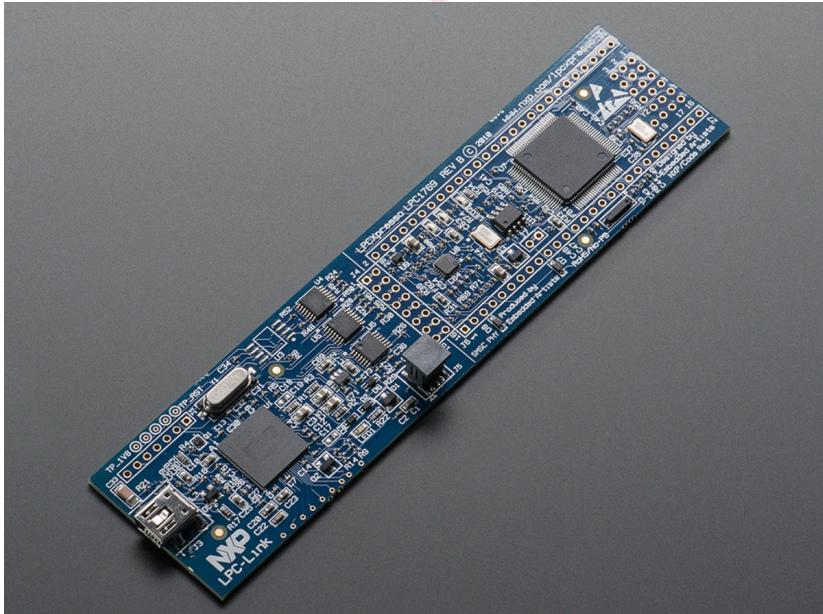




2016

UART(Serial Communication) with NXP LPC1769 using LPCXpresso



Author: Gurudatta Palankar

Reviewers:

Version: 1.0

Introduction:

LPCXpresso™ is a new, low-cost development platform available from NXP supporting NXP's ARM-based microcontrollers. The platform is comprised of a simplified Eclipse-based IDE and low-cost target boards which include an attached JTAG debugger. LPCXpresso™ is an end-to-end solution enabling engineers to develop their applications from initial evaluation to final production.

Step 1: Open LPCXpresso IDE

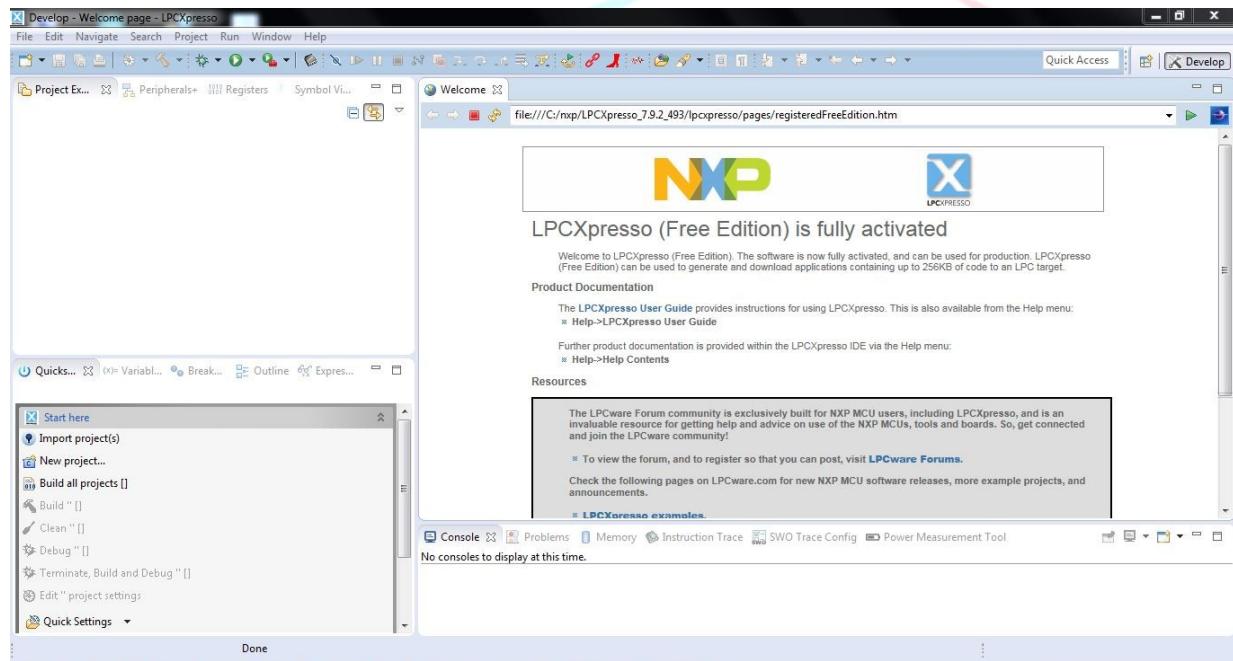


Figure 1

TENET
TECHNETRONICS

Step 2: Before writing a code, we have to Import some Library Files to the Workspace. Click on Import projects on Quickstart Panel on the bottom left of the window.

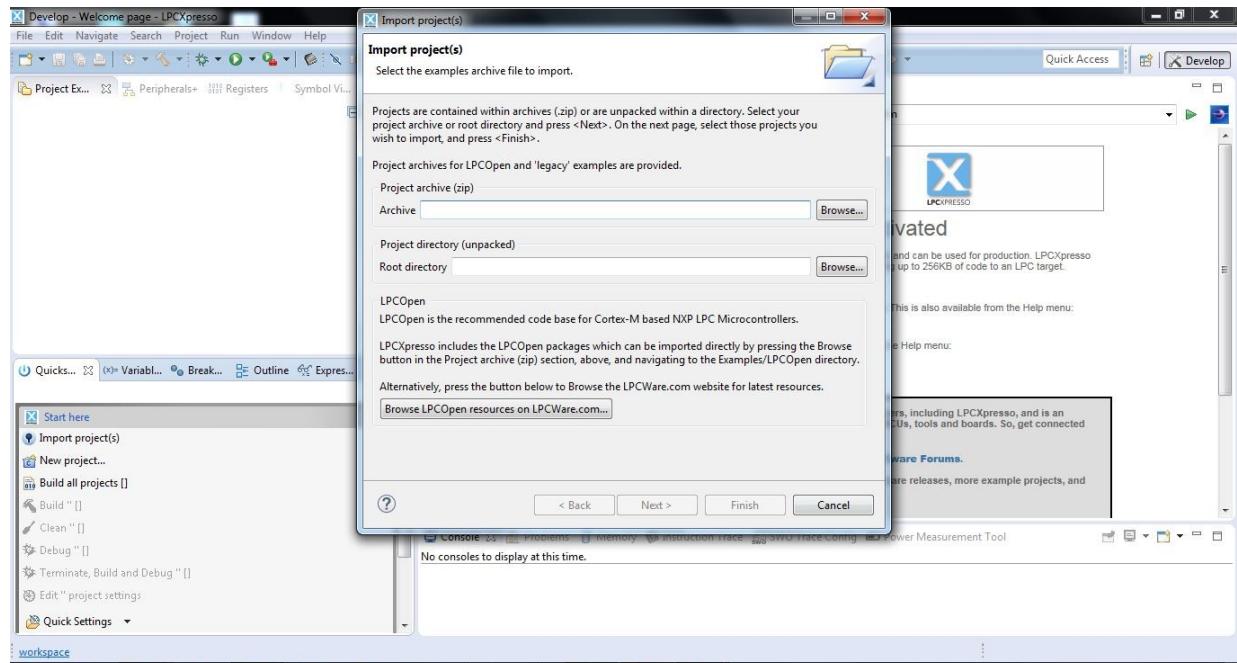


Figure 2

Step 3: Browse file, open the LPC1000 folder.

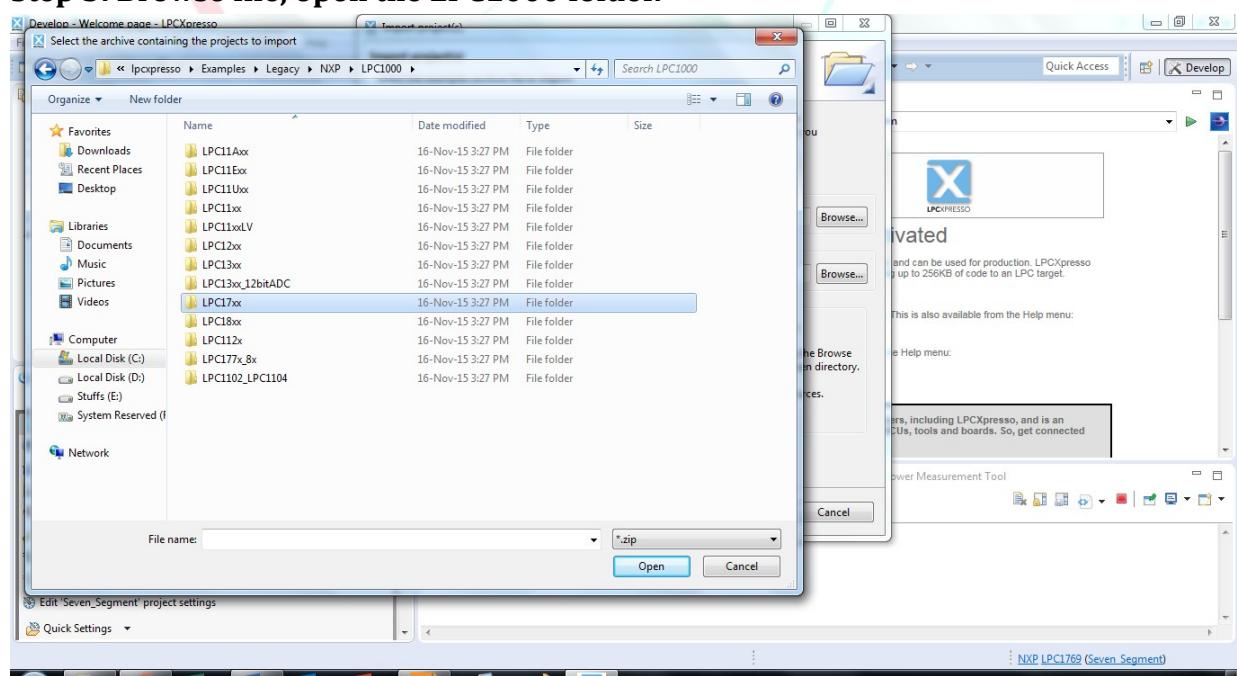


Figure 3

Step 4: Select the appropriate archive file. Let us select LPCXpresso176x_cmsis2. We can select CMSIS CORE library that include LPC17xx.h header file.

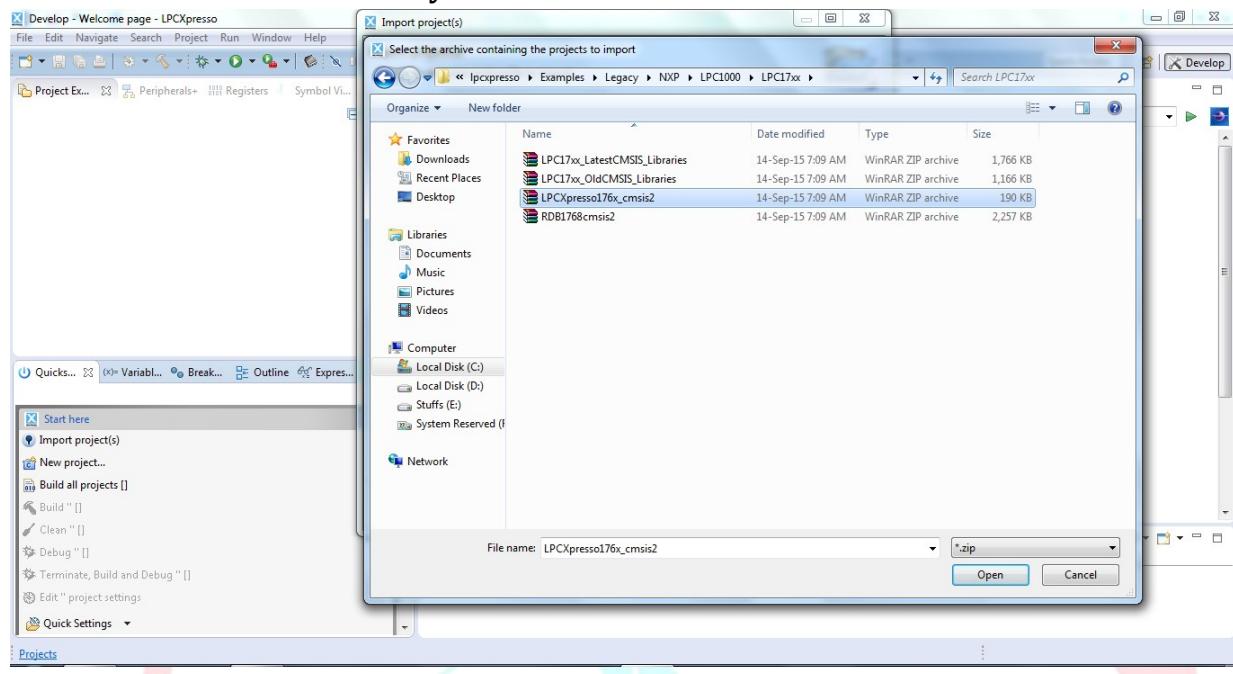


Figure 4

Step 5: After selecting you will be able to see the following libraries files. Let us select specific one.

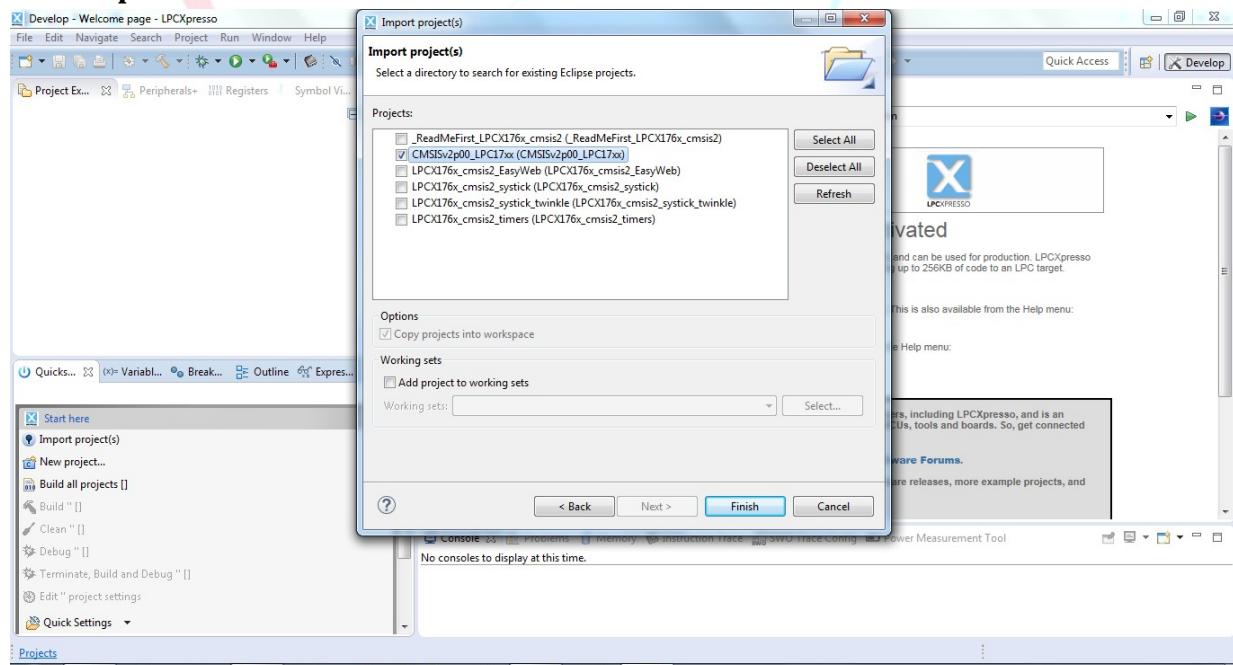


Figure 5

Step 6: Now we will be able to see those libraries in the workspace.

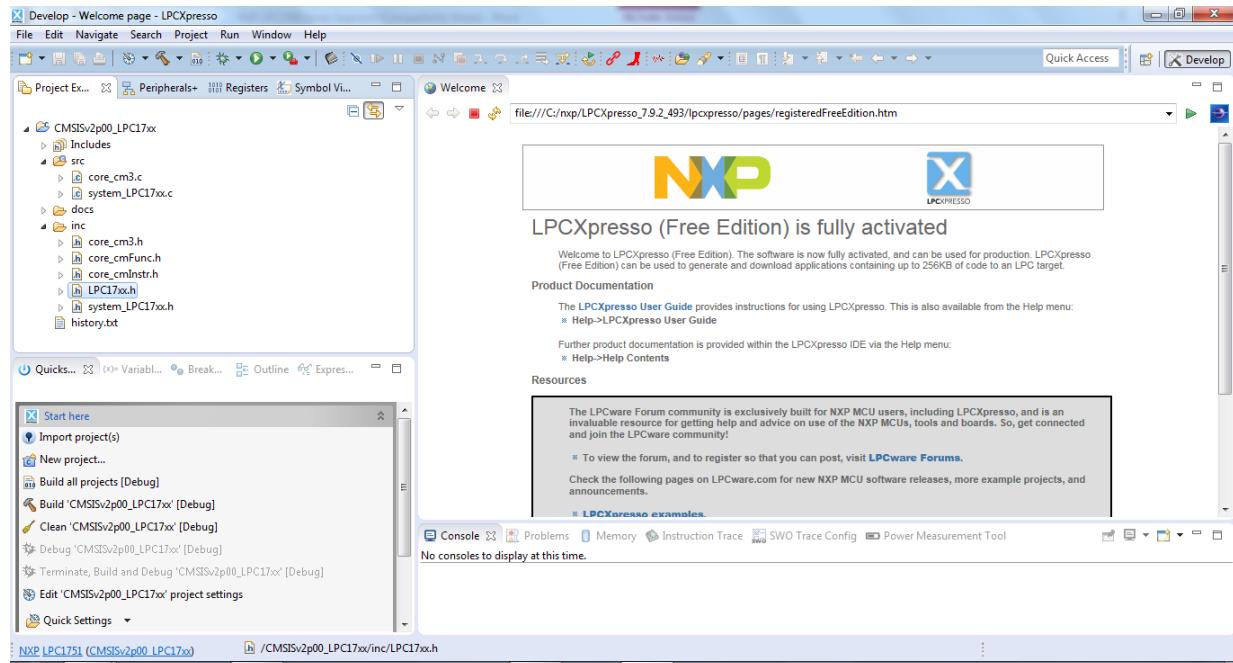


Figure 6

Step 7: Now we can start creating our new project. Goto File >> New >> Project. Select LPCXpresso C project.

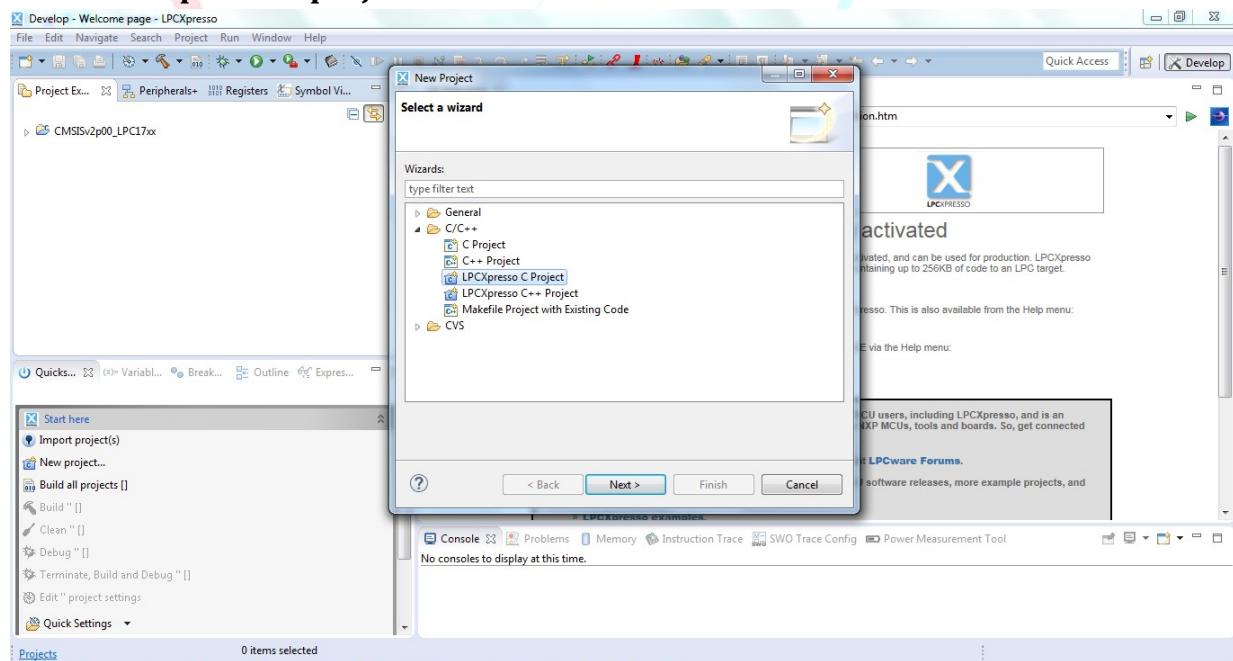


Figure 7

Step 8: Select LPC1769, C Project and give name to your project. Select target MCU as LPC1769.

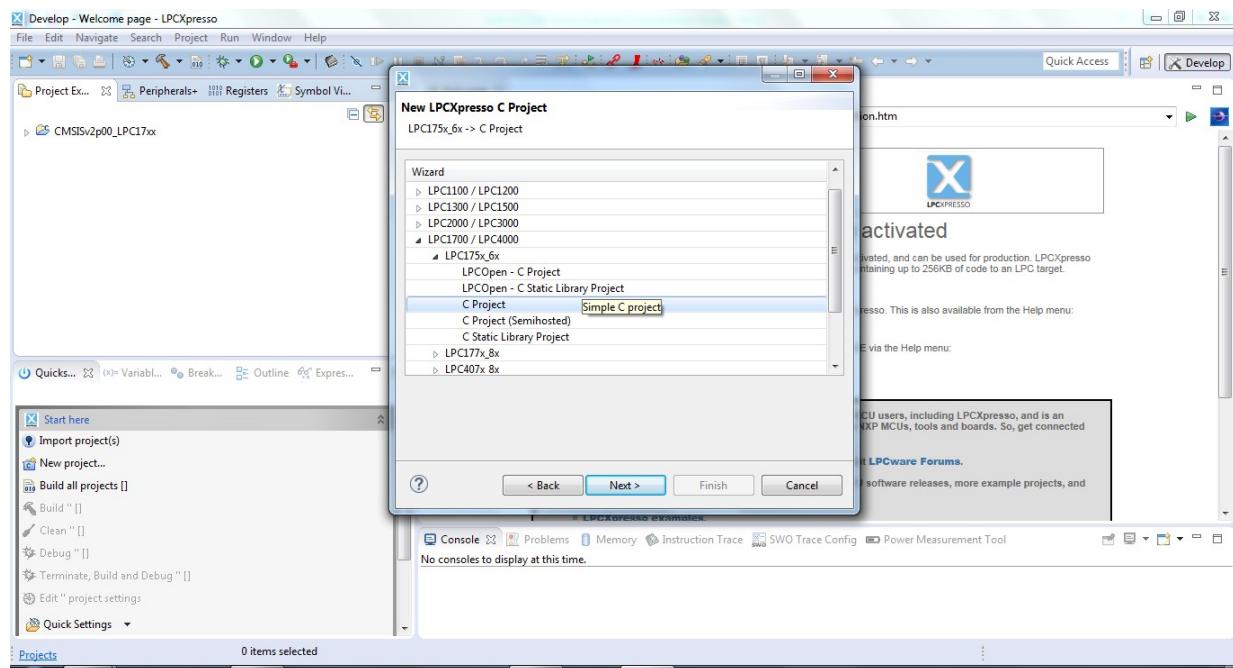


Figure 5

Step 9: Now select CMSIS Core library. Click on Next and keep all the other configurations as default and Finish.

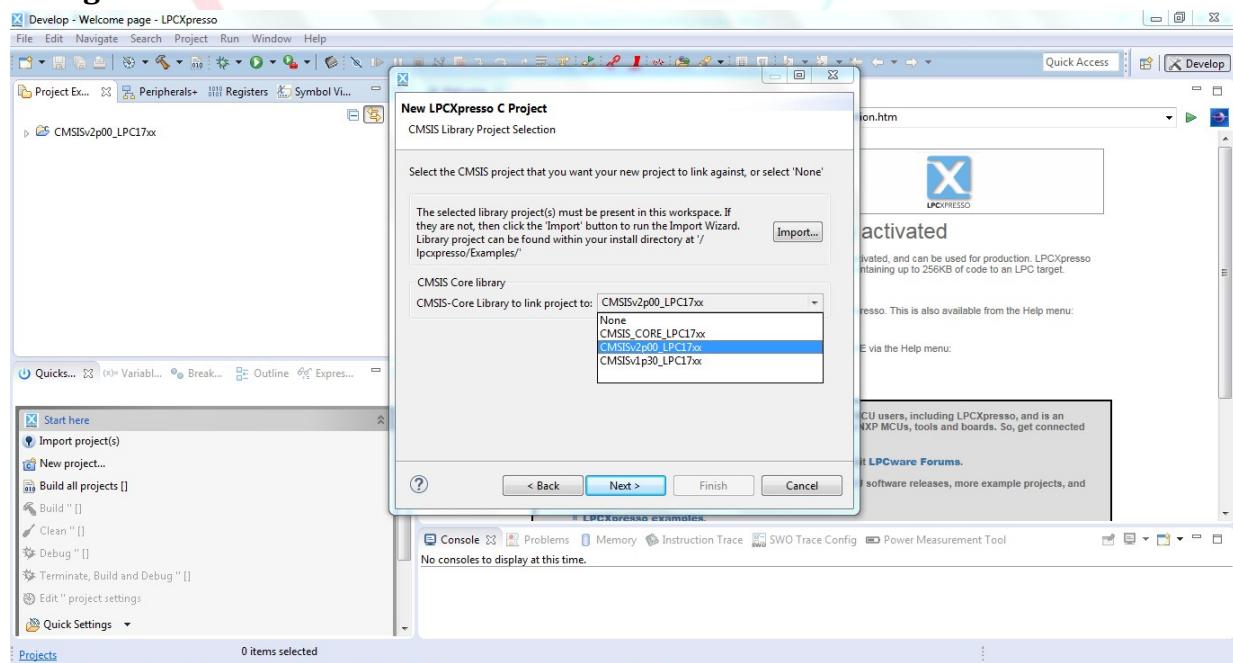


Figure 9

Step 10: Now we can see our project onto the workspace. Now by double clicking on LCD_display.c file, we can start writing code.

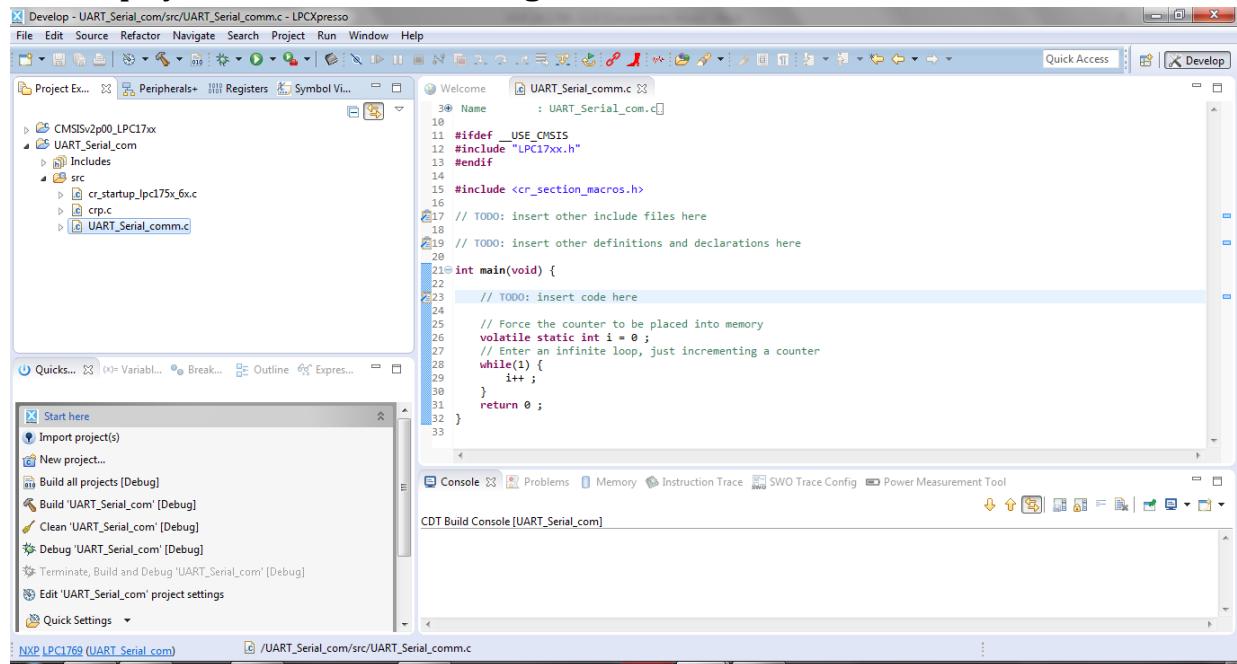


Figure 10

Step 11: Write a code as shown below.

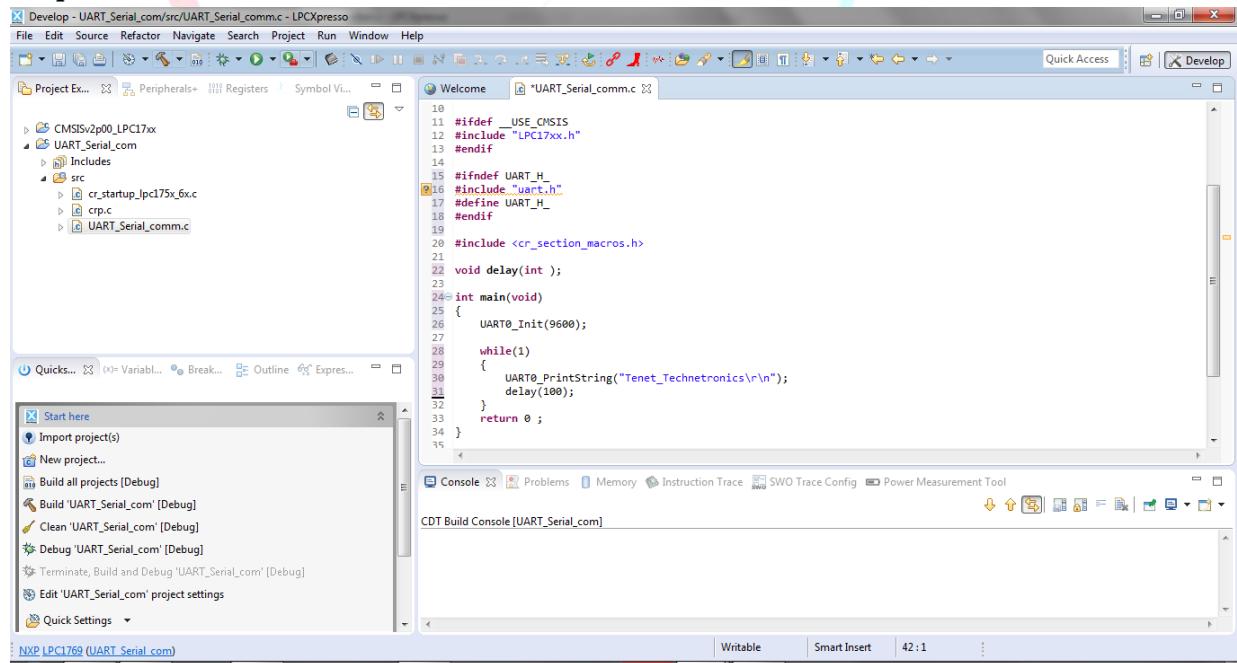


Figure 6

CODE:

```
#ifdef __USE_CMSIS
#include "LPC17xx.h"
#endif

#include "uart.h"

void delay(int );

int main(void)
{
    UART0_Init(9600);

    while(1)
    {
        UART0_PrintString("Tenet_Technetronics\r\n");
        delay(100);
    }
    return 0 ;
}

void delay(int a)
{
    int i,j;
    for(i=0; i<500000; i++)
        for(j=0; j<a; j++);
}
```

NOTE: The above code will not work until and unless we add Header File (uart.h) and Source File (uart.c) to the project.

TENET
TECHNETRONICS

Step 12: To create or add library files, right click on src file of your project file, then New > Source File.

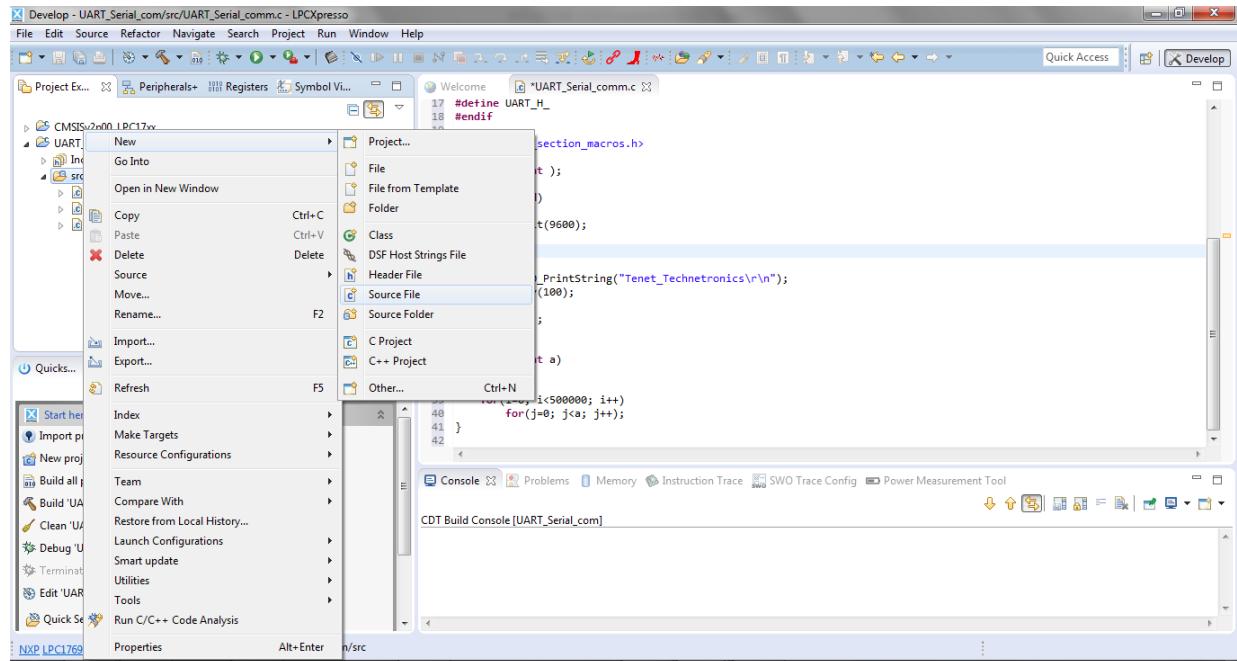


Figure 7

Step 13: Save the Source File name with .c file extension.

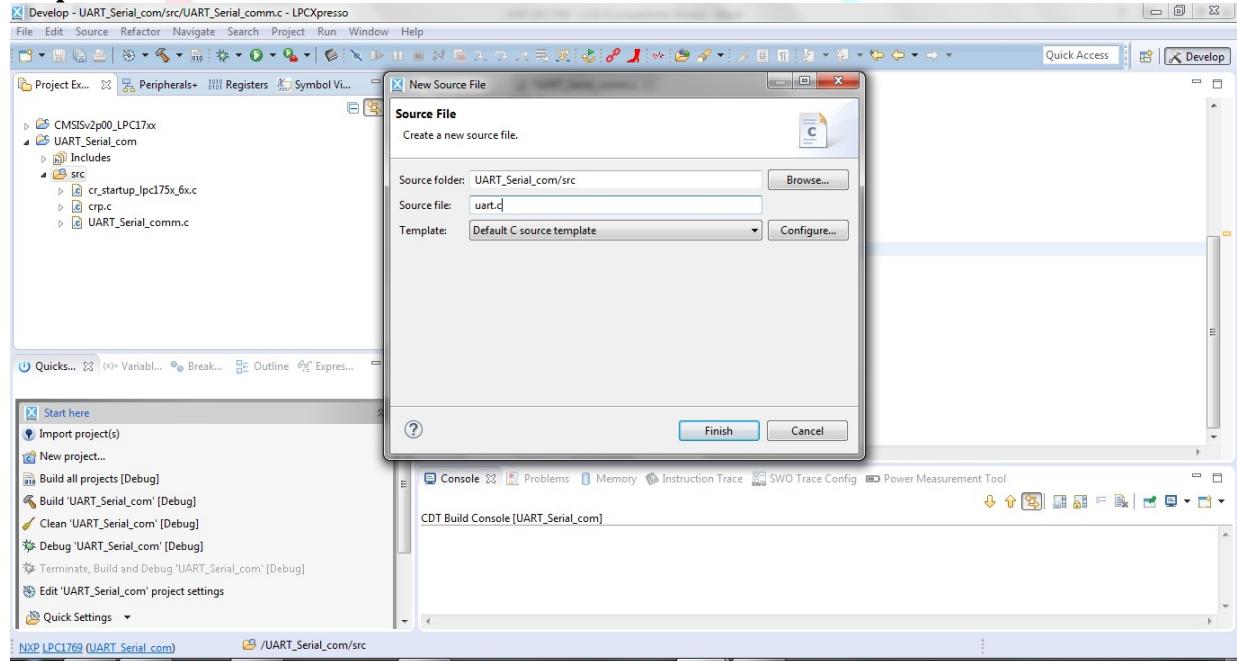
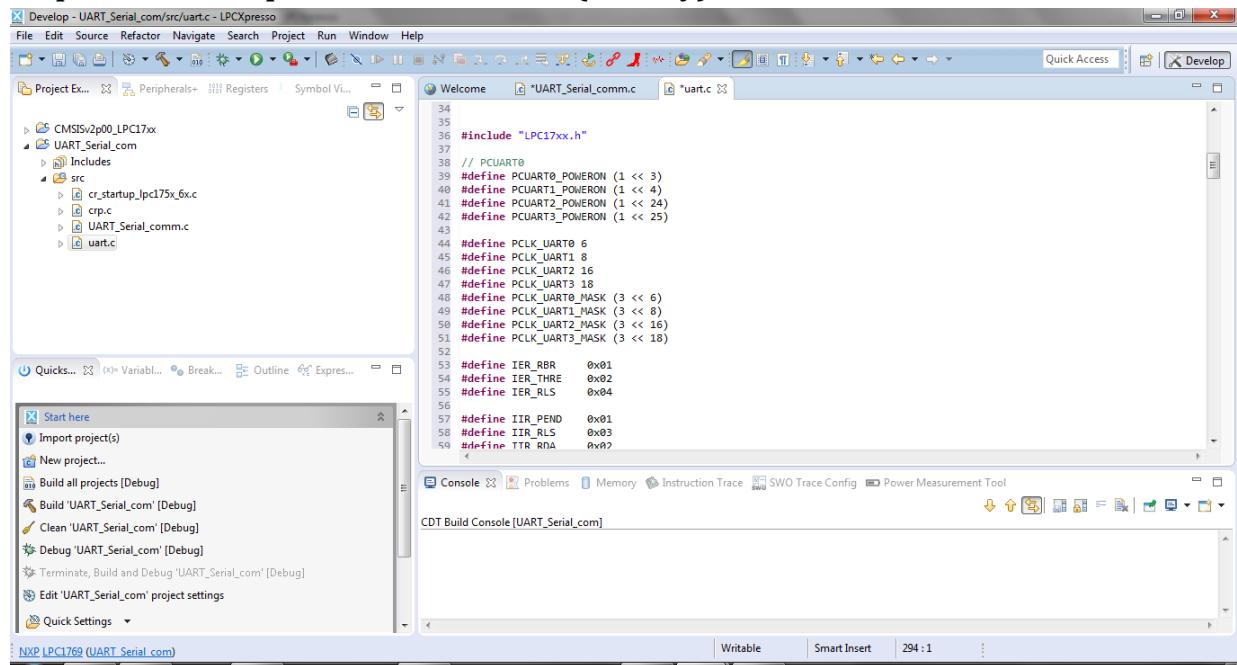


Figure 13

Step 14: Write or paste the Source File (Library).



```

34
35
36 #include "LPC17xx.h"
37
38 // PCUART0
39 #define PCUART0_POWERON (1 << 3)
40 #define PCUART1_POWERON (1 << 4)
41 #define PCUART2_POWERON (1 << 24)
42 #define PCUART3_POWERON (1 << 25)
43
44 #define PCLK_UART0_6
45 #define PCLK_UART1_8
46 #define PCLK_UART2_16
47 #define PCLK_UART3_18
48 #define PCLK_UART0_MASK (3 << 6)
49 #define PCLK_UART1_MASK (3 << 8)
50 #define PCLK_UART2_MASK (3 << 16)
51 #define PCLK_UART3_MASK (3 << 18)
52
53 #define IER_RBR 0x01
54 #define IER_THRE 0x02
55 #define IER_RLS 0x04
56
57 #define IIR_PEND 0x01
58 #define IIR_RLS 0x03
59 #define TTR RNA 0xA?
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400

```

Figure 14

Step 15: Similarly add the Header File.

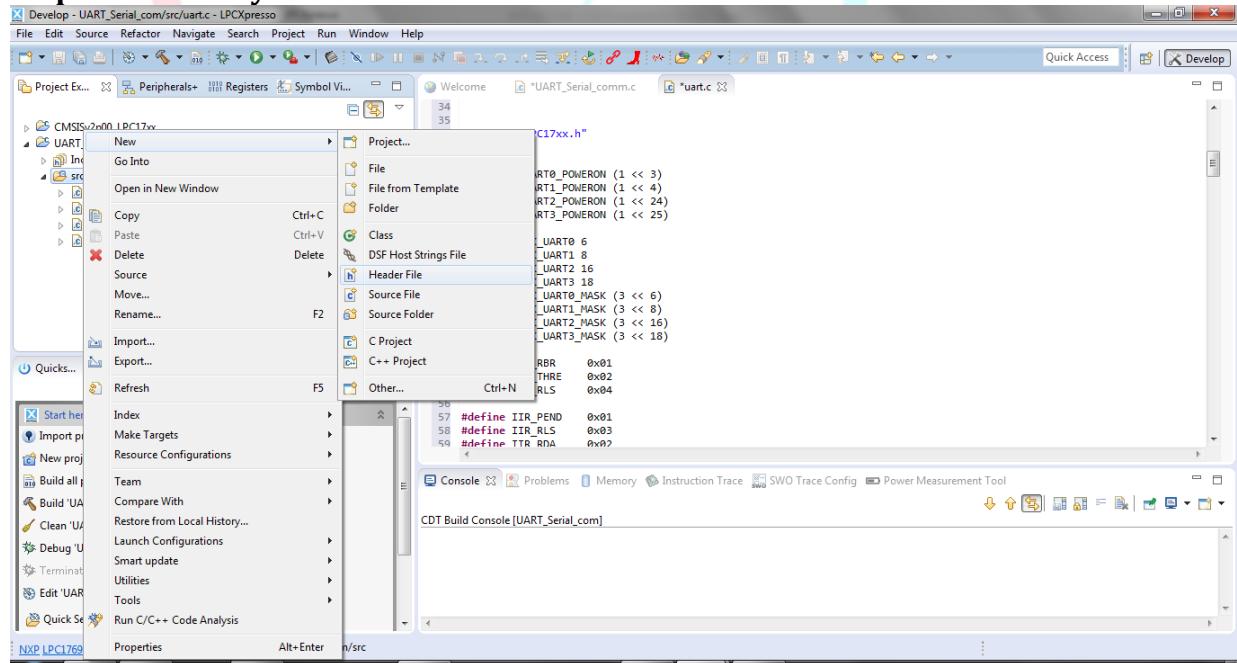


Figure 15

Step 16: Save the Header File name with .h file extension.

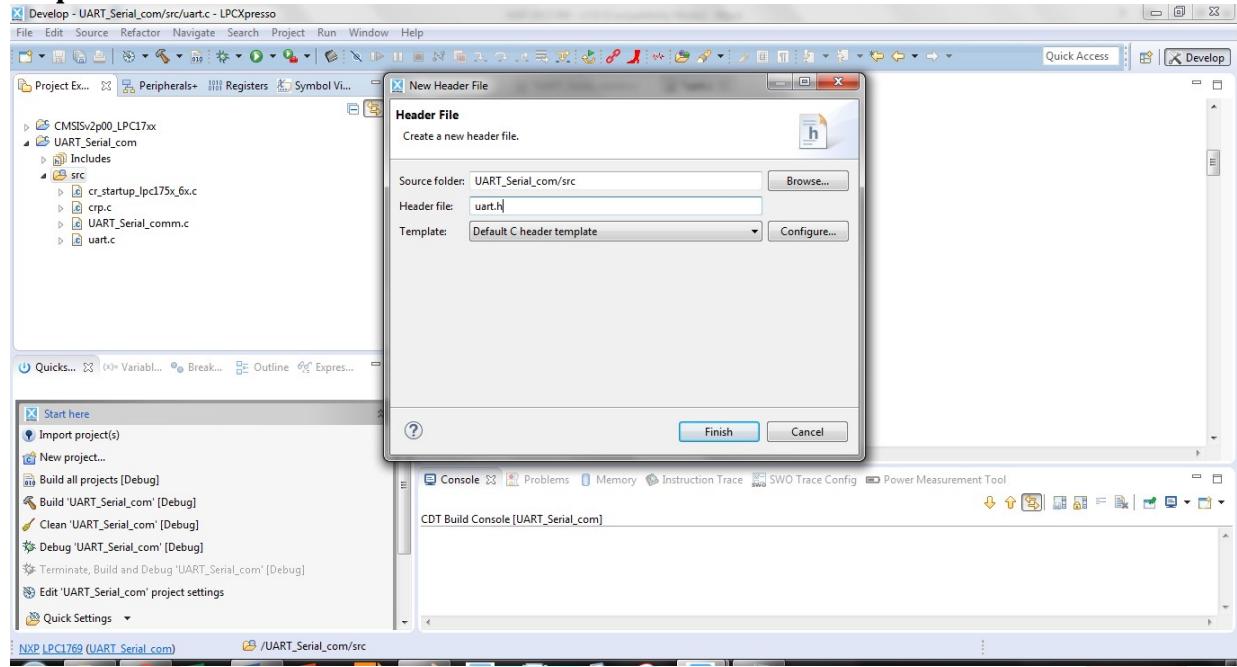


Figure 16

Step 17: Write or paste the Header File (Library).

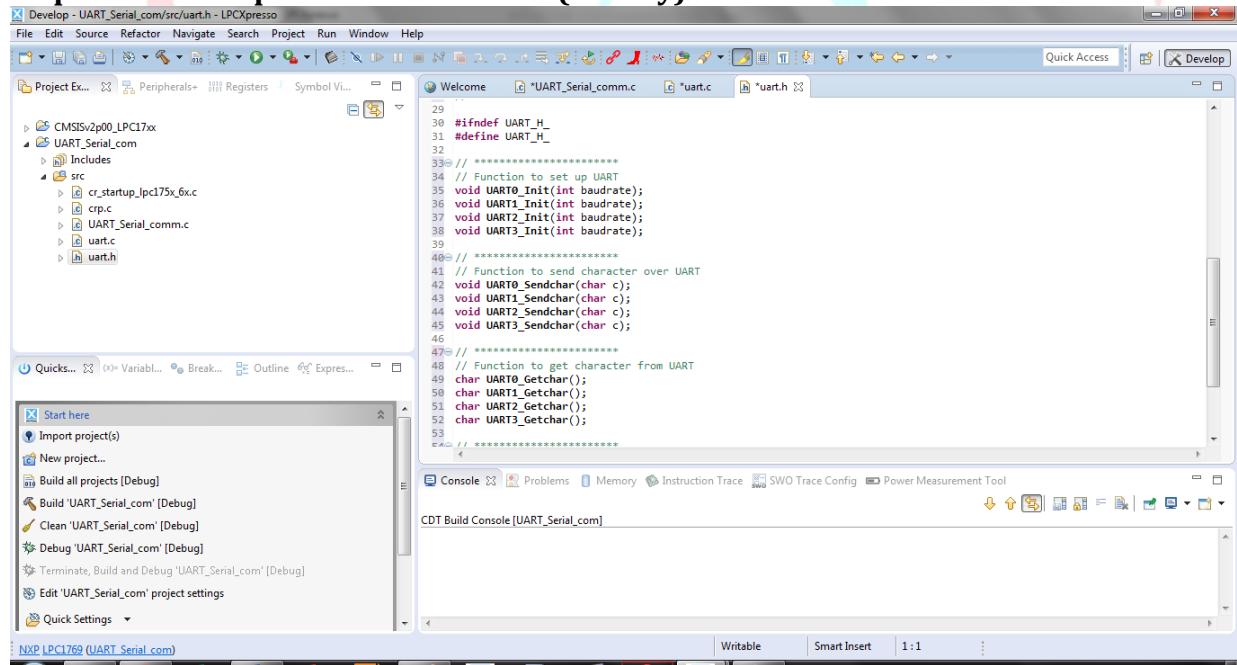


Figure 17

Step 18: After writing code and adding libraries, Build the project by clicking on Build UART_Serial_comm.c on the Quickstart Panel on the bottom left of the window.

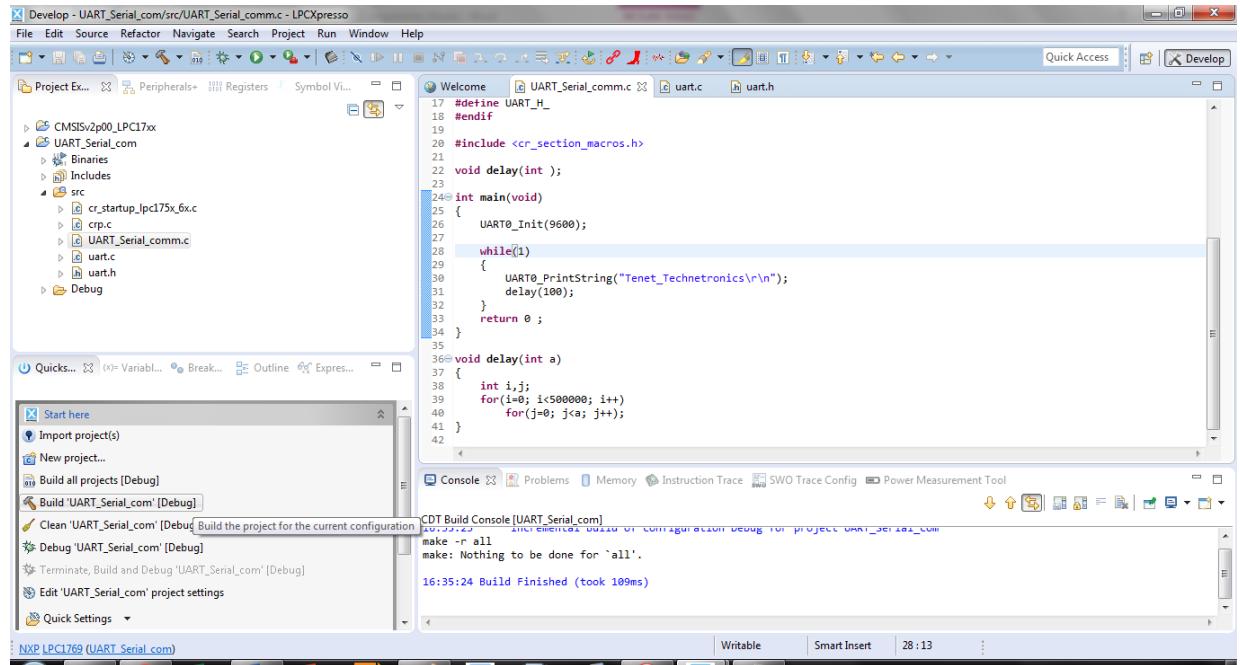


Figure 18

Step 19: Now, if all goes well connect the Micro B cable to LPC1769 and connect it to your computer. To upload the project file, click on the Program flash.

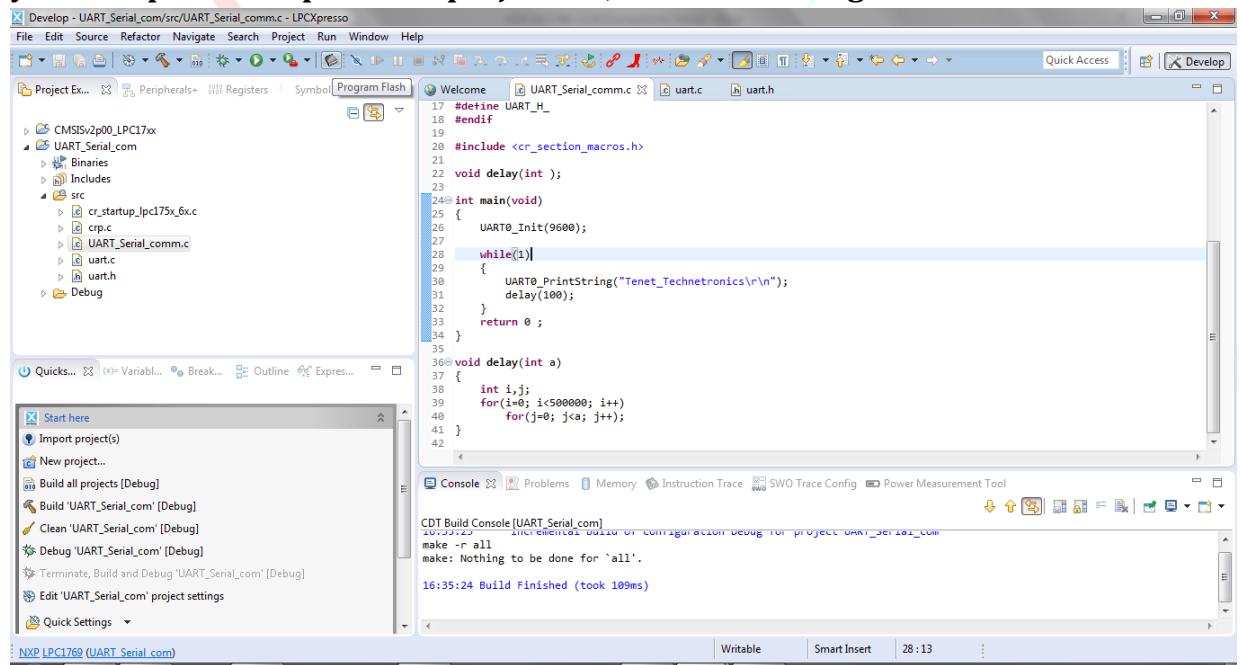


Figure 19

Step 20: Now select the Project file `UART_Serial_comm.axf`. We can find it in our project folder.

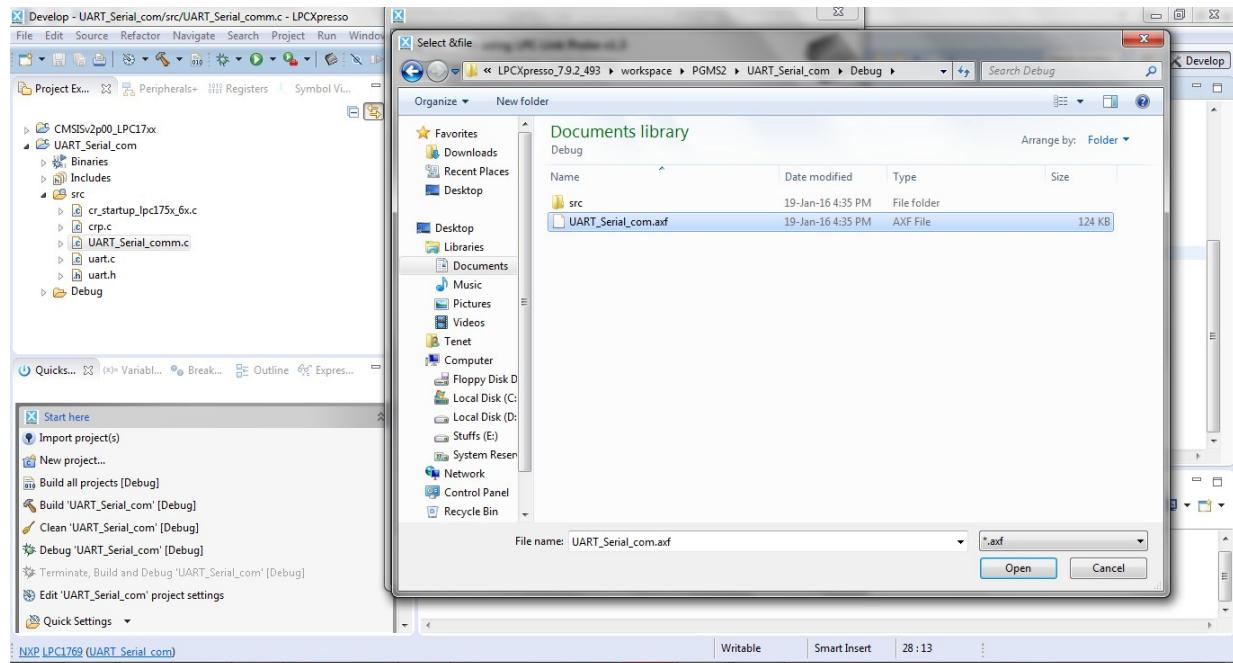


Figure 20

Step 21: Now this window shows we have finally dumped our project onto LPC1769.

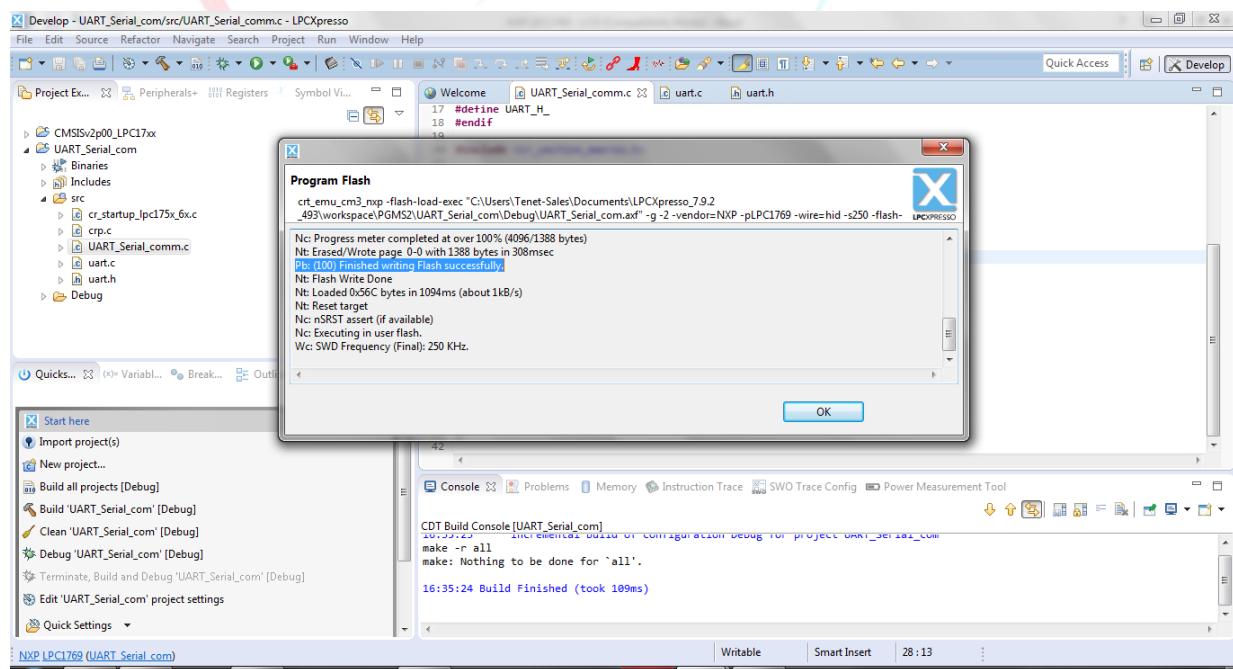


Figure 21

Hardware Requirements:

- LPC1769 Board
- Jumper wires

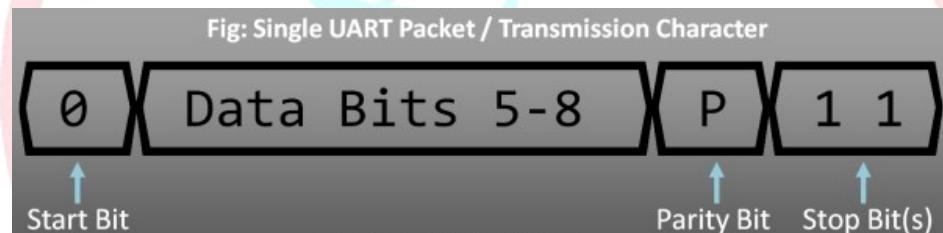
UART basics:

Whenever we want to communicate between PC and MCU or between two MCUs, the simplest way to achieve that is using UART. UART stands for Universal Asynchronous Receiver/Transmitter.

UART communication basically uses 2 pins for Data transfer and these are:

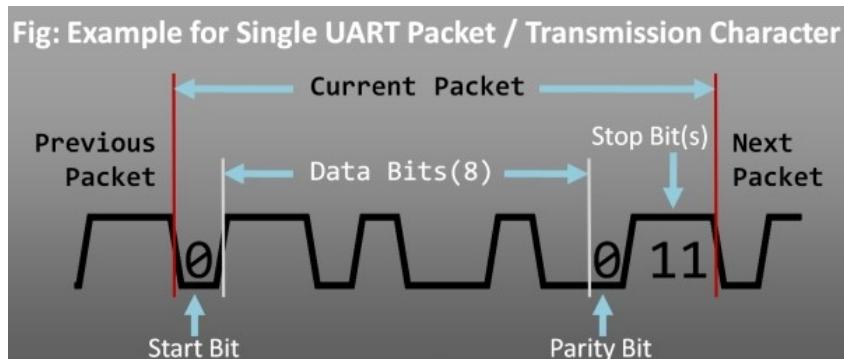
1. TxD (or Tx) – which is the Transmit Data Pin used to send data
2. RxD (or Rx) – which is the Receive Data Pin used to get data

UART sends & receives data in form of chunks or packets. These chunks or packets are also referred to as ‘transmission characters’. The structure of a data packets is as shown below:

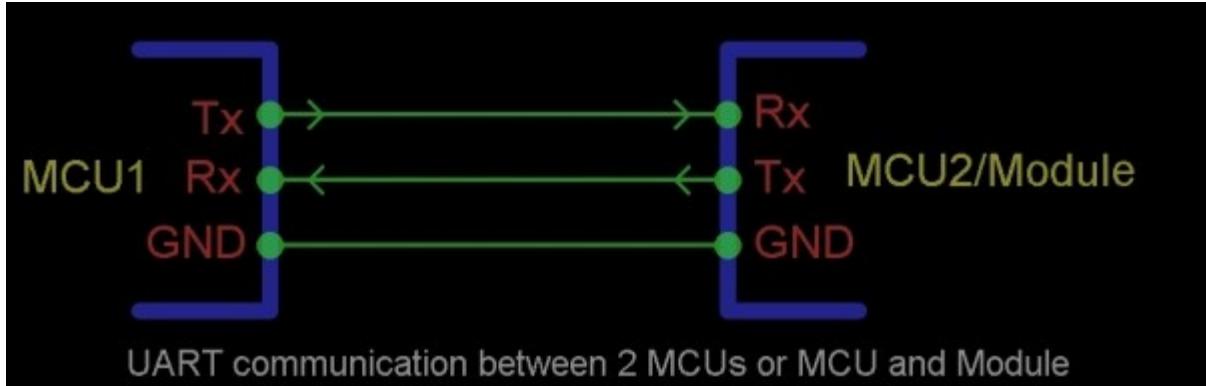


UART data packet begins with a ‘0’. This bit is also called as “Start Bit” which signifies incoming data. Next comes the actual data which can be 5 to 8 bits in length. After the data an optional parity bit can be used for error checking. Lastly comes the “Stop Bit(s)” which is a ‘1’ and this signifies end of current data packet. Note that either 1 or 2 stop bits can be used and the parity bit can be : Even , Odd , forced to 1 i.e. Mark parity , forced to 0 i.e. Space parity or None. (In UART/RS232 context a MARK means 1 and SPACE means 0, hence marking state means a stream (series) of 1s and Spacing state means a stream of 0s)

Here is an example of a packet having 8 data bits, odd parity bit and 2 stop bits:



1. UART communication between 2 MCUs or MCU and Module



OUTPUT:

UART communication between two micro-controllers, LPC1769 and FreeSoC2 (PSoC5LP).

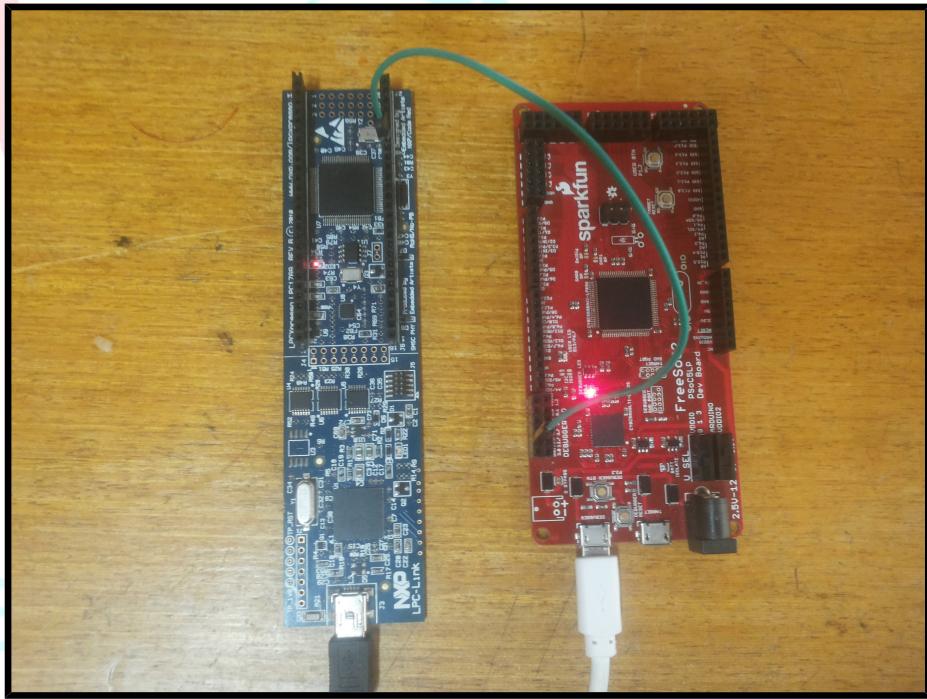
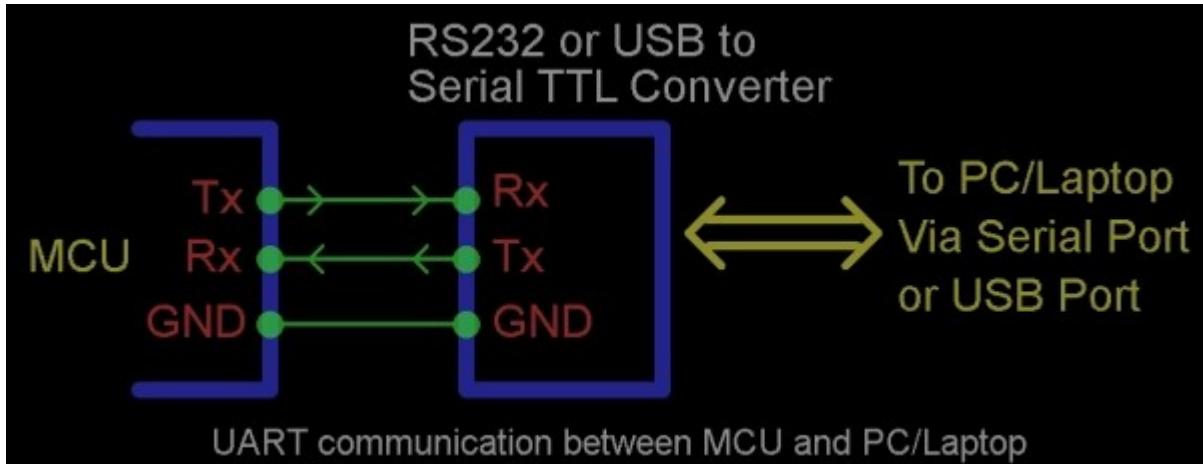


Figure 18

NOTE: To have Serial Communication between two microcontrollers, we have to program the second microcontroller to receive the transmitted signal.

2. UART communication between MCUs or Modules and PC/Laptop



USB to Serial TTL converter (Prolific cable)

GREEN - TX

WHITE - RX

RED - No Connection

BLACK - GND



The output can be viewed in a Serial monitors like HyperTerminal, Putty, XCTU and so on...

TENET
TECHNETRONICS

For product link:

1. <http://www.tenettech.com/product/1548/lpc1769-lpcxpresso-board>
2. <http://www.tenettech.com/product/7241/freesoc2-development-board-psoc5lp>

For more information please visit: www.tenettech.com

For technical query please send an e-mail: info@tenettech.com



TENET
TECHNETRONICS