Beaglebone mikroBUS Cape

Manufacture Part number: 3651

Description

The mikroBUS Cape is an expansion board for the BeagleBone that allows users to connect up to 4 mikroElektronika Click boards to the BeagleBone without any wiring.

Software Support & Compatibility

No software support is required by this board. All revisions are compatible with BeagleBone and BeagleBone Black.

Examples are compatible with BeagleBone and BeagleBone Black.

NOTE: At this point in time the Debian kernel is experiencing issues with applying device tree overlays, the examples may not work.

Getting Started

BeagleBone (White)

To allow BB to identify appropriate pin headers configuration, Click Cape have an onboard EEPROM programmed to make the use of the Cape “plug and play”, just insert the Cape and reboot the BB.

BeagleBone Black

From adoption of Linux kernel 3.x a new method to configure pin headers was introduced, at the boot time an in-kernel utility named “capemgr” dynamically load Device Tree definitions to configure an appropriate pin headers and bus initialization.

As in the case of the White, reading the information stored in the onboard EEPROM, the use of Click Cape is “plug and play” but needs that the user (just one time) put the *.dtbo file (downloadable form git or from our server) in /lib/firmware and reboot the system.

Before this setup it is strongly suggested to start with a fresh install of the OS, latest images are found here:Latest images It is also recommended to try a fresh install to make sure that the system has no user caused errors, or leftovers from previous projects.

Angström setup

```
$ git clone git://github.com/TIGAL/BBB-mikroBUS-Cape.git

$ cd BBB-mikroBUS-Cape

$ mkdir bbclickdtbo

$ tar -C /home/root/BBB-mikroBUS-Cape/bbclickdtbo/ -xvf BB_board_dtbo.tar

$ cd bbclickdtbo/

$ cp BB-MIKROBUS-01-00A1.dts /lib/firmware/

$ cp BB-MIKROBUS-01-00A1.dtbo /lib/firmware/

$ reboot
```

Debian setup

NOTE: At this time due to the changes being made in the kernel, the Device Tree Overlay is not functional. The Linux comminity is currently working on the Device Tree handling.

```
$ git clone git://github.com/TIGAL/BBB-mikroBUS-Cape.git

$ cd BBB-mikroBUS-Cape

$ mkdir bbclickdtbo

$ tar -C  /root/BBB-mikroBUS-Cape/bbclickdtbo/  -xvf BB_board_dtbo.tar

$ cd bbclickdtbo/

$ cp BB-MIKROBUS-01-00A1.dts /lib/firmware/
```

```
$ cp BB-MIKROBUS-01-00A1.dtbo /lib/firmware/

$ reboot
```

HDMI Disable

For the BeagleBone Black to function properly wit the cape the HDMI has to be disabled. To do this you have to replace the contents of the uEnv.txt file in the FAT partition of the BeagleBone Black:

```
$ mkdir /mnt/boot

$ mount /dev/mmcbk0p1 /mnt/boot

$ nano /mnt/boot/uEnv.txt
```

Replace the contents of uEnv.txt to this

```
optargs=quiet capemgr.disable_partno=BB-BONELT-HDMI,BB-BONELT-HDMIN
```

Then save and exit (*Control+X*, then *Y* then *Enter*), and reboot.

On debian this can be done by uncommenting the same line in the uEnv.txt

Recommended software

For terminal access through the debug port of the BeagleBone it is recommended to use PuTTY which can be downloaded here: PuTTY

For SD card imaging use Win32 Disk imager which is available from here: Win32 Disk Imager

Make your life easier

Make a .profile file in your root to make your life a lot easier when working with DT. Using the export command you can route the $PINS and $SLOTS, this allows you to easily see the pin statuses by using the 'cat $PINS' or 'cat $SLOTS' command. By adding this into a .profiles file these mappings are applied every time you start your board.

```
$ nano .profile
```

Then add the text below to it for Angström

```
export SLOTS=/sys/devices/bone_capemgr.8/slots

export PINS=/sys/kernel/debug/pinctrl/44e10800.pinmux/pins
```

Use this for Debian

```
export SLOTS=/sys/devices/bone_capemgr.9/slots

export PINS=/sys/kernel/debug/pinctrl/44e10800.pinmux/pins
```

Then save and exit (Control+X, then Y then Enter)

Use Github

All files and documents that are nessecary to get the Cape working is also availible on github. To use our github repositories do the following:

```
$ git clone git://github.com/TIGAL/BBB-mikroBUS-Cape.git
```

RTC2 Click

As the RTC2 Click is connected via I²C it can be connected to any port on the cape. With the default dtbo file in place it will certainly work without any programming. All you have to do is to set the date and time first

```
$ date -s 2014.07.24-15:00
```

Then write it to the RTC2 by

```
$ hwclock -w
```

As long as you have a good battery the correct time should be kept by the device after recovering from a power loss or a restart.

Specifications

Signal Usage

Click Host 1

Click Pin	BBB Pin	Signal	Mode	$PINS	ADDR/ OFFSET	GPIO NO.
AN	P9.38 (default) P8.26 (optional)	AIN3 (default) GPIO1_29 (optional)	- 7	-	-	-
RST	P9.12	GPIO1_28	7	30	0x878/078	60
CS	P9.28	SPI1_CS0	3	103	0x99c/19c	113
SCK	P9.31	SPI1_SCLK	3	100	0x990/190	110
MISO	P9.29	SPI1_D0	3	101	0x994/194	111
MOSI	P9.30	SPI1_D1	3	102	0x998/198	112
+3.3V	P9.3 - P9.4	VDD_3V3EXP	-	-	-	-
GND	P9.43 - P9.46	GND	-	-	-	-
PWM	P9.14	EHRPWM1A	6	18	0x848/048	50
INT	P9.15	GPIO1_16	7	16	0x840/040	48
RX	P9.21	UART2_TXD	1	85	0x954/154	3
TX	P9.22	UART2_RXD	1	84	0x950/150	2
SCL	P9.19	I2C2_SCL	3	95	0x97c/17c	13
SDA	P9.20	I2C2_SDA	3	94	0x978/178	12
+5V	NC	-	-	-	-	-
GND	P9.43 - P9.46	GND	-	-	-	-

Click Host 2

Click Pin	BBB Pin	Signal	Mode	$PINS	ADDR/ OFFSET	GPIO NO.
AN	P9.37 (default) P8.15 (optional)	AIN2 (default) GPIO1_15 (optional)	- 7	-	-	-
RST	P9.23	GPIO1_17	7	17	0x844/044	49
CS	P9.42	SPI1_CS1	2	89	0x964/1a0	7
SCK	P9.31	SPI1_SCLK	3	100	0x990/190	110
MISO	P9.29	SPI1_D0	3	101	0x994/194	111
MOSI	P9.30	SPI1_D1	3	102	0x998/198	112
+3.3V	P9.3 - P9.4	VDD_3V3EXP	-	-	-	-
GND	P9.43 - P9.46	GND	-	-	-	-
PWM	P9.16	EHRPWM1B	6	19	0x84c/04c	51
INT	P9.41	GPIO0_20	7	-	0x9a8/1a8	116
RX	NC	-	-	-	-	-
TX	NC	-	-	-	-	-
SCL	P9.19	I2C2_SCL	3	95	0x97c/17c	13
SDA	P9.20	I2C2_SDA	3	94	0x978/178	12
+5V	NC	-	-	-	-	-

Click Host 3

Click Pin	BBB Pin	Signal	Mode	$PINS	ADDR/ OFFSET	GPIO NO.
AN	P9.40 (default) P8.12 (optional)	AIN1 (default) GPIO1_12 (optional)	- 7	-	-	-
RST	P8.14	GPIO0_26	7	10	0x828/028	26
CS	NC	-	-	-	-	-
SCK	NC	-	-	-	-	-
MISO	NC	-	-	-	-	-
MOSI	NC	-	-	-	-	-
+3.3V	P9.3 - P9.4	VDD_3V3EXP	-	-	-	-

GND	P9.43 - P9.46	GND	-	-	-	-
PWM	P8.19	EHRPWM2A	4	8	0x820/020	22
INT	P8.18	GPIO2_1	7	35	0x88c/08c	65
RX	P9.24	UART1_TXD	0	97	0x984/184	15
TX	P9.26	UART1_RXD	0	96	0x980/189	14
SCL	P9.19	I2C2_SCL	3	95	0x97c/17c	13
SDA	P9.20	I2C2_SDA	3	94	0x978/178	12
+5V	NC	-	-	-	-	-
GND	P9.43 - P9.46	GND	-	-	-	-

Click Host 4

Click Pin	BBB Pin	Signal	Mode	$PINS	ADDR/ OFFSET	GPIO NO.
AN	P9.39 (default) P8.11 (optional)	AIN0 (default) GPIO1_13 (optional)	- 7	-	-	-
RST	P8.16	GPIO1_14	7	14	0x838/038	46
CS	NC	-	-	-	-	-
SCK	NC	-	-	-	-	-
MISO	NC	-	-	-	-	-
MOSI	NC	-	-	-	-	-
+3.3V	P9.3 - P9.4	VDD_3V3EXP	-	-	-	-
GND	P9.43 - P9.46	GND -	-	-	-	-
PWM	P8.13	EHRPWM2B	4	9	0x824/024	23
INT	P8.17	GPIO0_27	7	11	0x82c/02c	27
RX	P9.13	UART4_TXD	6	29	0x874/074	31
TX	P9.11	UART4_RXD	6	28	0x879/070	30
SCL	P9.19	I2C2_SCL	3	95	0x97c/17c	13
SDA	P9.20	I2C2_SDA	3	94	0x978/178	12
+5V	NC -	-	-	-	-	-
GND	P9.43 - P9.46	GND	-	-	-	-

Analog Pin note - Solder Jumpers

Each “AN” Click pin by default is connected to the analog pin of the BBB, if you want to use these “AN” pin as a digital pin you must to cut the wire in the jumper in the bottom of the Cape and solder it as needed.

+5 Volt Enable

To use clickboards that require 5 volts you must solder the 5V enable contact at the bottom. **WARNING:** Be careful, the ARM processor is only 3.3V tolerant, giving 5v to any input will result in a broken pin or even a broken ARM chip making the board nonfunctional.

EEPROM

The dip-switch “ADDR” in the bottom of the Cape allow user to modify I2C address of the EEPROM, this is useful in case of conflict or cascade of Capes. The two pin represent the last two bits of the address so the allowable address range is composed of four address, from 0x54 to 0x57 (default 0x57).

EEPROM Support	No
Board Name	BeagleBone mikrobus Cape
Version	Rev.1
Manufacturer	Tigal
Part Number	TIG-02413
Pins Used	0

Custom EEPROM

Data stored in the EEPROM is a simple structure formatted in JSON (http://en.wikipedia.org/wiki/JSON) which describes Cape and pin configuration needed, for a complete discussion see the BBB Reference Manual.

Here you can download the original JSON file we used to program Click Cape. Starting from there the user can create his custom EEPROM data structure.

With the command line routine “eeprom.js” we can parse JSON file and create the binary file who is write in the memory.

First create a symbolic link in the home to facilitate the work

```
$ cd

$ ln -s /usr/lib/node_modules/bonescript/eeprom.js eeprom.js
```

Write the binary file

```
$ node eeprom.js –w clickcape.json
```

Check the binary file

```
$ hexdump –C clickcape.eeprom
```

Write data in the EEPROM

```
$ cat clickcape.eeprom > /sys/bus/i2c/drivers/at24/1-0057/eeprom
```

Note: 1-0057 means I2C bus number 1 and address 0x57.

Reboot the system.

For more information see chapter 7 of the BeagleBone System Reference Manualhttp://beagleboard.org/static/beaglebone/latest/Docs/Hardware/BONE_SRM.pdf.

Custom DTS

For each different pin configuration a new DTS file needs to be configured, so if you would like to use a configuration that differs from the standard configuration then you need to build your own configuration. On github there is a documentation for what the standard pins and values are for and what default mode they are in.

At boot an in-kernel utility “capemgr” check the records “partNumber” and “version” in the EEPROM header and load DT according the file name from directory /lib/firmware.

If you want a custom pin configuration you must edit DT Source file (.dts), compile it and copy in the directory.

To use any of the build files from the git you will need to change its permissons using the “chmod 777 build” command.

```
$ dtc -O dtb -o BB-MIKROBUS-01-00A1.dtbo -b 0 -@ BB-MIKROBUS-01-00A1.dts

$ cp /lib/firmware/BB-MIKROBUS-01-00A1.dtbo /lib/firmware/BB-MIKROBUS-01-00A1.dtbo.bak

$ cp BB-MIKROBUS-01-00A1.dtbo /lib/firmware/BB-MIKROBUS-01-00A1.dtbo
```

Then reboot.

Check if DT is already loaded on Angström by typing

```
$ cat /sys/devices/bone_capemgr.8/slots
```

To check on Debian there is only one number difference

```
$ cat /sys/devices/bone_capemgr.9/slots
```

If you added the recommended settings in the .profile file then you can just type

$ cat $SLOTS

Here you can download the original DTS file. Starting from there the user can create his custom Device Tree definitions.

Note: you can load a DT definitions without reboot on Angström with

```
$ echo BB-MIKROBUS-01:00A1 > /sys/devices/bone_capemgr.8/slots
```

On debian

```
$ echo BB-MIKROBUS-01:00A1 > /sys/devices/bone_capemgr.9/slots
```

After this you can check again if the cape has been loaded.

```
$ cat $SLOTS
```

If you have left the PWM checks in the DT files then you should see the PWM pins being checked. If there is a module connected that is not using PWM the PPWM check will fail, but otherwise all tests should be successful.

Accessing pins directly

Once you have correctly built a dtbo file and successfully loaded the firmware with no errors, you can directly access the gpios. To do this you will have to find the PIN number and export it. Lets export the PWM pin that switches RL1 on the mikrobus relay click board.

```
$ cd /sys/class/gpio

$ echo 50 > export

$ cd gpio50
```

Within that folder the folders are what you can echo and set a value for, and see the status by cat-ing the value. For example lets set gpio50 direction to out so that the relay can function.

```
$ echo "out" > direction
```

To turn on the relay just set the value in a similar fashion

```
$ echo 1 > value
```

After you are done you need to unexport the GPIO

```
$ echo 50 > unexport
```