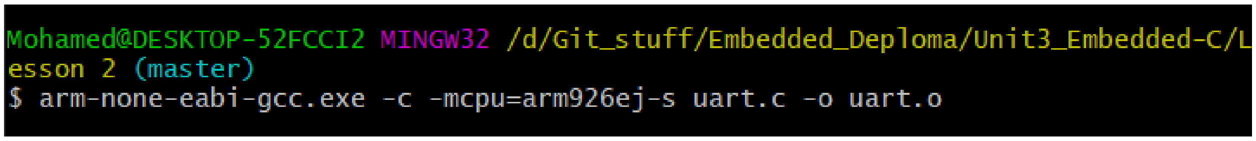
LAB 1

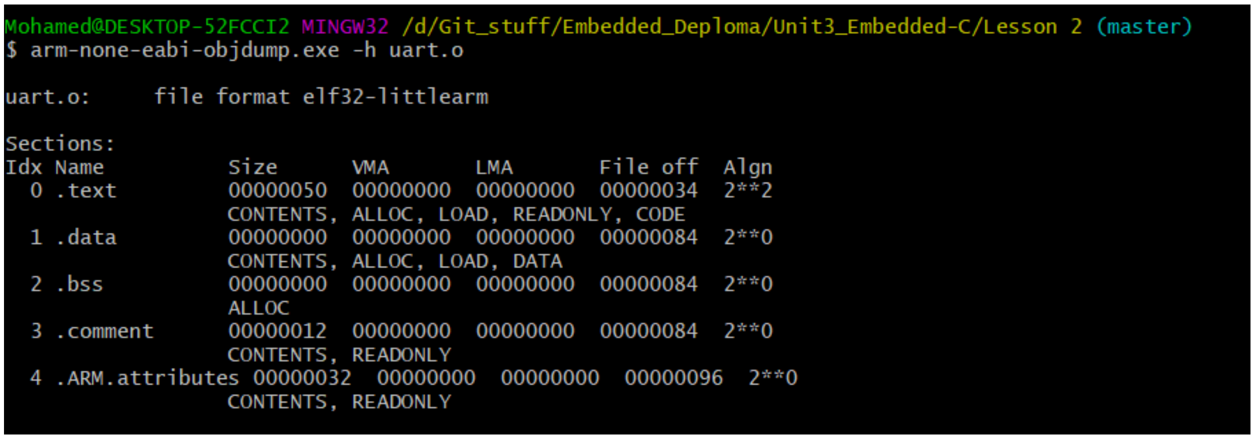
Using VersatilePB virtual board in QEMU and ARM toolchain

1. Getting .obj files from source files and analyzing the sections :

Uart.c file

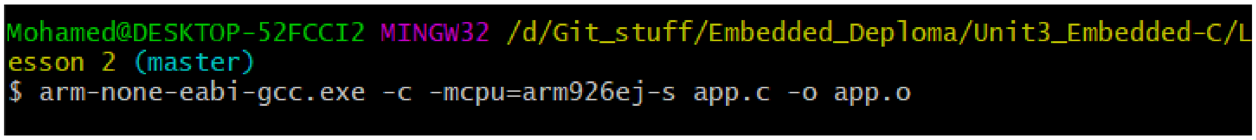


```
Mohamed@DESKTOP-52FCCI2 MINGW32 /d/Git_stuff/Embedded_Deploma/Unit3_Embedded-C/L
esson 2 (master)
$ arm-none-eabi-gcc.exe -c -mcpu=arm926ej-s uart.c -o uart.o
```

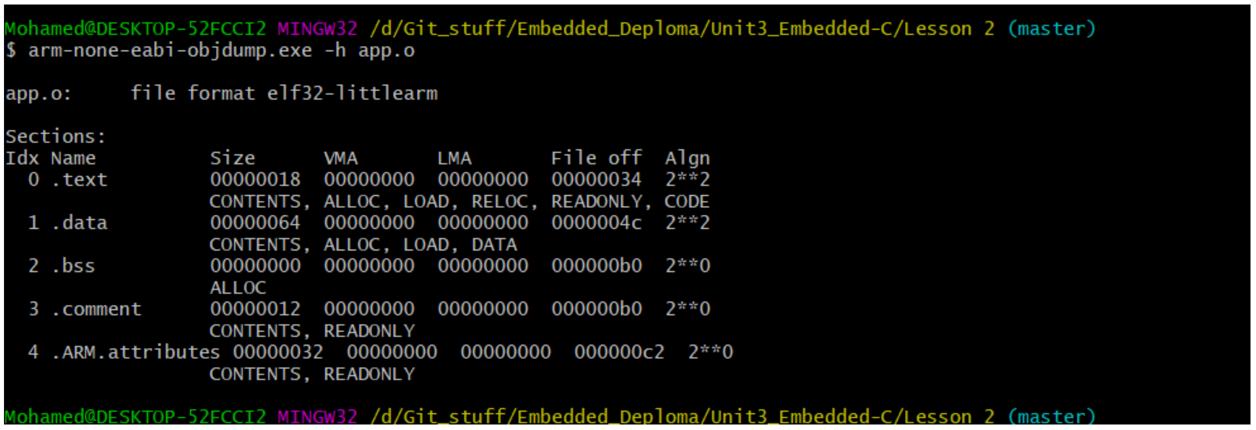


```
Mohamed@DESKTOP-52FCCI2 MINGW32 /d/Git_stuff/Embedded_Deploma/Unit3_Embedded-C/Lesson 2 (master)
$ arm-none-eabi-objdump.exe -h uart.o

uart.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000050  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .data         00000000  00000000  00000000  00000084  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  00000084  2**0
                  ALLOC
  3 .comment      00000012  00000000  00000000  00000084  2**0
                  CONTENTS, READONLY
  4 .ARM.attributes 00000032  00000000  00000000  00000096  2**0
                  CONTENTS, READONLY
```

App.c file

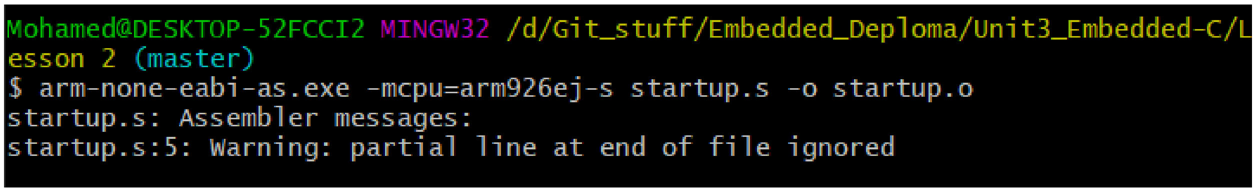


```
Mohamed@DESKTOP-52FCCI2 MINGW32 /d/Git_stuff/Embedded_Deploma/Unit3_Embedded-C/L
esson 2 (master)
$ arm-none-eabi-gcc.exe -c -mcpu=arm926ej-s app.c -o app.o
```

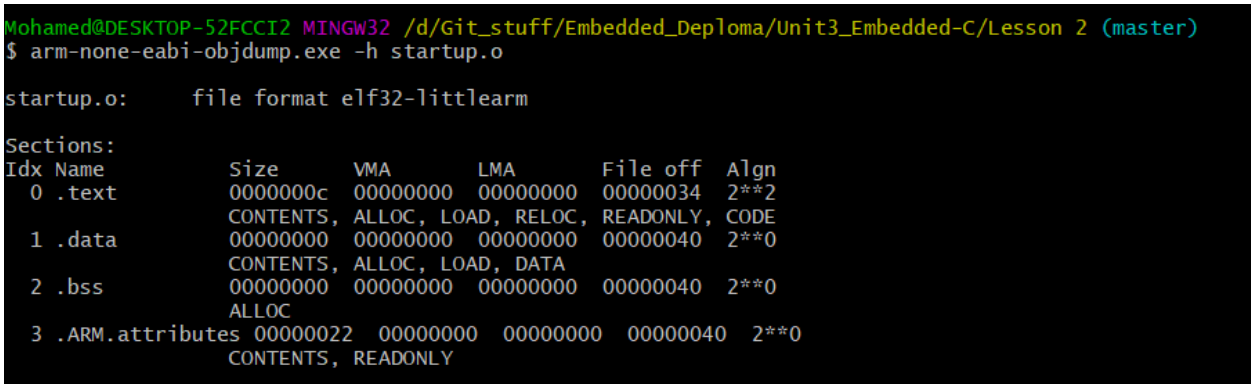


```
Mohamed@DESKTOP-52FCCI2 MINGW32 /d/Git_stuff/Embedded_Deploma/Unit3_Embedded-C/Lesson 2 (master)
$ arm-none-eabi-objdump.exe -h app.o

app.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000018  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000064  00000000  00000000  0000004c  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  000000b0  2**0
                  ALLOC
  3 .comment      00000012  00000000  00000000  000000b0  2**0
                  CONTENTS, READONLY
  4 .ARM.attributes 00000032  00000000  00000000  000000c2  2**0
                  CONTENTS, READONLY

Mohamed@DESKTOP-52FCCI2 MINGW32 /d/Git_stuff/Embedded_Deploma/Unit3_Embedded-C/Lesson 2 (master)
```

Startup.s file

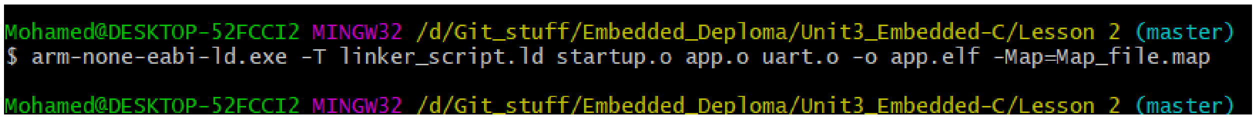


```
Mohamed@DESKTOP-52FCCI2 MINGW32 /d/Git_stuff/Embedded_Deploma/Unit3_Embedded-C/L
esson 2 (master)
$ arm-none-eabi-as.exe -mcpu=arm926ej-s startup.s -o startup.o
startup.s: Assembler messages:
startup.s:5: Warning: partial line at end of file ignored
```

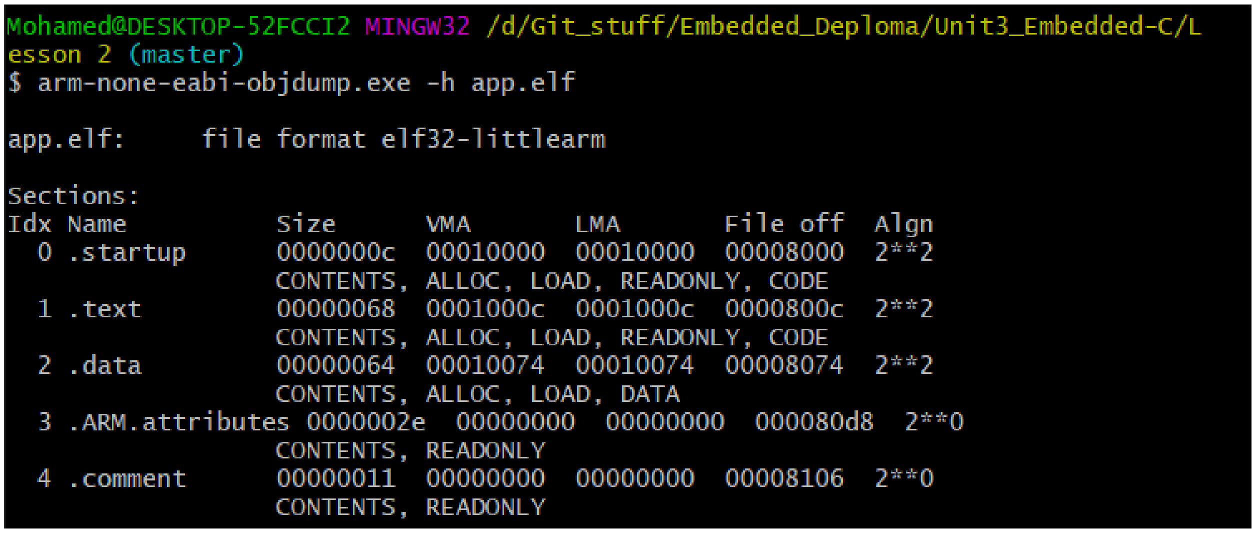


```
Mohamed@DESKTOP-52FCCI2 MINGW32 /d/Git_stuff/Embedded_Deploma/Unit3_Embedded-C/Lesson 2 (master)
$ arm-none-eabi-objdump.exe -h startup.o

startup.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         0000000c  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  00000000  00000000  00000040  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  00000040  2**0
                  ALLOC
  3 .ARM.attributes 00000022  00000000  00000000  00000040  2**0
                  CONTENTS, READONLY
```

2. Linking .obj files to get .elf file and analyzing the sections :

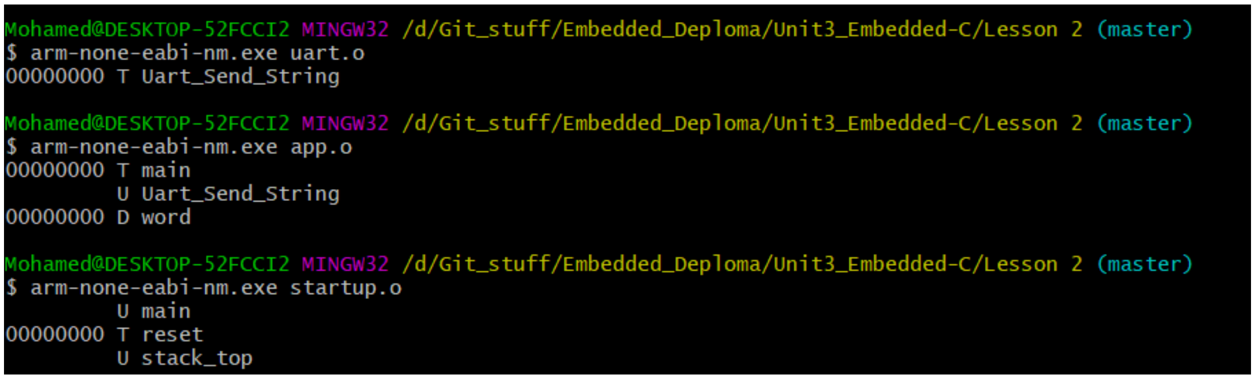


```
Mohamed@DESKTOP-52FCCI2 MINGW32 /d/Git_stuff/Embedded_Deploma/Unit3_Embedded-C/Lesson 2 (master)
$ arm-none-eabi-ld.exe -T linker_script.ld startup.o app.o uart.o -o app.elf -Map=Map_file.map

Mohamed@DESKTOP-52FCCI2 MINGW32 /d/Git_stuff/Embedded_Deploma/Unit3_Embedded-C/Lesson 2 (master)
```

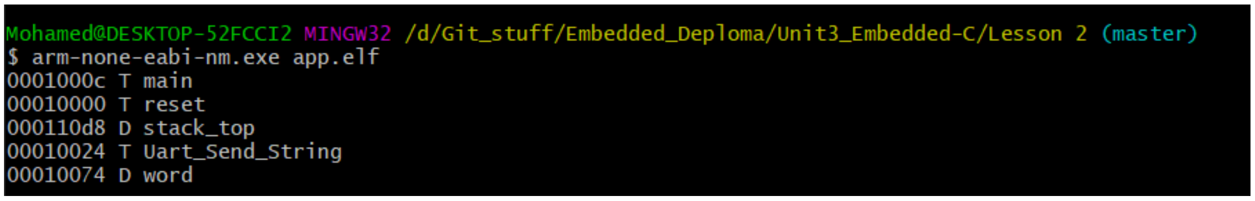


```
Mohamed@DESKTOP-52FCCI2 MINGW32 /d/Git_stuff/Embedded_Deploma/Unit3_Embedded-C/L
esson 2 (master)
$ arm-none-eabi-objdump.exe -h app.elf

app.elf:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .startup      0000000c  00010000  00010000  00008000  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .text         00000068  0001000c  0001000c  0000800c  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  2 .data         00000064  00010074  00010074  00008074  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  3 .ARM.attributes 0000002e  00000000  00000000  000080d8  2**0
                  CONTENTS, READONLY
  4 .comment      00000011  00000000  00000000  00008106  2**0
                  CONTENTS, READONLY
```

3. Analyzing the symbol table for .obj files and .elf file :

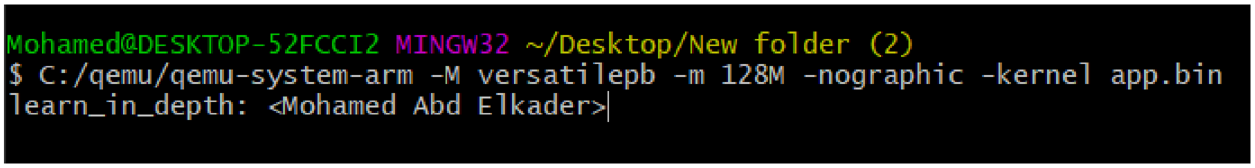


```
Mohamed@DESKTOP-52FCCI2 MINGW32 /d/Git_stuff/Embedded_Deploma/Unit3_Embedded-C/Lesson 2 (master)
$ arm-none-eabi-nm.exe uart.o
00000000 T Uart_Send_String

Mohamed@DESKTOP-52FCCI2 MINGW32 /d/Git_stuff/Embedded_Deploma/Unit3_Embedded-C/Lesson 2 (master)
$ arm-none-eabi-nm.exe app.o
00000000 T main
         U Uart_Send_String
00000000 D word

Mohamed@DESKTOP-52FCCI2 MINGW32 /d/Git_stuff/Embedded_Deploma/Unit3_Embedded-C/Lesson 2 (master)
$ arm-none-eabi-nm.exe startup.o
         U main
00000000 T reset
         U stack_top
```



```
Mohamed@DESKTOP-52FCCI2 MINGW32 /d/Git_stuff/Embedded_Deploma/Unit3_Embedded-C/Lesson 2 (master)
$ arm-none-eabi-nm.exe app.elf
0001000c T main
00010000 T reset
000110d8 D stack_top
00010024 T Uart_Send_String
00010074 D word
```

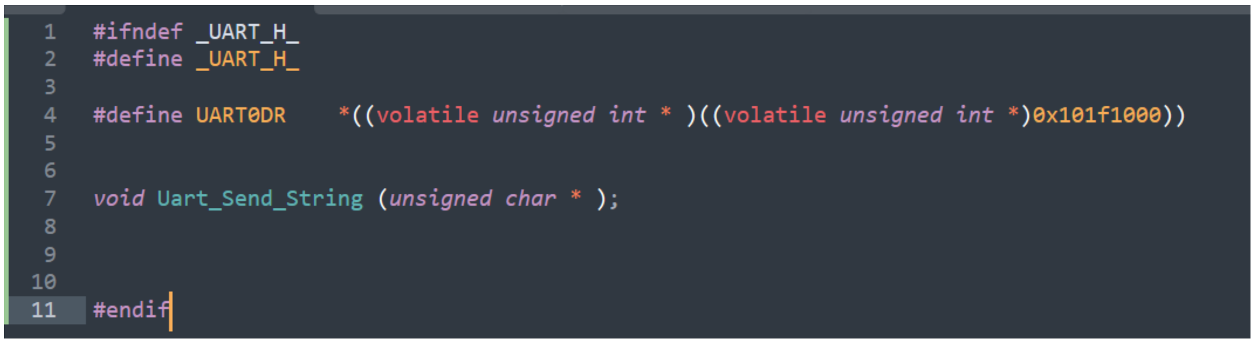
4. **Run the final .bin file on QEMU and get the final output :**



```
Mohamed@DESKTOP-52FCCI2 MINGW32 ~/Desktop/New folder (2)
$ C:/qemu/qemu-system-arm -M versatilepb -m 128M -nographic -kernel app.bin
learn_in_depth: <Mohamed Abd Elkader>
```

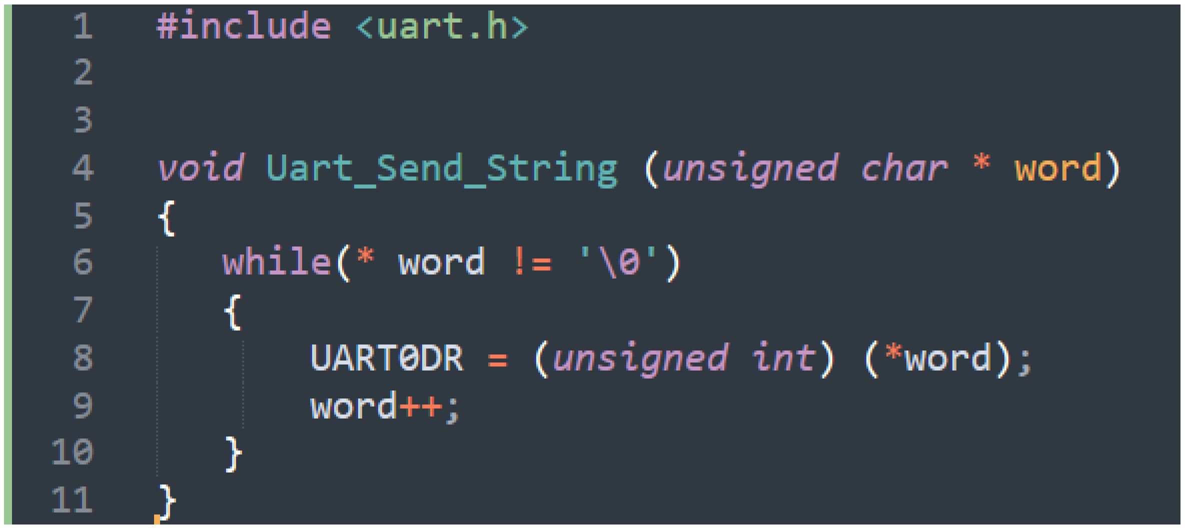
The source files Shots:

Uart.h



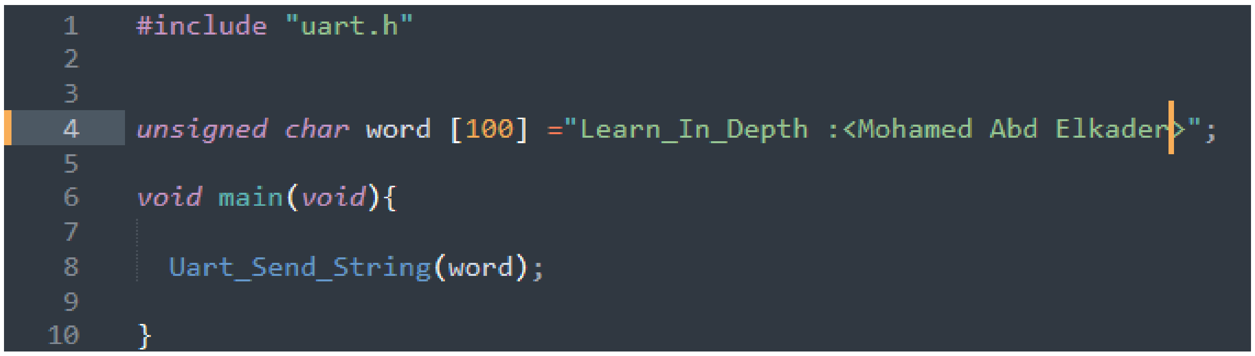
```c
#ifndef _UART_H_
#define _UART_H_

#define UART0DR    *((volatile unsigned int * )((volatile unsigned int *)0x101f1000))


void Uart_Send_String (unsigned char * );



#endif
```

Uart.c



```c
#include <uart.h>


void Uart_Send_String (unsigned char * word)
{
    while(* word != '\0')
    {
        UART0DR = (unsigned int) (*word);
        word++;
    }
}
```

App.c



```c
#include "uart.h"


unsigned char word [100] ="Learn_In_Depth :<Mohamed Abd Elkader>";

void main(void){

 Uart_Send_String(word);

}
```