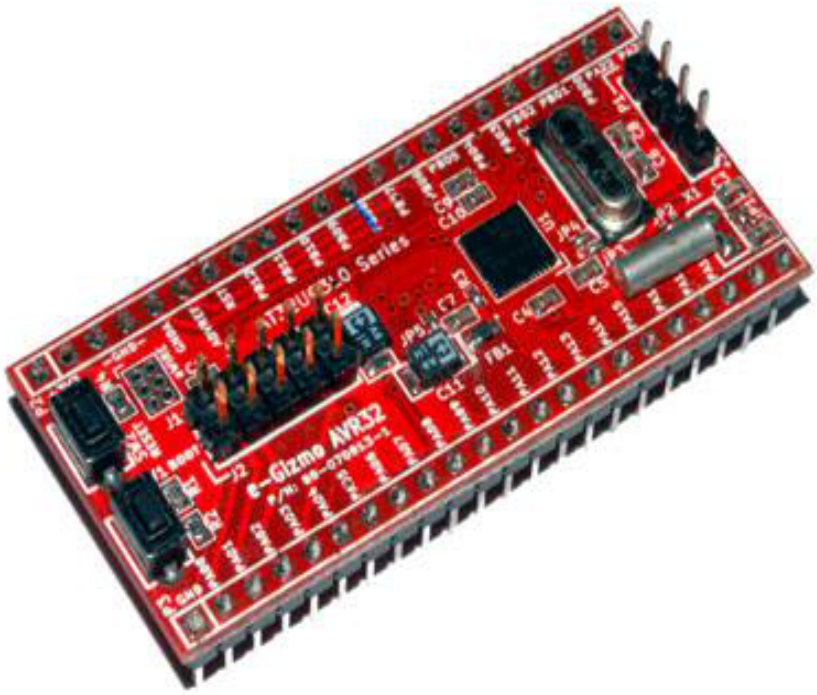
AVR32

Technical Manual Rev 1r0



AVR32 AT32UC3L0128/256

The AVR32 MCU board features a high-speed 32-bit MCU engine with 64DMIPS at 50MHz. With this performance, it can perform calculations and functions fast enough for scientific and high-end programming applications. With a built-in Memory protection unit, it ensures excellent performance for real-time applications accompanied by 8-channel 12-bit analog to digital converters.

It can also be used for multiple serial control used especially for wireless serial devices and alike. It also includes a JTAG port for JTAG interfacing. It also makes use of the PL2303 driver for uploading programs using a locally-made unique IDE.

Its ADCs can handle up to more or less 1.98V and each pin also features an internal pull-up resistor.

Features & Specifications:

- Flash memory/Microcontroller: AT32UC3L0128 (128K Flash) , AT32UC3L0256 (256K Flash)
- SRAM: 32kb
- High speed performance, 64DMIPS at 50MHz
- General functions: 2xTWI, 4xUART 2xSPI, 36xPWM, 8x12-bit ADC, Touch Module, frequency meter, etc.
- 40-pin 900 mils wide dual in line foot print.
- Clock frequency: 8 MHz up to 42MHz
- JTAG compatible
- +3.3V Supply input
- Default Baud rate: 115200bps

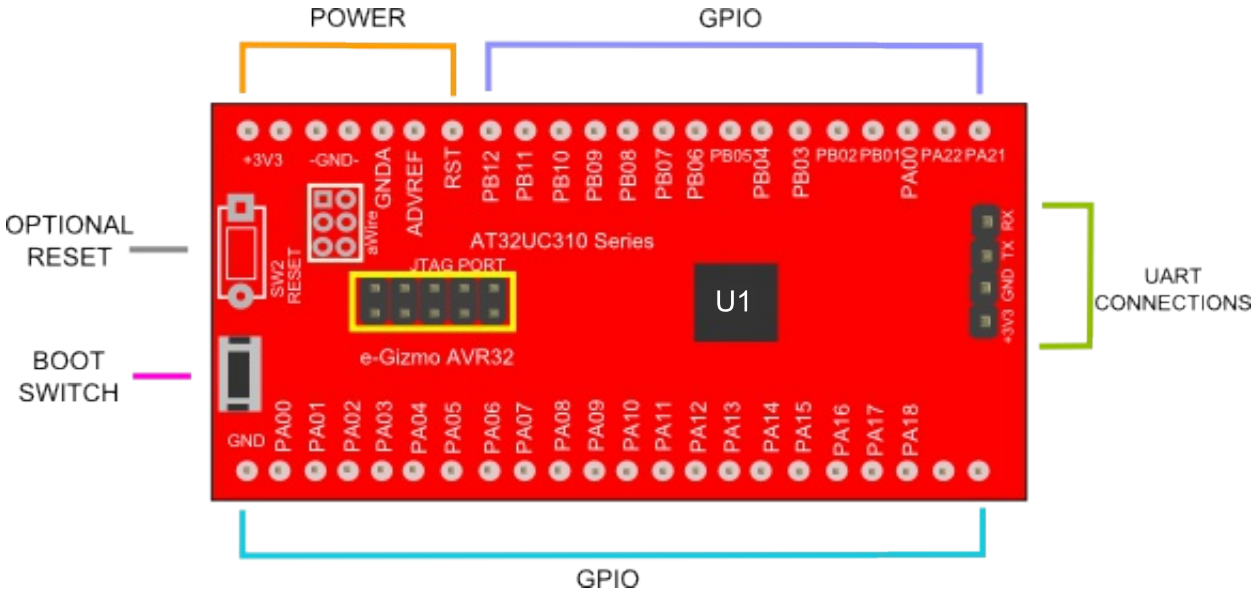


***Figure 1**. AVR32 AT32UC3L0128/256 Major Parts Presentation*

Component Specification:

U1 - AT32UC3L0128/256

Chip Comparison:

AT32UC3L0128
Flash 128kb, SRAM 32kb

AT32UC3L256
Flash 256kb, SRAM 32kb

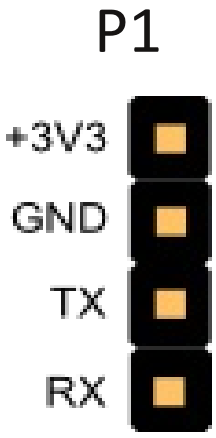


***Figure 2.** AVR32 P1 Pin Illustration*

***Table 1.** P1 Pin assignments*

Board Label	I.D.	Description
+3V3	+3.3V	+3.3V Power supply
GND	GND	Ground
TX	TX	Serial Transmit
RX	RX	Serial Receive

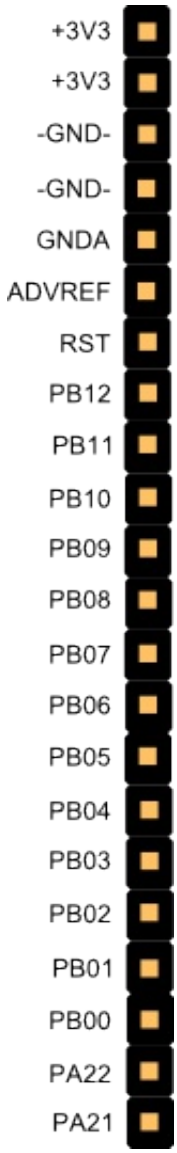


***Figure 3**. AVR32 P2 Pin Illustration*

Descriptions:

CLK(0-2) - Clock Input 0 to 2
AD(0-8) - Analog Input 0 to 8
TWI - Two wire interface
RXD(0-2) - UART Data Receive
TXD(0-2) - UART Data Transmit
MOSI - Master Out Slave In
MISO - Master In Slave Out

High drive I/O - high-drive output capable pins
Normal I/O - normal output pins

***Table 2.** P2 Pin assignments*

Board Label	I.D.	Description
+3V3	+3.3V	+3.3V Power supply
+3V3	+3.3V	+3.3V Power supply
GND	GND	Ground
GND	GND	Ground
GND	GND	Ground
GNDA	GNDA	Analog Ground
ADVREF	ADVREF	Analog Reference Voltage
PB12	44	Normal I/O (CLK0)
PB11	43	Normal I/O (CLK1)
PB10	42	Normal I/O (CLK2)
PB09	41	Normal I/O
PB08	40	Normal I/O (AD8)
PB07	39	Normal I/O (AD7)
PB06	38	Normal I/O (AD6)
PB05	37	Normal I/O (TWI)
PB04	36	Normal I/O(TWI)
PB03	35	Normal I/O
PB02	34	Normal I/O
PB01	33	High Drive I/O (RXD2)
PB00	32	Normal I/O (TXD2)
PA22	22	Normal I/O
PA21	21	Normal I/O (TWI RXD1)

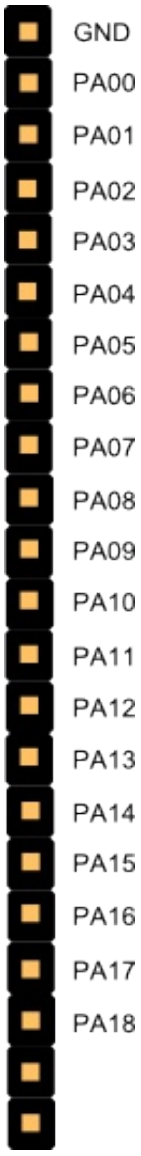


***Figure 4**. AVR32 P3 Pin Illustration*

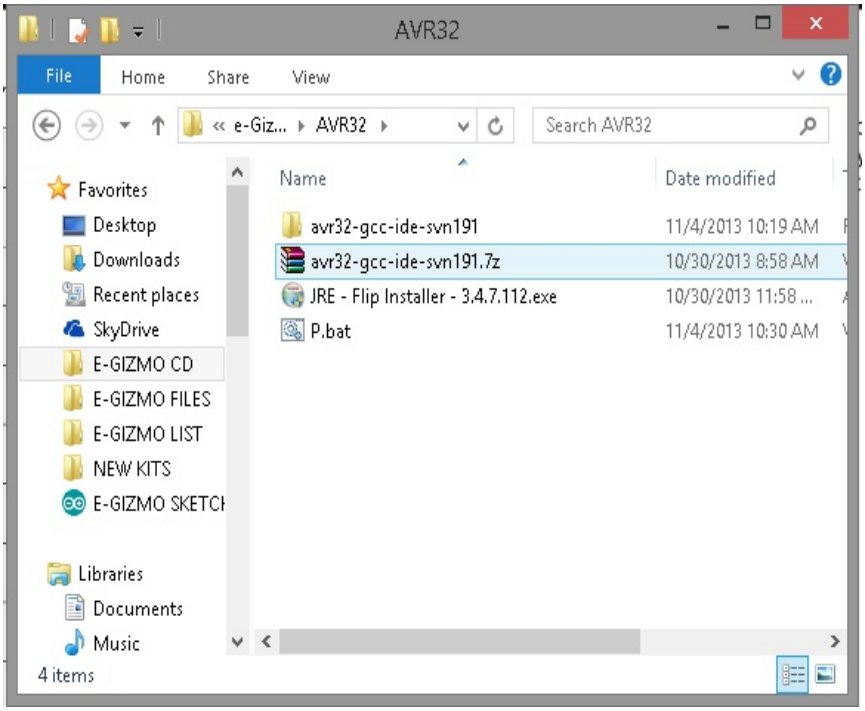
Descriptions:

CLK(0-2) - Clock Input 0 to 2
AD(0-8) - Analog Input 0 to 8
TWI - Two wire interface
RXD(0-2) - UART Data Receive
TXD(0-2) - UART Data Transmit
MOSI - Master Out Slave In
MISO - Master In Slave Out

High drive I/O - high-drive output capable pins
Normal I/O - normal output pins

***Table 3.** P3 Pin assignments*

Board Label	I.D.	Description
GND	GND	Ground
PA00	0	Normal I/O
PA01	1	Normal I/O
PA02	2	High Drive I/O
PA03	3	Normal I/O
PA04	4	Normal I/O(MISO)
PA05	5	Normal I/O (TWI/MOSI)
PA06	6	High Drive I/O
PA07	7	Normal I/O (TWI)
PA08	8	High Drive I/O (TXD0)
PA09	9	High Drive I/O (RXD0)
PA10	10	Normal I/O (TWD)
PA11	11	Normal I/O
PA12	12	Normal I/O
PA13	13	Normal I/O
PA14	14	Normal I/O(AD0)
PA15	15	Normal I/O(AD1)
PA16	16	Normal I/O(AD2)
PA17	17	Normal I/O (TWI)
PA18	18	Normal I/O (AD4)
PA19	19	Normal I/O (AD5)
PA20	20	Normal I/O (TXD1)



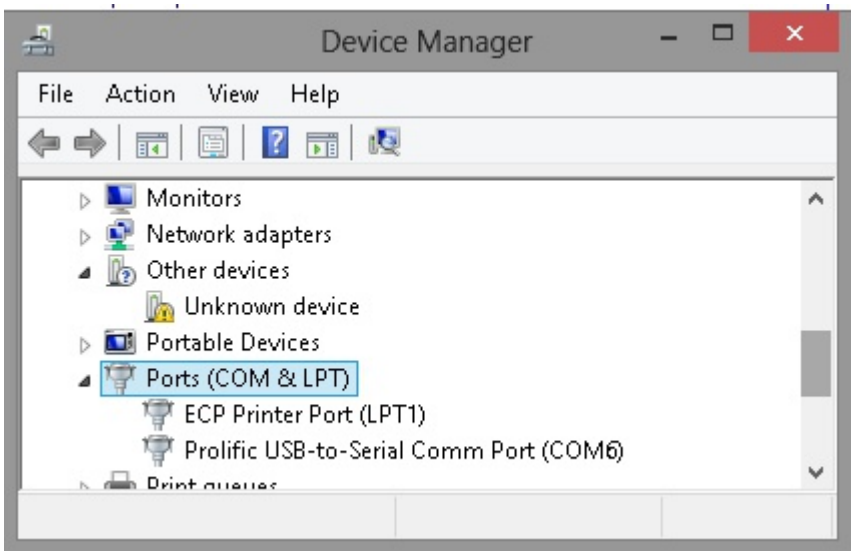
STEP1:
Open your e-Gizmo CD and open the folder e-Gizmo Kits. Inside you will find the folder of the AVR32 that includes three files, the AVR32 IDE zip file, Flip installer and P.bat.

Unzip the AVR32 IDE. Then after which, install FLIP 3.4.7.112 which includes the Java Runtime Environment required for the IDE.

STEP2:
Copy the batch file "P.bat" to your desktop. Right click the icon, then click edit.

When prompted, choose notepad for editing the batch file.

Once inside the editor, replace certain texts of the batch file depending on the parameters used for your project.



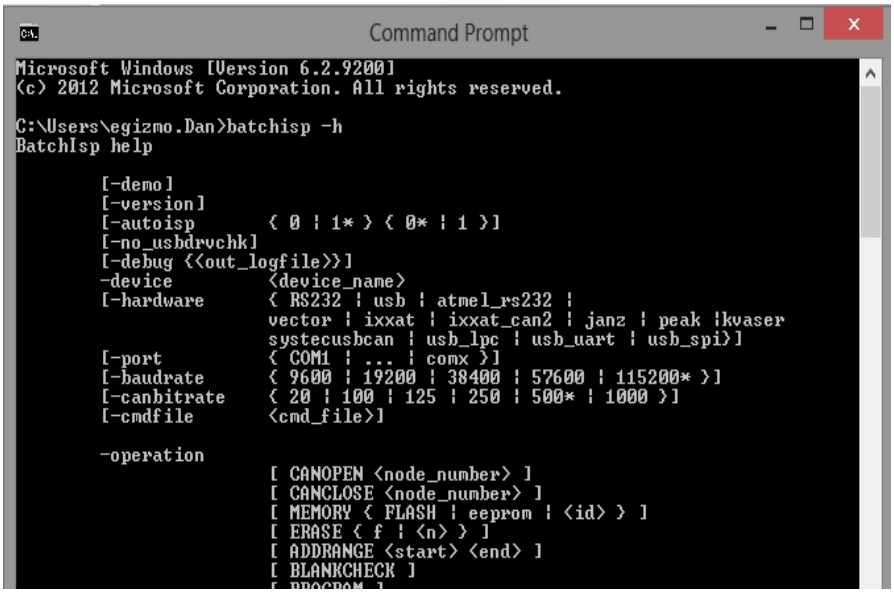

```
Batchisp -device AT32UC3L0256 -hardware RS232 -port COM6 -baudrate 115200 -operation onfail
abort memory flash erase f blankcheck loadbuffer BlinkTest.hex program verify start reset 0
```

Chip used:
AT32UC3L0256 OR
AT32UC3L0128

COM port used:
(found using device manager [windows button+pause/break])
Example:
COMX where X = 0,1,2,3,4

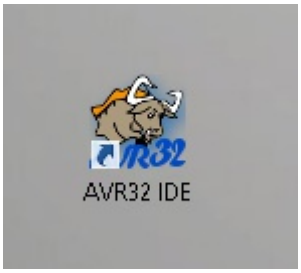
Filename of the program or project:
Example:
HelloWorld.hex

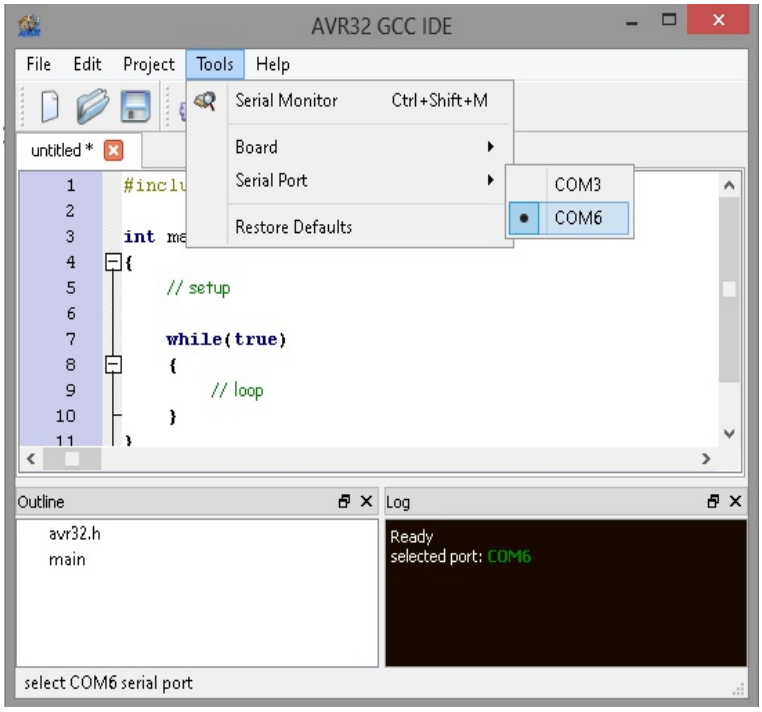
STEP3:
Just in case you want to check out what the batch file means, run command prompt and type in "batchisp -h". This command shall confirm whether FLIP has been properly installed and lists all the commands one can use using the batchisp.



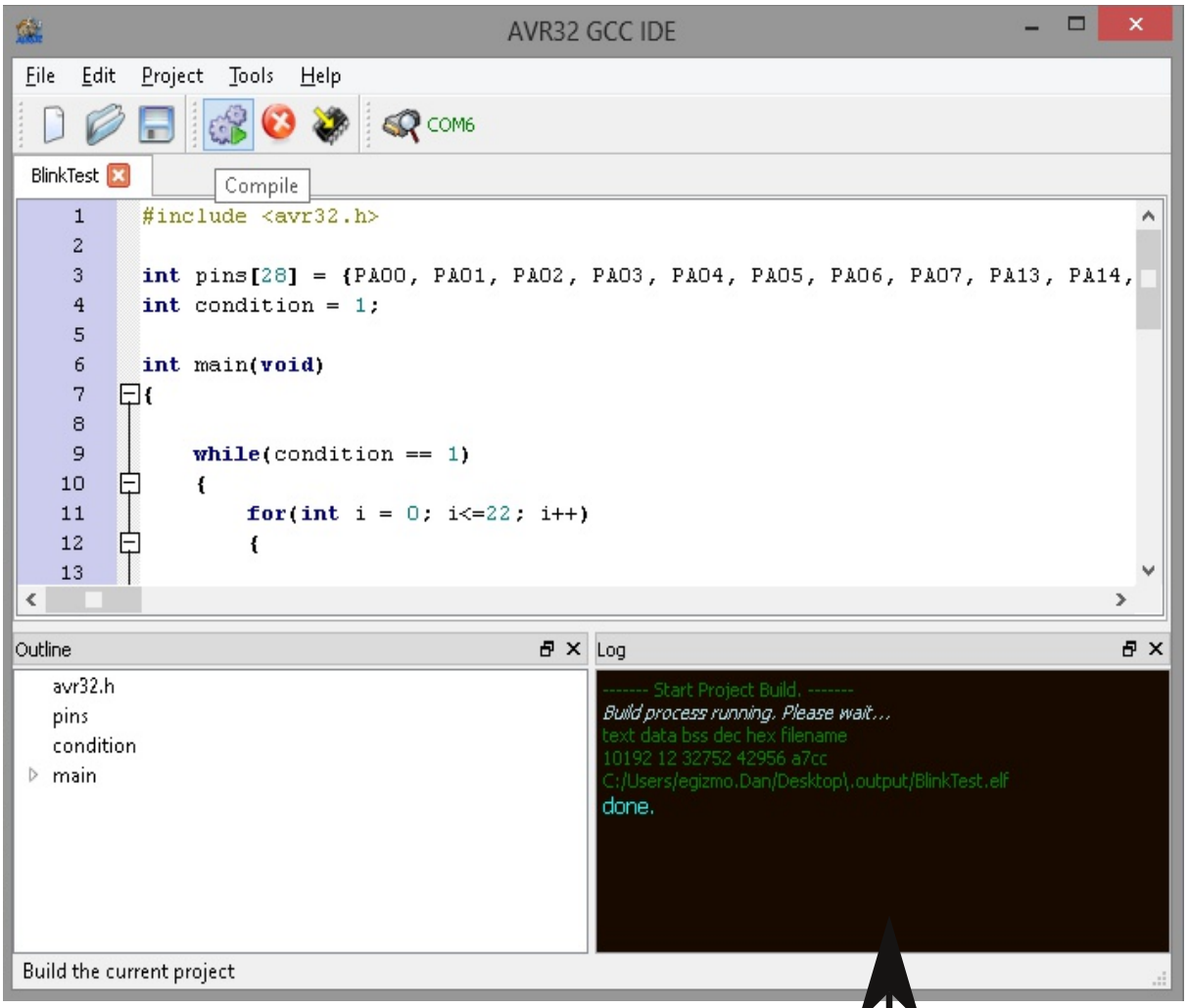
UPLOADING PROGRAMS:

STEP4:
Go back to your AVR32 folder then open the AVR32 IDE. Once opened, select the right COM port typed inside the batch file recently by opening Tools > Serial Port.





STEP5:
Before uploading, copy the sample program located at avr32-gcc-ide-svn191 > avr32-ide > examples > 01.Basics > BlinkTest.hex, to your desktop. Drag this file to your IDE and this will show up on your IDE:

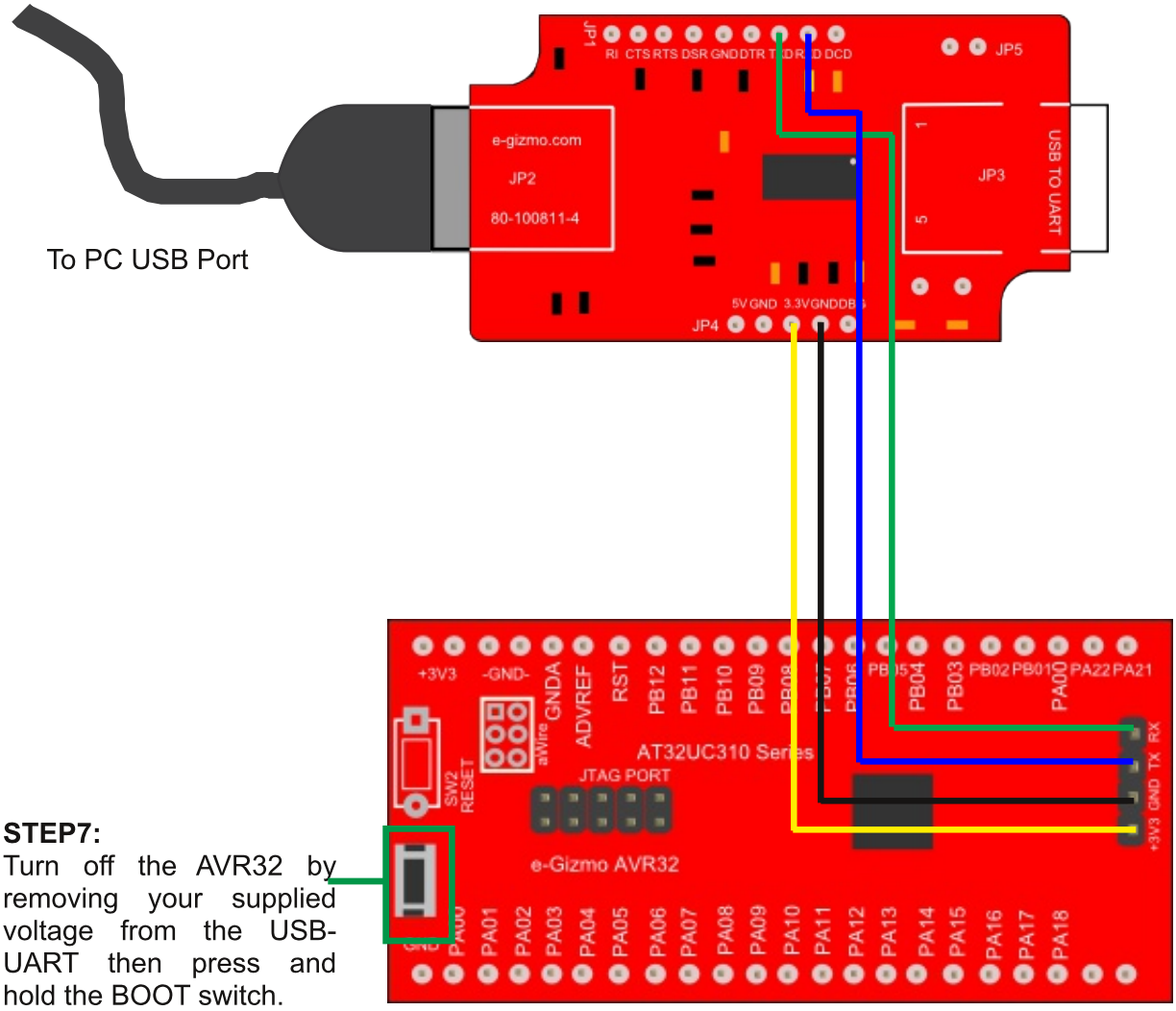


STEP6:
For every changes on the program, you **need** to click compile for the IDE to check your codes. You will know that the codes are correct if the following will show up:

How to upload hex files on your AVR32 using the USB-UART:

Connections:

USB-UART	AVR32
+3.3V	+3.3V
GND	GND
TXD	RX
RXD	TX



STEP7:
Turn off the AVR32 by removing your supplied voltage from the USB-UART then press and hold the BOOT switch.

***Figure 5**. Wiring connections of the AVR32 to the USB-UART*

STEP8:
Then while holding the BOOT switch, put back the supply voltage to turn on the AVR32 once again. Release the BOOT switch and you may now upload your program. Uploading is successful if you see the following statements in the log below:

Congratulations and enjoy programming your high speed MCU.
NOTE: STEP7 should be done every time you want to upload your program and each time you change your codes, it should be compiled first before uploading.