
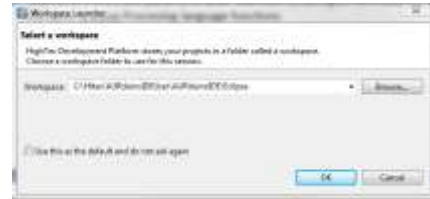


## Using The Eclipse IDE

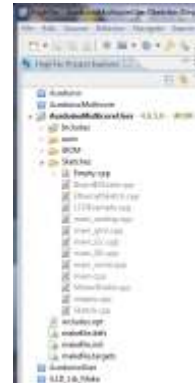
If you want to use the Eclipse environment, start the toolchain with the  icon. When prompted, open the workspace at:

C:\Hitex\AURduinoIDE\Eclipse




The default project is AURduinoMulticoreUser:

Arduino-style sketches are stored in the Sketches directory. The default sketch “Empty.cpp” is a simple program that uses all three cores. You can overwrite the statements we used with your own.



## Debugging Programs Using The Standalone PLS UDE Debugger




To get your programs into the ShieldBuddy, use the  PLS UDE debugger

Open the workspace (ShieldBuddy with TC275 DC step processor):

“C:\Hitex\AURduinoIDE\Eclipse\AurduinoMulticoreUser\.ude\ShieldBuddyWorkspace\_27xD.wsx”

Or (ShieldBuddy with TC275 CA step processor) :

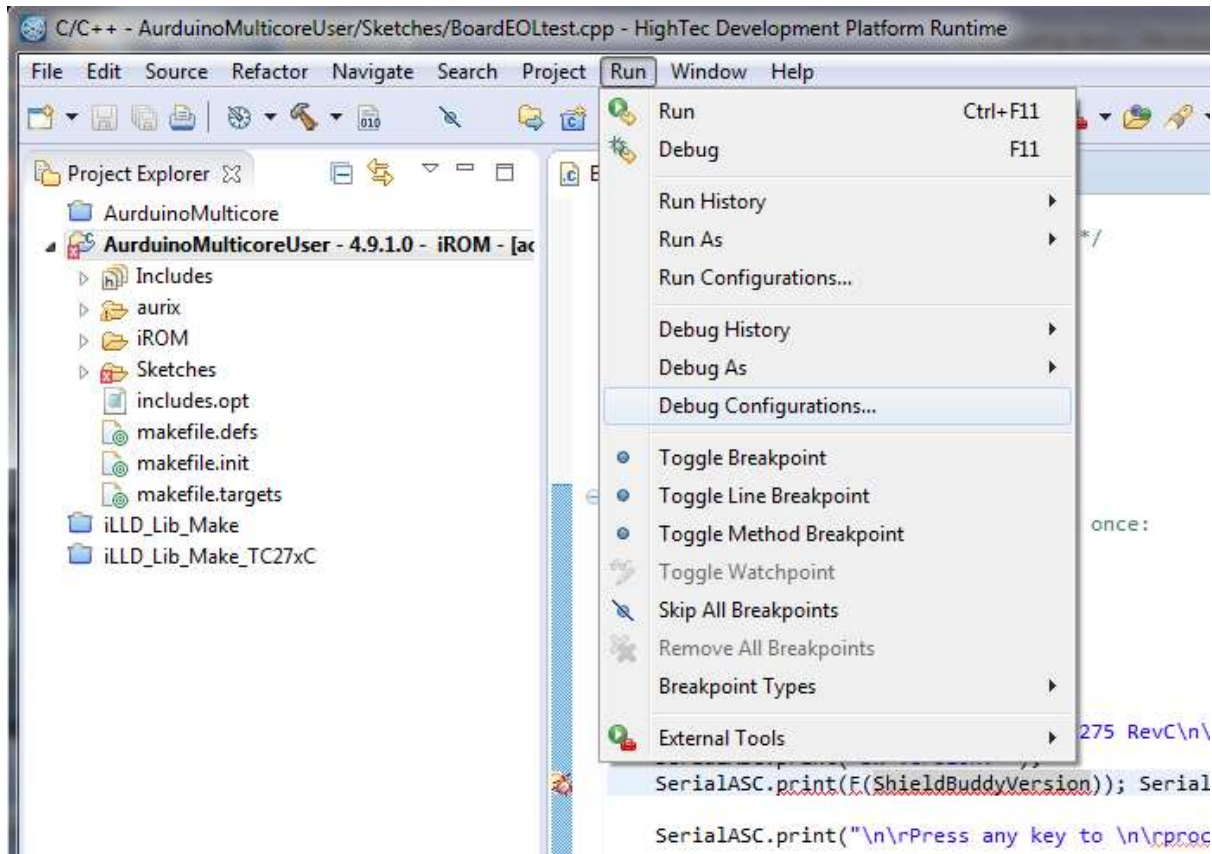
“C:\Hitex\AURduinoIDE\Eclipse\AurduinoMulticoreUser\.ude\ShieldBuddyWorkspace\_27xC.wsx”


The program will automatically load. You can run it by clicking the  icon and stop it with the  icon. To reset the program, use the  icon. You can find more information on using the Eclipse tools and the PLS UDE debugger in the guide supplied with the FreeToolChain.

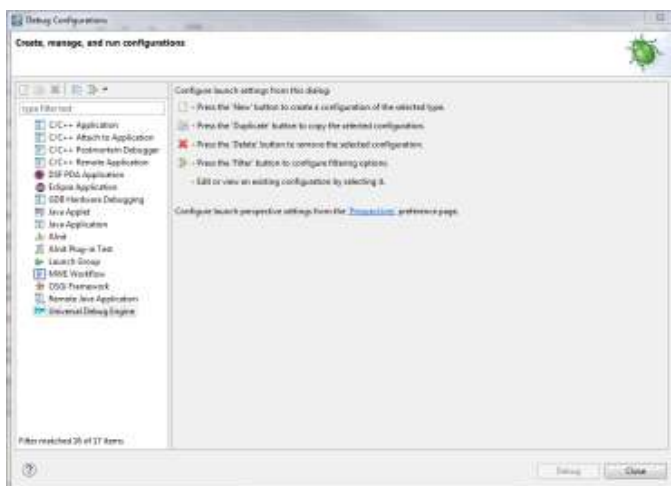
## Debugging Programs Using Eclipse PLS UDE Debug Perspective

Unfortunately some configuring is required to use the UDE perspective for debugging under Eclipse. This is performed as follows. Please note that this will depend on whether you have a TC275 CA or DC step on your ShieldBuddy.

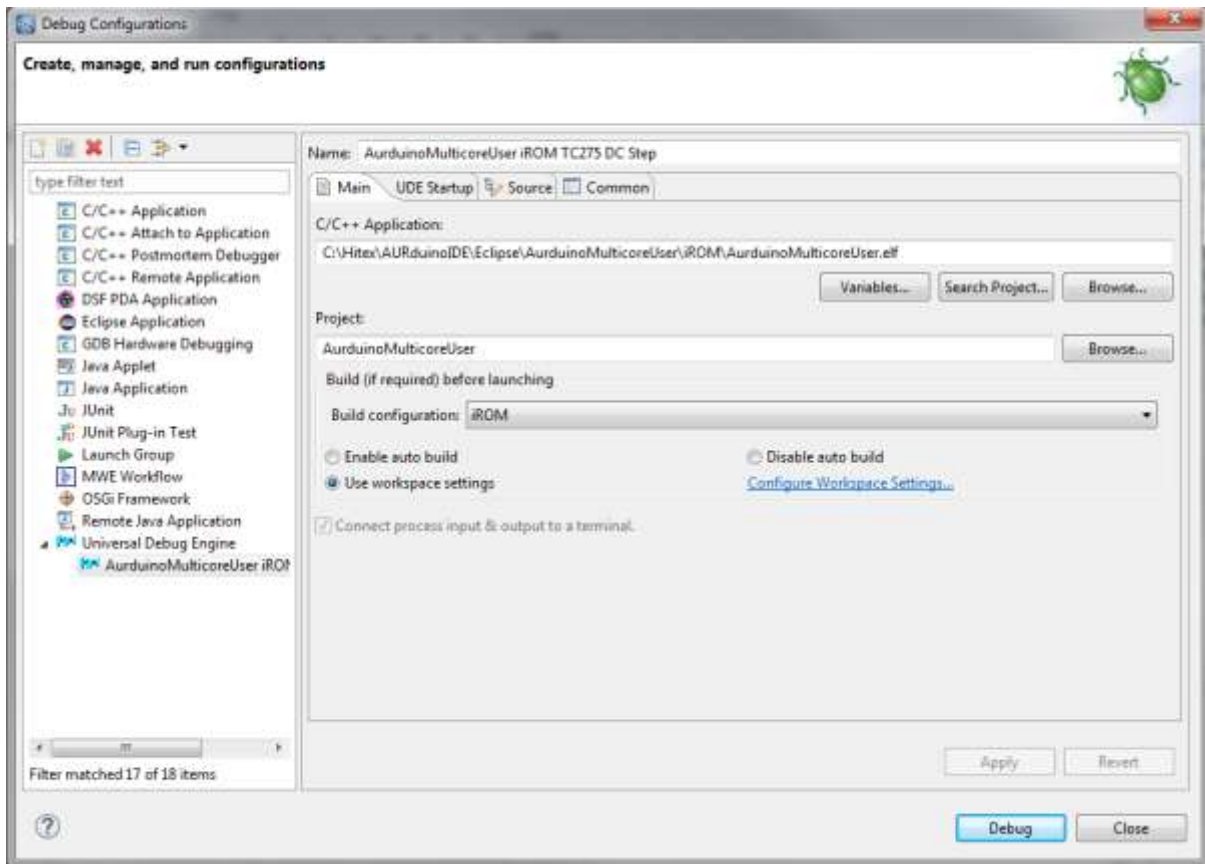
Create a new Debug Target to suit your Aurix version. Here we will do the TC275 DC first.



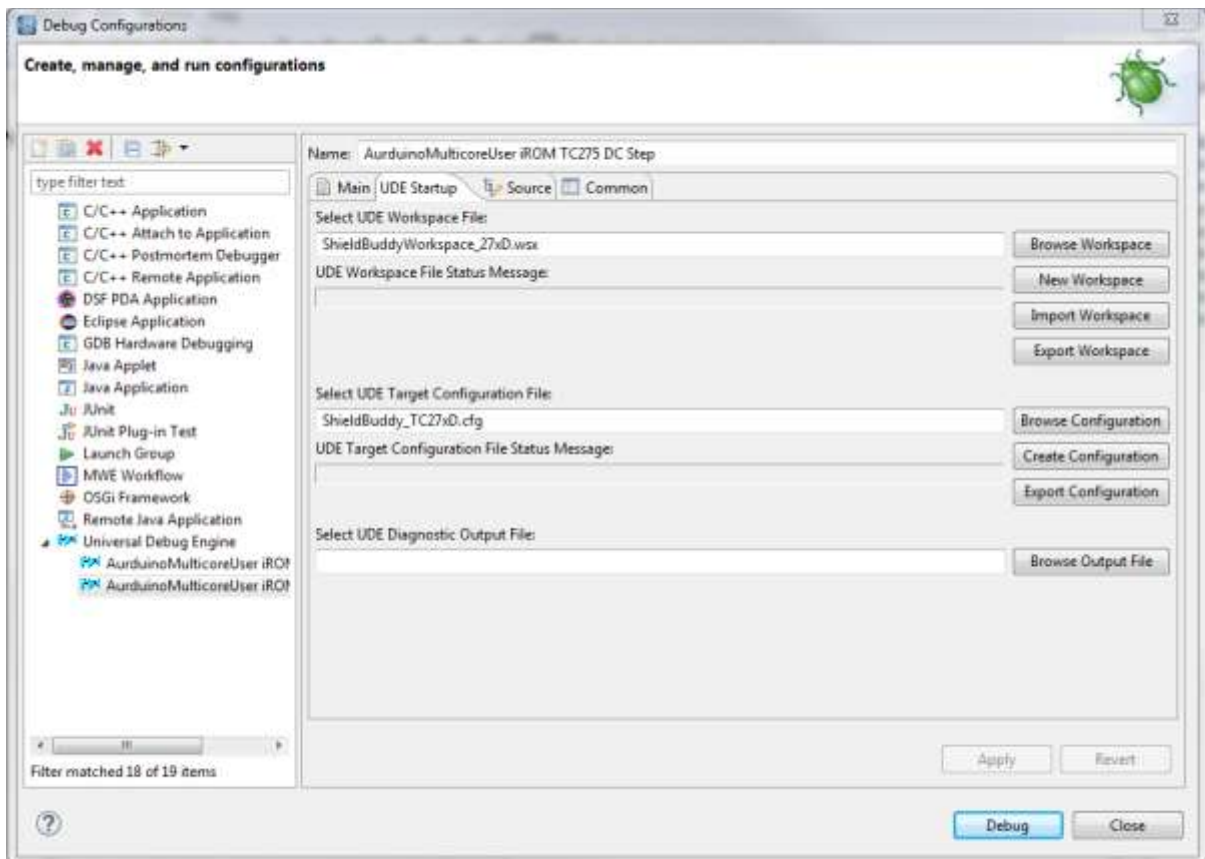
Then create a new debug configuration with  ...



Here we are setting up for the TC275 DC step. The C/C++ Application will be as shown – you will need to browse for this.



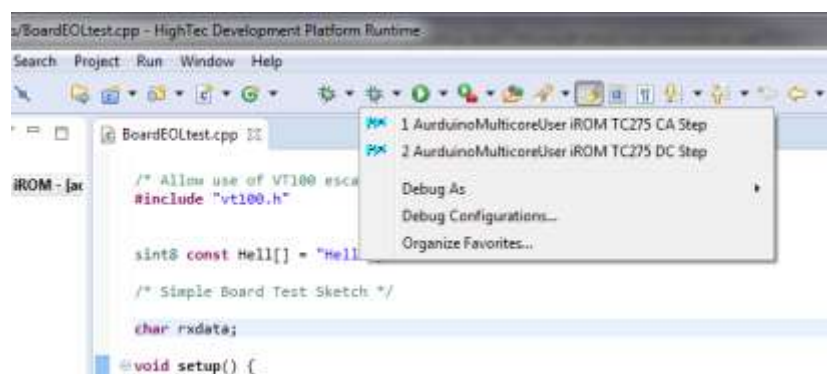
Under the “UDE Startup” tab, you need to specify the UDE workspace file to use. This is predefined in “C:\Hitex\AURduinoIDE\Eclipse\ArduinoMulticoreUser\.ude”. The Target Configuration file is predefined in C:\Hitex\AURduinoIDE\Eclipse\ArduinoMulticoreUser\.ude\target”. You will need to browse for both of these files.



Now click the Apply button. You can run the debugger directly from here using “Debug”...



...or you can do it from the Debug icon in Eclipse.



The UDE debug perspective should

now

open...

```
#include <stdint.h>
#include "UART.h"
#include "UART15C.h"

#define BAUD_RATE 9600

void setup() {
  // set up UART
  UART.begin(BAUD_RATE);
  UART15C.begin(9600);
  UART15C.print("VT200");
  UART15C.print("VT200");
  UART15C.print("VT200");

  Serial15C.print("Test");
  Serial15C.print("VT");
  Serial15C.print("VT200");

  Serial15C.print("VT");

  char* rdata = char[Serial15C];
  Serial15C.print("VT");
  Serial15C.print("VT200");
  Serial15C.print("VT");

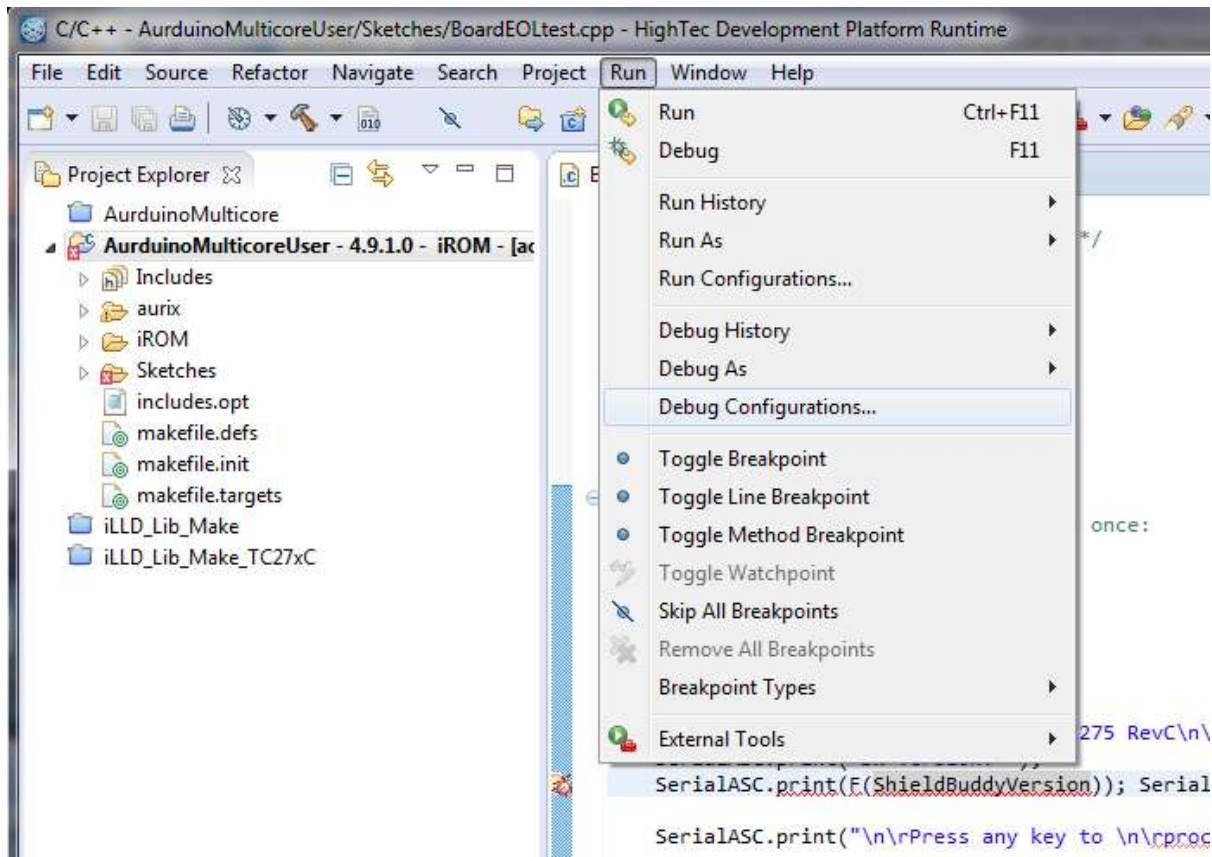
  // enable the LED
  pinMode(LED, OUTPUT);
}


void loop() {
  // enable the LED
  digitalWrite(LED, HIGH);
}
```

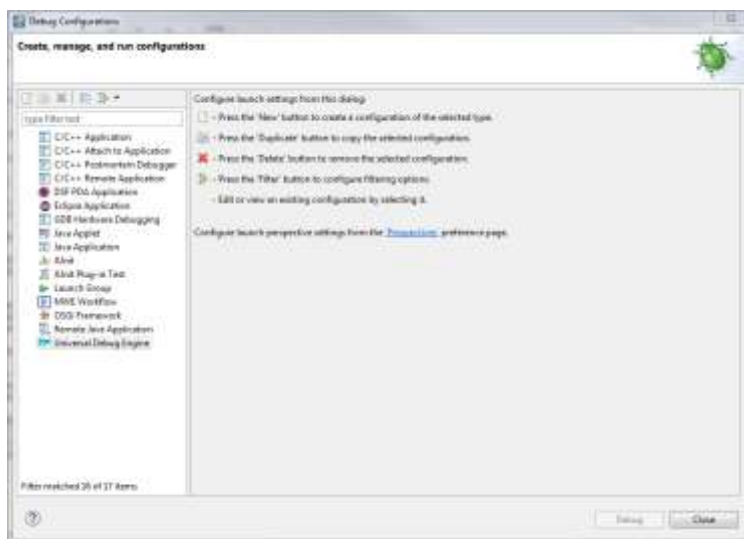
ID	Type	Time	Source	Message
17	Success	10:55:48	Core0: UART15C	Connection to TC270C target established: Triforce (Core0)
18	Success	10:55:48	Core0: UART15C	Connection to TC270C target established: Triforce (Core0)
19	Success	10:55:49	Core0: UART15C	Program with ID 0x1 - code size 113994 bytes - was loaded!
20	Success	10:55:49	Core0: UART15C	Program with ID 0x1 - code size 113994 bytes - was loaded!
21	Success	10:55:49	Core0: UART15C	Program with ID 0x1 - code size 113994 bytes - was loaded!
22	Info	10:55:55	Core0: UART15C	Alternate Boot Mode detected
23	Info	10:55:55	Core0: UART15C	Progressing with special loader headline ...
24	Success	10:56:02	Core0: UART15C	Program successfully succeeded!
25	Info	10:56:25	WorkSpace	Copy C:\Users\ATB\Documents\Eclipse\Ardino\multicore\firmware\Shield\firmware\workspace_270C.exe C:\Users\ATB\Documents\Eclipse\...

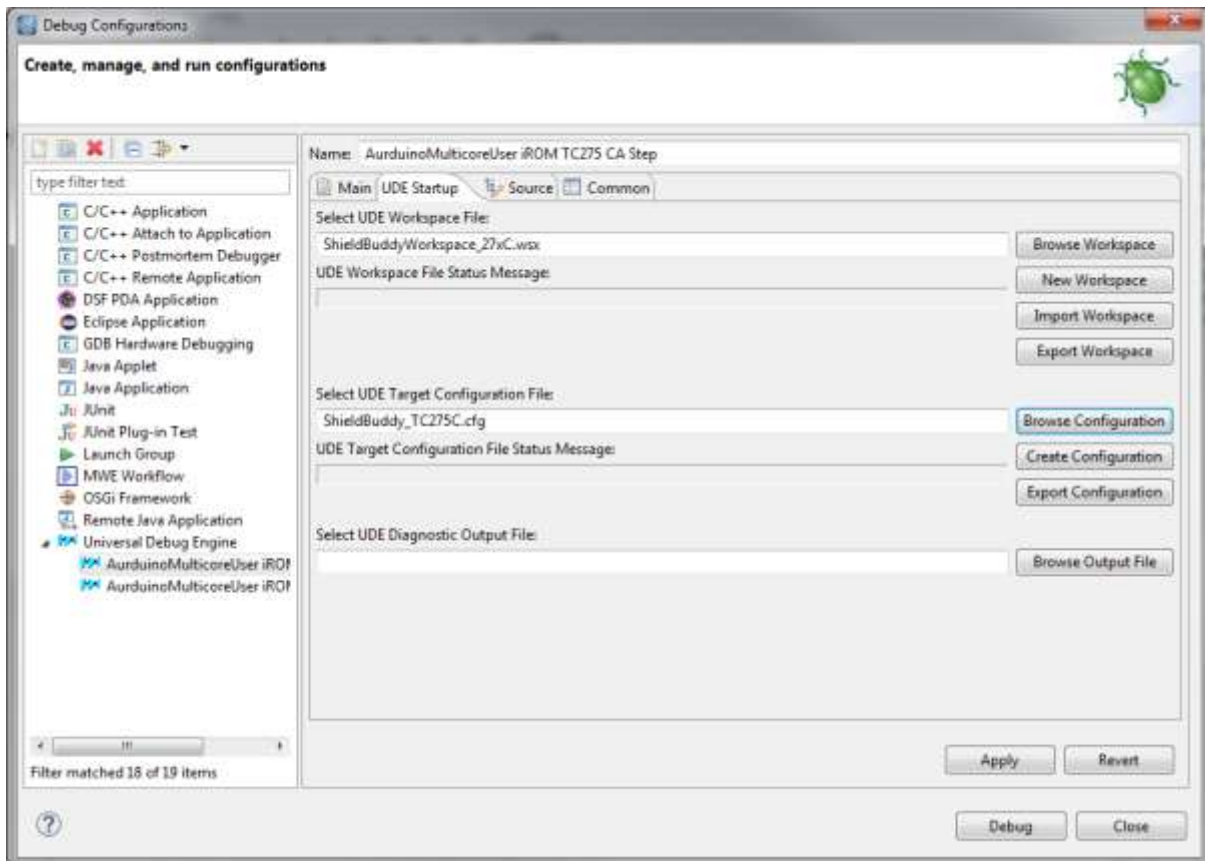
If you are using a TC275 CA step board then you need to define a debug configuration for this in the same way as we did for the DC step.

Create a new Debug Target ...



Then create a new debug configuration with  ...





Now click the Apply button. You can run the debugger directly from here using “Debug”...



...or you can do it from the Debug icon in Eclipse.



You can now use the UDE debug perspective in the usual; Eclipse manner!





