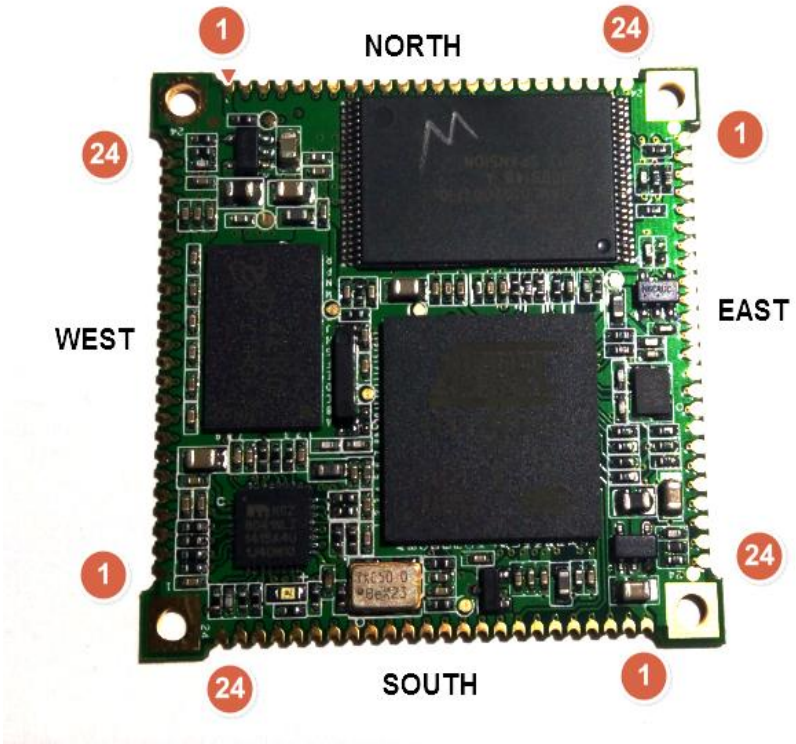


Figure1ARM926-CBSequence definition

Pin [1]#	IC[2] Pin	Default	Alt	GPIO[3]	Note
N1		3V3			Power In Vcc

[1]N means North， E means EAST ,S means South， W means WEST
[2]Corresponding to AT91SAM9G25-CUPins on chip
[3] Corresponding to the ID code of Linux

N2	PC0	SDA1	GPIO	64	I2C bus 1 Data or GPIO
N3	PC1	SCL1	GPIO	65	I2C bus 1 Clock or GPIO
N4	PC2	GPIO		66	
N5	PC3	GPIO		67	
N6	PC4	GPIO		68	
N7	PC5	GPIO		69	
N8	PC6	GPIO		70	
N9	PC7	GPIO		71	
N10	PC8	UTXD0	GPIO	72	UART0: **/dev/ttyS5** TXD
N11	PC9	URXD0	GPIO	73	UART0: **/dev/ttyS5** RXD
N12	PC10	GPIO		74	
N13	PC11	GPIO		75	
N14	PC12	GPIO		76	
N15	PC13	GPIO		77	
N16	PC14	GPIO		78	
N17	PC15	GPIO		79	
N18	PC16	UTXD1	GPIO	80	UART1: **/dev/ttyS6** TXD
N19	PC17	URXD1	GPIO	81	UART1: **/dev/ttyS6** TXD
N20	PC18	GPIO	PWM0	82	GPIO or Pulse Wave Modulation Out 0
N21	PC19	GPIO	PWM1	83	GPIO or Pulse Wave Modulation Out 1
N22	PC20	GPIO	PWM2	84	GPIO or Pulse Wave Modulation Out 2
N23	PC21	GPIO	PWM3	85	GPIO or Pulse Wave Modulation Out 3
N24		GND			Power In GND
E1		GND			Power In Vcc
E2	PC22	TXD3	GPIO	86	USART3: **/dev/ttyS4 TXD**
E3	PC23	RXD3	GPIO	87	USART3: **/dev/ttyS4 RXD**
E4	PC24	RTS3	GPIO	88	USART3: **/dev/ttyS4 RTS**
E5	PC25	CTS3	GPIO	89	USART3: **/dev/ttyS4 CTS**
E6	PC26	GPIO		90	
E7	PC27	RTS1	GPIO	91	USART1: **/dev/ttyS2 RTS**
E8	PC28	CTS1	GPIO	92	USART1: **/dev/ttyS2 CTS**
E9	PC29	GPIO		93	
E10	PC30	GPIO		94	
E11	PC31	GPIO		95	
E12	USBCP	D+			USB 2.0 Host full-speed port C
E13	USBCN	D-			USB 2.0 Host full-speed port C

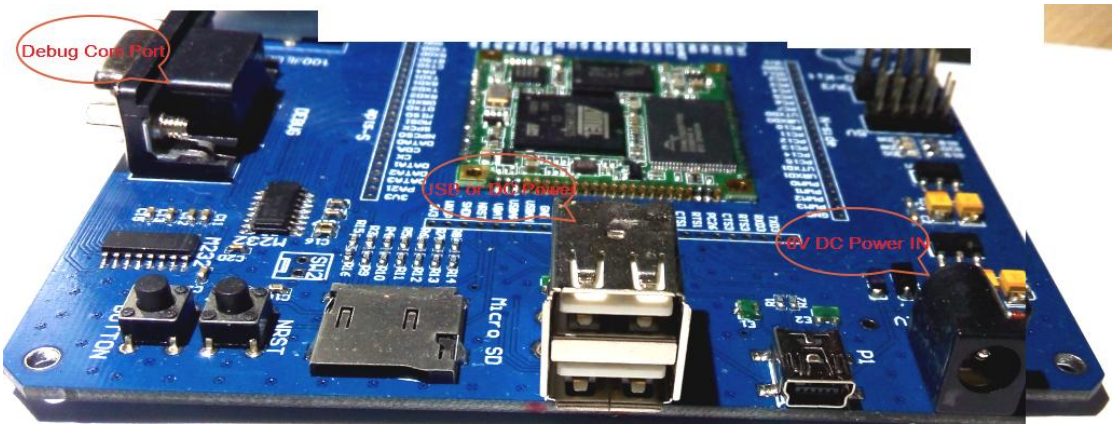
E14		GND			Power In GND
E15	USBBN	D-			USB 2.0 Host hi-speed port B
E16	USBBP	D+			USB 2.0 Host hi-speed port B
E17		GND			Power In GND
E18	USBAN	D-			USB 2.0 Host/Device hi-speed port A
E19	USBAP	D+			USB 2.0 Host/Device hi-speed port A
E20	VBAT				RTC battery backup +3 volt DC input
E21	NRST				Reset input signal (active low)
E22	SHDN				Shutdown output signal (active low)
E23	WKUP				Wake-up input signal (active low)
E24		3V3			Power In Vcc
S1		3V3			Power In Vcc
S2	PA21	1W	GPIO	21	Bit banging 1-wire bus or GPIO
S3	PA20	DA3		20	microSD Card memory
S4	PA19	DA2		19	microSD Card memory
S5	PA18	DA1		18	microSD Card memory
S6	PA17	CK		17	microSD Card memory
S7	PA16	CDA		16	microSD Card memory
S8	PA15	DA0		15	microSD Card memory
S9	PA14	NPCS0	GPIO	14	SPI bus 0 chip select 0 or GPIO
S10	PA13	SPCK	GPIO	13	SPI bus 0 clock or GPIO
S11	PA12	MOSI	GPIO	12	SPI bus 0 Master Output or GPIO
S12	PA11	MISO	GPIO	11	SPI bus 0 Master Input or GPIO
S13	PA10	DTXD		10	Debug serial port
S14	PA9	DRXD		9	Debug serial port
S15	PA8	RXD2	GPIO	8	USART2: **/dev/ttyS3 RXD** or GPIO
S16	PA7	TXD2	GPIO	7	USART2: **/dev/ttyS3 TXD** or GPIO
S17	PA6	RXD1	GPIO	6	USART2: **/dev/ttyS2 RXD** or GPIO
S18	PA5	TXD1	GPIO	5	USART2: **/dev/ttyS2 TXD** or GPIO
S19	PA4	GPIO		4	
S20	PA3	CTS0	GPIO	3	USART1: **/dev/ttyS1 CTS** or GPIO
S21	PA2	RTS0	GPIO	2	USART1: **/dev/ttyS1 RTS** or GPIO
S22	PA1	RXD0	GPIO	1	USART1: **/dev/ttyS1 RXD** or GPIO
S23	PA0	TXD0	GPIO	0	USART1: **/dev/ttyS1 TXD** or GPIO
S24		GND			Power In GND
W1		GND			Power In GND
W2		ETHRXP			Eth RX+

W3		ETHRXN			Eth RX-
W4		ETHTXP			Eth TX+
W5		ETHTXN			Eth TX-
W6		ETH3V3			Eth 3V3
W7		ETHLED1			Eth Yellow led (traffic)
W8		ETHLED2			Eth Green led (link)
W9	PA22	GPIO		22	
W10	PA23	GPIO		23	
W11	PA24	GPIO		24	
W12	PA25	GPIO		25	
W13	PA26	GPIO		26	
W14	PA27	GPIO		27	
W15	PA28	GPIO		28	
W16	PA29	GPIO		29	
W17	PA30	SDA0	GPIO	30	I2C bus 0 Data or GPIO
W18	PA31	SCL0	GPIO	31	I2C bus 0 Clock or GPIO
W19	ADVREF				A/D converter voltage reference In (max 3.3 volt)
W20	PB11	AD0	GPIO	43	A/D converter Input 0 or GPIO
W21	PB12	AD1	GPIO	44	A/D converter Input 1 or GPIO
W22	PB13	AD2	GPIO	45	A/D converter Input 2 or GPIO
W23	PB14	AD3	GPIO	46	A/D converter Input 3 or GPIO
W24		3V3			Power In Vcc

1-3 ARM926-CBPower source

Items	Min Typical Max	
DC	3.3V	
DC Power consumption	TBD	

1-4 Quick start



1) Connect the serial port to Com Port Debug if you need to monitor the system's boot information, if not, you can ignore the steps, serial port is set as: 115200, 8N1, no flow control；
2) Power supply, you can choose the following 2 ways, USB power supply is recommended

- USB power supply
- DC Adapter power supply

When the power supply is successful, you can see the board power indicator light；

3) If the Led starts to flash after about 20s, the system starts successfully, if operating step 1, you can see the system to start the print output information.

2. Software Introduction:

2-1. Cross compiling environment（Install Cross-Compile tools）

This article illustrates how to install on a Ubuntu Linux PC the complete toolchain to cross compile the Linux Kernel, the Linux device drivers, the Linux applications, C and C++ applications and the boot loader AT91Bootstrap and uboot.

I tested on a PC with Linux Ubuntu Version 12.04. If you are running Ubuntu 10.04 read this post.

Install the gcc tools in your ubuntu system, if you have install the gcc on your PC, don't need to do it again.

```
sudo apt-get install build-essential

sudo apt-get install libncurses5-dev

sudo apt-get install mtd-utils

sudo apt-get install u-boot-tools
```

Copy the arm-2011.09-70-arm-none-linux-gnueabi-i686-pc-linux-gnu.tar.bz2 and mkubifsimage to /tmp folder and uncompress it to the /opt/codesourcery/, the tools was in the CD://linux/tools/ folder.

```
# sudomkdir /opt/codesourcery/

# sudo tar -jxvf /tmp/arm-2011.09-70-arm-none-linux-gnueabi-i686-pc-linux-gnu.tar.bz2 -C /opt/codesourcery/

# sudocp /tmp/mkubifsimage /bin/

# sudochmod 777 /bin/mkubifsimage
```

Add Path in your environment file

Modify your ~/.bashrc file to add a new path with editor (gedit or vi)

```
# gedit ~/.bashrc
```

add the follow into the .bashrc file

```
export PATH=/opt/codesourcery/arm-2011.09/bin:$PATH
```

To apply this change, login again or restart the .bashrc

```
# source ~/.bashrc
```

Check it:

```
# arm-none-linux-gnueabi-gcc -v

Using built-in specs.

Target: arm-none-linux-gnueabi

…

gcc version 4.6.1 (SourceryCodeBench Lite 2011.09-70)
```

Now you are ready to cross-compile on your PC all the source availablefor the ARM926-CB Boards based on ATMEL AT91SAM CPUs.

2-2. Application development examples

To creat the hello.cfile and add the following example to the hello.c file.

```
#include "stdio.h"


int main(void) {

printf("Hello world !\n");

return 0;

}
```

Compile it typing:

```
# arm-none-linux-gnueabi-gcchello.c -o hello
```

Then copy the "hello" to the ARM926-CB board by USB Disk or TF card, lauch it typing:

```
# cp /media/mmcblk0p1/hello /home

# chmod /home/hello

# /home/hello

Hello world !
```

2-3. BSPsystem construction method

- **Compiling AT91Bootstrap for ARM926-CB**

AT91Bootstrap is the 2nd level bootloader for Atmel AT91 SoC providing a set of algorithms to manage the hardware initialization such as clock speed configuration, PIO settings, DDR2 DRAM initialization. After that it downloads your main application, usually the Linux Kernel image, from the first microSD partition to DDR2 RAM main memory and runs it from there.

How to compile AT91Bootstrap:

Install the cross compiler and the toolchain required on your Linux Ubuntu PC following this article (Tested on Ubuntu 12.04):

- **Install the ARM9 cross toolchain to compile the Linux Kernel and AT91Bootstrap**

Copy the ARM926-CB at91bootstrap-master.tar.bz2 from the CD://linux/code/ to the PC /tmp folder, then do the follow command to uncompress it:

```
~$ cd ~

~$ tar xvf /tmp/at91bootstrap-master.tar.bz2 -C .
```

This is a fork of the original Atmel version available on GitHub.

Move inside the at91bootstrap directory and launch Make:

```
~$ cd at91bootstrap-master

~/at91bootstrap$ makemrproper

~/at91bootstrap$ make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi-
at91sam9x5eknf_uboot_defconfig
```

When the configuration in ready save and launch the compilation by typing:

```
~/at91boostrap$ make -j4 ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi-
```

Copy the binary file to USB Disk, and then you can insert it to the board to update the bootstrap file(refer to the Update the system image by USB Disk)

```
~/at91bootstrap$cp binaries/at91sam9x5ek-nandflashboot-uboot-3.5.3.bin /media/usbhd-sda1/
```

/media/usbhd-sda1/ *is the USB Disk.*

- **Compiling U-boot for ARM926-CB**

U-Boot is the booloader for the ARM926-CB board.

How to compile u_boot:

Install the cross compiler and the toolchain required on your Linux Ubuntu PC following this article (Tested on Ubuntu 12.04):

- Install the ARM9 cross toolchain to compile the Linux Kernel and AT91Bootstrap

Copy the ARM926-CB **u-boot-at91-master.tar.bz2** from the CD://linux/code/ to the PC /tmp folder, then do the follow command to uncompress it:

```
~$ cd ~

~$ tar xvf /tmp/u-boot-at91-master.tar.bz2 -C .
```

This is a fork of the original Atmel version available on GitHub.

Move inside the at91bootstrap directory and launch Make:

```
~$ cd u-boot-at91-master/

~/u-boot-at91-master$ make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi- distclean

~/u-boot-at91-master$ make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi-
at91sam9x5ek_nandflash_config
```

If you use the 256MB DDR2 RAM version, run the follow command instead the above command:

```
~/u-boot-at91-master$ make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi-
at91sam9x5ek_nandflash_256M_config
```

When the configuration in ready save and launch the compilation by typing:

```
~/u-boot-at91-master$ make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi- -j4
at91sam9x5ek_nandflash
```

Copy the binary file u-boot.bin to USB Disk, and then you can insert it to the board to update the bootstrap file(refer to the Update the system image by USB Disk)

```
~/u-boot-at91-master$cp u-boot.bin /media/usbhd-sda1/
```

/media/usbhd-sda1/ is the USB Disk

- **Compiling the Linux Kernel 3.6.9 for the ARM926-CB**

This article illustrates how to generate a bootable Linux Kernel 3.6.9 image for ARM926-CB

Hardware requirements:

- an Ubuntu Linux PC (we used an Ubuntu 12.04)
- an ARM926-CB with base board ARM926-CB-CON
- a Debug Serial Port Interface

Installing the toolchain:

Install the cross compiler and the toolchain required on your PC following this article:

- Install the ARM9 cross toolchain to compile the Linux Kernel and AT91Bootstrap

Prepare the Kernel sources and build it:

Copy the ARM926-CB linux-at91.tar.bz2 from the CD to the tmp folder, then do the follow command to uncompress it:

```
~$ cd ~

~$ tar xvf /tmp/linux-at91.tar.bz2 -C .
```

then move in the linux-at91 folder:

```
~$ cd linux-at91

~/linux-at91$ make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi- distclean

~/linux-at91$ make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi- at91_dt_defconfig
```

*A file called **.config** will be generated inside the Linux source root directory. All the changes you will make using **make menuconfig** will be saved on this file so if you want to create you own default*

*configuration simple copy the new **.config** file in **arch/arm/configs** with a name like at91_dt_defconfig.*

the device tree source file (.dts) for your board is in arch/arm/boot/dts/at91sam9g25ek.dts.

then compile it to generate a device tree blob file (.dtb) by typing:

```
~/linux-at91$ make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi- dtbs
```

This file will be used by the Linux Kernel at startup time to configure the initial state of your hardware.

Customize the default Linux Kernel configuration:

If you need customize the Kernel configuration by using **menuconfig** type:

```
~/linux-at91$ make ARCH=arm menuconfig
```

Running the Kernel compilation

To compile the Linux Kernel type:

```
~/linux-at91$ make -j4 ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi- uImage

...

Kernel: arch/arm/boot/uImage is ready
```

Copying the binary files generated in the USB Disk

Insert a USB Disk and copy the bootable Linux kernel file **uImage** file and the hardware description file **at91sam9g25ek.dtb** into the USB Disk:

```
~/linux-at91$cp arch/arm/boot/dts/at91sam9g25ek.dtb /media/usbdisk/

~/linux-at91$ cp arch/arm/boot/uImage /media/usbdisk/
```

The /media/usbdisk is the USB Disk folder in ubuntu

- **Build an Buildroot root filesystem**

We use the Buildroot as the root file system.

How to Build the Buildroot:

Install the cross compiler and the toolchain required on your Linux Ubuntu PC following this article (Tested on Ubuntu 12.04):

- Install the cross toolchain to build system image
- Download the root file system source code

Copy the rootfs.tar.bz2 to the PC /tmp folder, then do the follow command to uncompress it:

```
~$ cd ~

~$ sudo tar xvf /tmp/rootfs.tar.bz2 -C .
```

This is a fork of the original Atmel version available on GitHub.

Do the filesystem with the follow command:

```
~$ sudomkubifsimagerootfs/ ubifs.img
```

Then we will get the ubifs.img file, you can copy it to the USB Disk/Micro TF card for update the board system image.

The atmel provide the method to Build the Buildrootfilesystem, if you want to do it by yourself, refer the atmel linux4sam website.

2-4. System upgrade method

- **Update the system image by USB Disk**

Prepare

- Connect the Serial port to the PC(refer to the chapter "Debug Serial Port Interface")
- Confirm the ARM926-CB can boot the U-boot(if it can't boot the U-boot, please Recovery the pre-install System by SD card)

Step-by-step guide

1. Prepare the USB Disk, Copy your system image file(only copy the file you want to update) to the USB Disk:

bootstarp file: **at91sam9x5ek-nandflashboot-uboot-3.5.3.bin**

U-boot file: **u-boot.bin**

DTS file: **at91sam9g25ek.dtb**

Linux kernel: **uImage**

File system: **ubifs.img**

please confirm the file name is the same that you copy it to the USB Disk, only copy the image that your want to update, don't need copy all the file.

If you use the 256MB DDR2 RAM version, you should rename the ***at91sam9x5ek-nandflashboot-uboot-256M-3.5.3.bin to at91sam9x5ek-nandflashboot-uboot-3.5.3.bin, or rename the u-boot-256M.bin to u-boot.bin.***

2. Insert the USB Disk to the ARM926-CB-CON board.

3. Power on the board, then the board will update the system auto, when the Terminal print the follow info, it means update system success.

```
### Update OK ### Please RESET the board ###
```

4. Remove the USB disk, and power on the board again.

- **Recovery the System by SD card**

Prepare

- Connect the Serial port to the PC(refer to the chapter "Debug Serial Port Interface")

Step-by-step guide

1. Prepare the TF card, Copy the follow recovery system image file to the TF card:

The follow recovery system image file position: ***CD:\\Linux\image\SDrecovery\128MB_DDR2_RAM***

bootstarp file: **boot.bin**

U-boot file: **u-boot.bin**

Recovery system file: **usbupgrad_sam9g25ek.dt**

2. Insert the TF card to the ARM926-CB-CON board.

3. Power on the board, then the board will enter the system recovery mode, and recovery the system auto. when the Terminal print the follow info, it means update system success.

```
### Update OK ### Please RESET the board ###
```

4. Remove the TF card, and power on the board again.

- **System partition table**

0x00000-0x40000 : "bootstrap"
0x40000-0xc0000 : "uboot"
0xc0000-0x100000 : "env"

0x100000-0x140000 : "env_redundant"
0x140000-0x180000 : "spare"
0x180000-0x200000 : "dtb"
0x200000-0x800000 : "kernel"
0x800000-0x10000000 : "rootfs"

3. System configuration

3-1. Set the system date

Your ARM926-CB board has two clocks:

- One called **System Clock** that is a software clock maintained by the Linux Operating System
- Another one called Real Time Clock (RTC) and implemented in hardware inside the Atmel CPU. This clock is powered by a Lithium battery to mantain the clock alive when the main power supply is off.

Setting the System Clock

date is the Linux command to manage the systems clock.

To read the currently System Clock type:

```
~# date
Fri Oct  8 17:44:42 CEST 2010
```

To set it type:

```
~# date -s 201311231517
Sat Nov 23 15:17:00 UTC 2013
```

This time is active until the main power supply board is on. When off the system clock is lost.

Setting the Real Time Clock

To read the Hardware Clock type:

```
~# hwclock -r
Fri Oct  8 17:46:43 2010  -0.004115 seconds
```

To set the Hardware Clock using the System Clock current time type:

```
~# hwclock -w
```

Now check it typing:

```
~# date
Fri Oct  8 18:49:02 CEST 2010
~# hwclock -r
Fri Oct  8 18:49:10 2010  -0.004076 seconds
```

3-2. Change the hostname

When you get access to the ARM926-CB board command line the prompt is something like this:

```
ARM926-CB:~#
```

Where **ARM926-CB** is the current hostname.

If more ARM926-CB boards or other Linux systems are reacheable on your LAN it could be important to have different hostnames tfor each board to avoid for example to give commands to the wrong system.

The hostname is saved in **/etc/hostname**.

Change it for example with **myboard** by typing:

```
~# echo "myboard"> /etc/hostname
```

3-3. SD Card opration examples

Enter the SD Card

```
~# cd /sys/class/mmc_host/mmc?/mmc?:*
/sys/class/mmc_host/mmc0/mmc0:aaaa# cat serial
0x015c5340
```

A lot of other info are available:

```
/sys/class/mmc_host/mmc0/mmc0:aaaa# ls -al
total 0
drwxr-xr-x 4 root root      0 Jun 27 15:37 .
drwxr-xr-x 4 root root      0 Jun 27 15:37 ..
drwxr-xr-x 3 root root      0 Jun 27 15:37 block
```

```
-r--r--r-- 1 root root 4096 Jun 28 07:18 cid
-r--r--r-- 1 root root 4096 Jun 28 07:28 csd
-r--r--r-- 1 root root 4096 Jun 28 07:28 date
lrwxrwxrwx 1 root root       0 Jun 27 15:37 driver -> ../../../../../../bus/mmc/drk
-r--r--r-- 1 root root 4096 Jun 28 07:28 erase_size
-r--r--r-- 1 root root 4096 Jun 28 07:21 fwrev
-r--r--r-- 1 root root 4096 Jun 28 07:21 hwrev
-r--r--r-- 1 root root 4096 Jun 28 07:21 manfid
-r--r--r-- 1 root root 4096 Jun 27 15:37 name
-r--r--r-- 1 root root 4096 Jun 28 07:18 oemid
drwxr-xr-x 2 root root       0 Jun 28 07:28 power
-r--r--r-- 1 root root 4096 Jun 28 07:28 preferred_erase_size
-r--r--r-- 1 root root 4096 Jun 28 07:28 scr
-r--r--r-- 1 root root 4096 Jun 27 15:37 serial
lrwxrwxrwx 1 root root       0 Jun 27 15:37 subsystem -> ../../../../../../bus/mmc
-r--r--r-- 1 root root 4096 Jun 28 07:28 type
-rw-r--r-- 1 root root 4096 Jun 27 15:37 uevent
```

Mount sd card

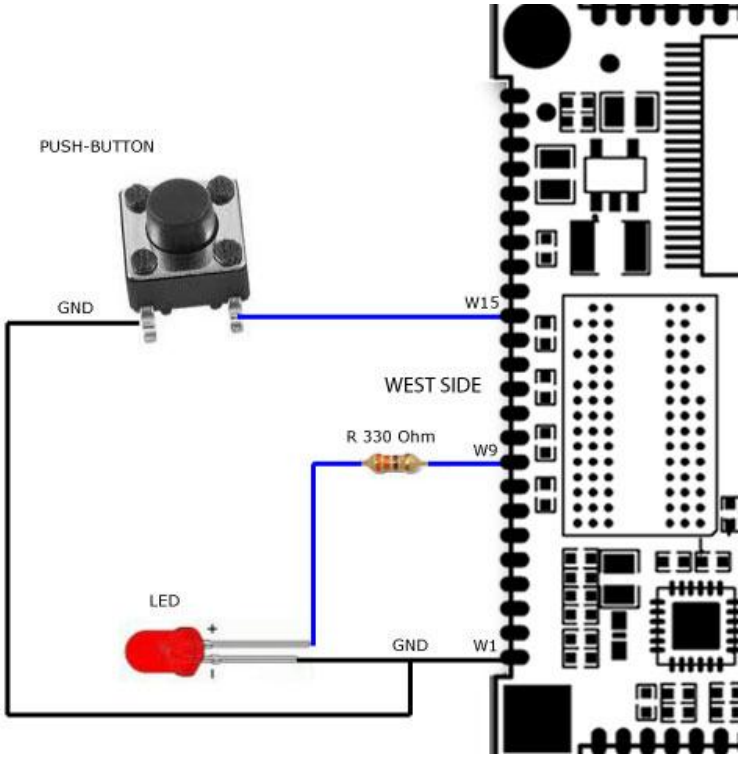
```
mount -t vfat /dev/mmcblk0p1 /mnt/sd
```

Umountsd card

```
umount /mnt/sd
```

GPIO lines

Using this basic example of wiring:



ARM926-CB gpio Output expample

```
# cd /sys/class/gpio/

echo 22 >export

# ls

export          gpiochip32    gpiochip96    pioA28

gpiochip0       gpiochip64    pioA22unexport

# echo "out">pioA22/direction

# echo 1 > pioA22/value

# echo 0 > pioA22/value
```

ARM926-CBgpio Input expample

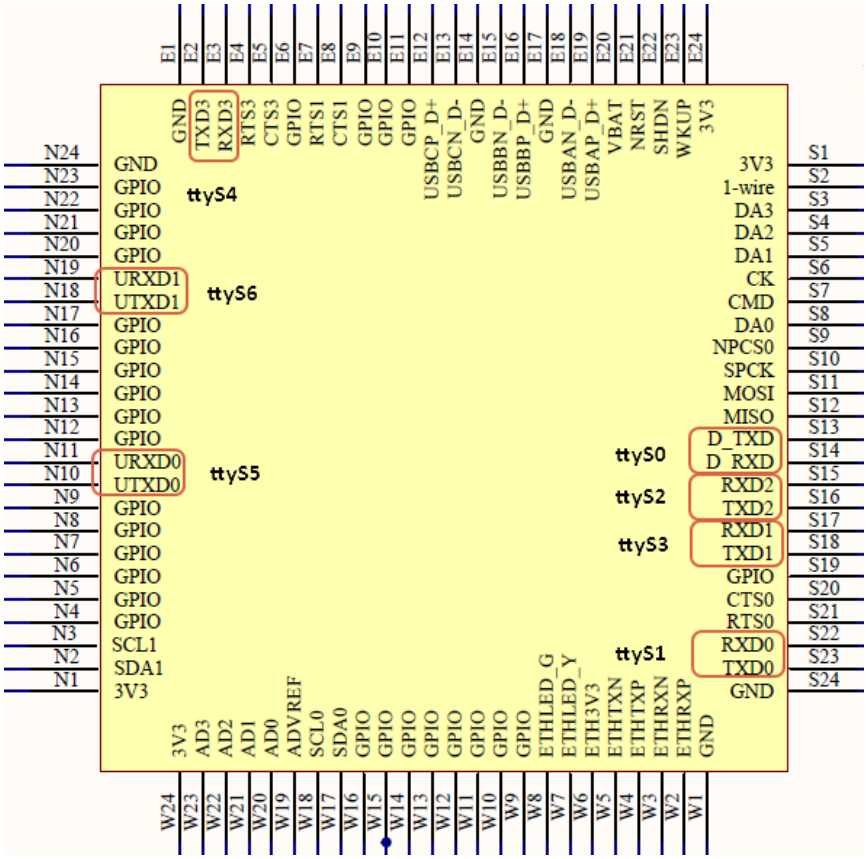
it is possible to define the **W15** line as an iutput and get the IO status with the following command:

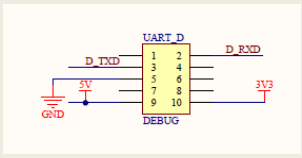
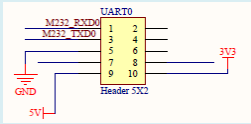
```
# cd /sys/class/gpio/

# echo "28">export
```

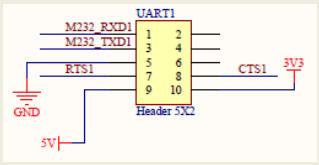
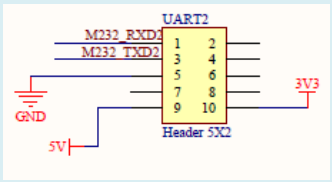
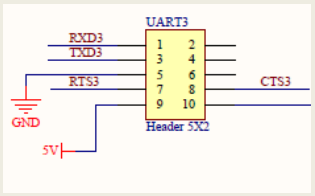
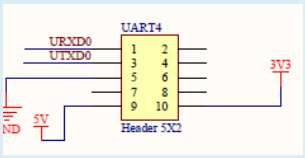
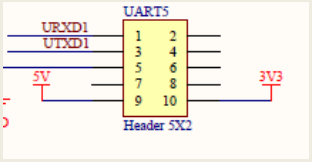
```
# echo "in">pioA28/direction

# cat pioA28/value

# echo "28">unexport
```

SerialInterface

Hardware Pinout



Serial Port	Linux Inode	Hardware Pinout
Debug serial	/dev/ttyS0	
Usart0	/dev/ttyS1	

Usart1	/dev/ttyS2	
Usart2	/dev/ttyS3	
Usart3	/dev/ttyS4	
Uart0	/dev/ttyS5	
Uart1	/dev/ttyS6	

Serial C Code routine

```
/* rs232_send.c*/

#include <stdio.h> /*Standard input and output definitions*/

#include <stdlib.h> /*Standard function library definition*/

#include <unistd.h> /*Unix Standard function library definition*/

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h> /*File control definition*/

#include <termios.h> /*PPSIX Terminal control definition*/

#include <errno.h> /*Error number definition*/

#define BAUDRATE B115200//38400

char *MODEMDEVICE[] = {

        "/dev/ttyS1",

        "/dev/ttyS1",
```

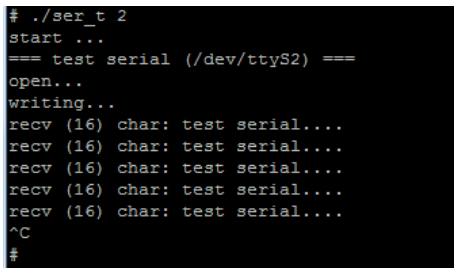
```
        "/dev/ttyS2",
        "/dev/ttyS3",
        "/dev/ttyS4",
        "/dev/ttyS5",
        "/dev/ttyS6"
};
int main(intargc, char* argv[])
{
        intfd,c=0,res;
        structtermiosoldtio,newtio;
        intch;
        static char s1[20];
        charbuf[] = {"test serial...."};

        printf("start ...\n");
        if(argc ==1)
        {
                printf("use default serial: tty1\r\n");
                ch = 1;
        }

        else
                ch = atoi(argv[1]);

        printf("=== test serial (%s) ===\r\n",MODEMDEVICE[ch]);
        /*打开 PC 的 COM1 口*/
        fd = open(MODEMDEVICE[ch],O_RDWR|O_NOCTTY);
        if (fd< 0)
        {
```

```
                perror(MODEMDEVICE[ch]);
                exit(1);
        }
        printf("open...\n");
        /*Put the old communication parameters intooldtio structure*/
        tcgetattr(fd,&oldtio);
        /*Initialize the newnewtio */
        bzero(&newtio,sizeof(newtio));
        /*8N1*/
        newtio.c_cflag = BAUDRATE|CS8|CLOCAL|CREAD;
        newtio.c_iflag = IGNPAR;
        newtio.c_oflag = 0;

        /*Normal mode*/
//      newtio.c_lflag = ICANON;

        /*Abnormal mode*/
        newtio.c_lflag = 0;
        newtio.c_cc[VTIME] = 0;
        newtio.c_cc[VMIN] = sizeof(buf);

        tcflush(fd,TCIFLUSH);
        /The new temios is the communication port parameter*/
        tcsetattr(fd,TCSANOW,&newtio);
        printf("writing...\n");

        while(1)
        {
                res = write(fd,buf,sizeof(buf));
```

```
                bzero(buf, sizeof(buf));
                res = read(fd,buf,sizeof(buf));
                printf("recv (%d) char: %s \r\n",res, buf);
                if(buf[0]==9) break;
                sleep(1);
        }

        printf("close...\n");
        close(fd);
        /*Restore the old parameters*/
        tcsetattr(fd,TCSANOW,&oldtio);
        return 0;
}
```

Test methods：

The external return, self transmitting and self receiving, make the received data short cut with the sent data.



A/D converter lines

The A/D converter input lines are available on the following pins:

Signal	Description	ARM926-CB
3.3V	3.3 volt power line	W24
AVDD	Clean 3.24V out for A/D circuitry	
VREF	A/D voltage reference input	W19
AGND	Analog GND	
AD0	Analog input 0	W20

Signal	Description	ARM926-CB
AD1	Analog input 1	W21
AD2	Analog input 2	W22
AD3	Analog input 3	W23
GND	Digital GND	W1

More Details, Reading the A/D converter lines:

http://www.at91.com/linux4sam/bin/view/Linux4SAM/IioAdcDriver

4. Technical Support

For technical support, please send e-mail to: technical@wireless-tag.com

Disclaimer: We reserve the final interpretation and modification rights to update the product manuals without notice at any time!