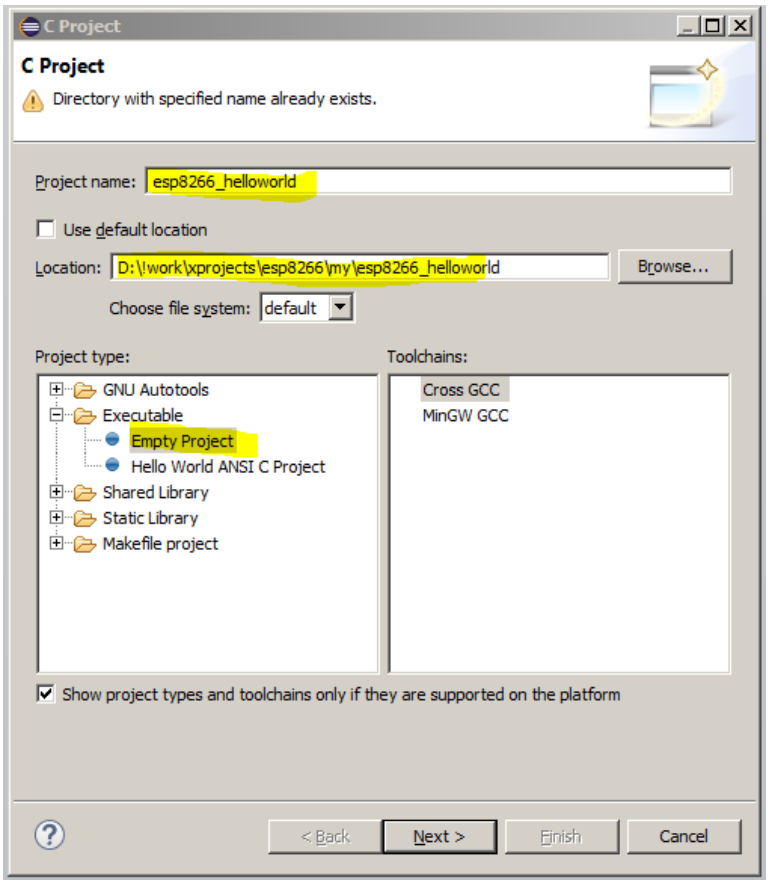
File -> New -> C project

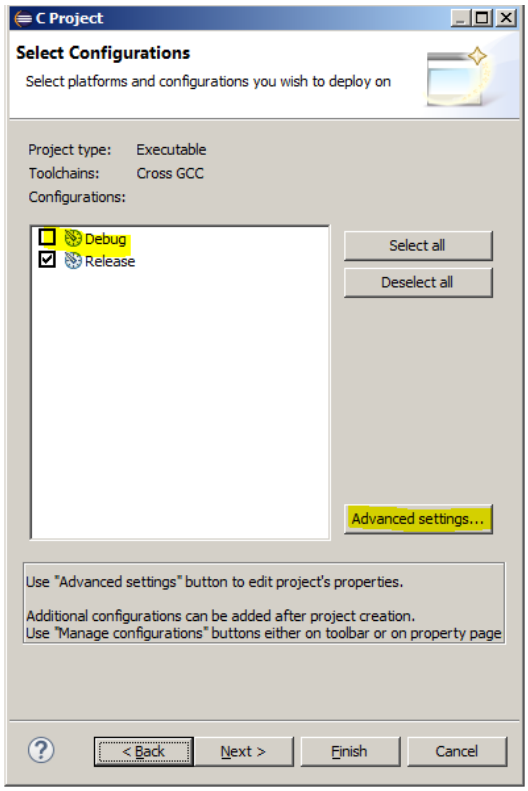



Give your project a name

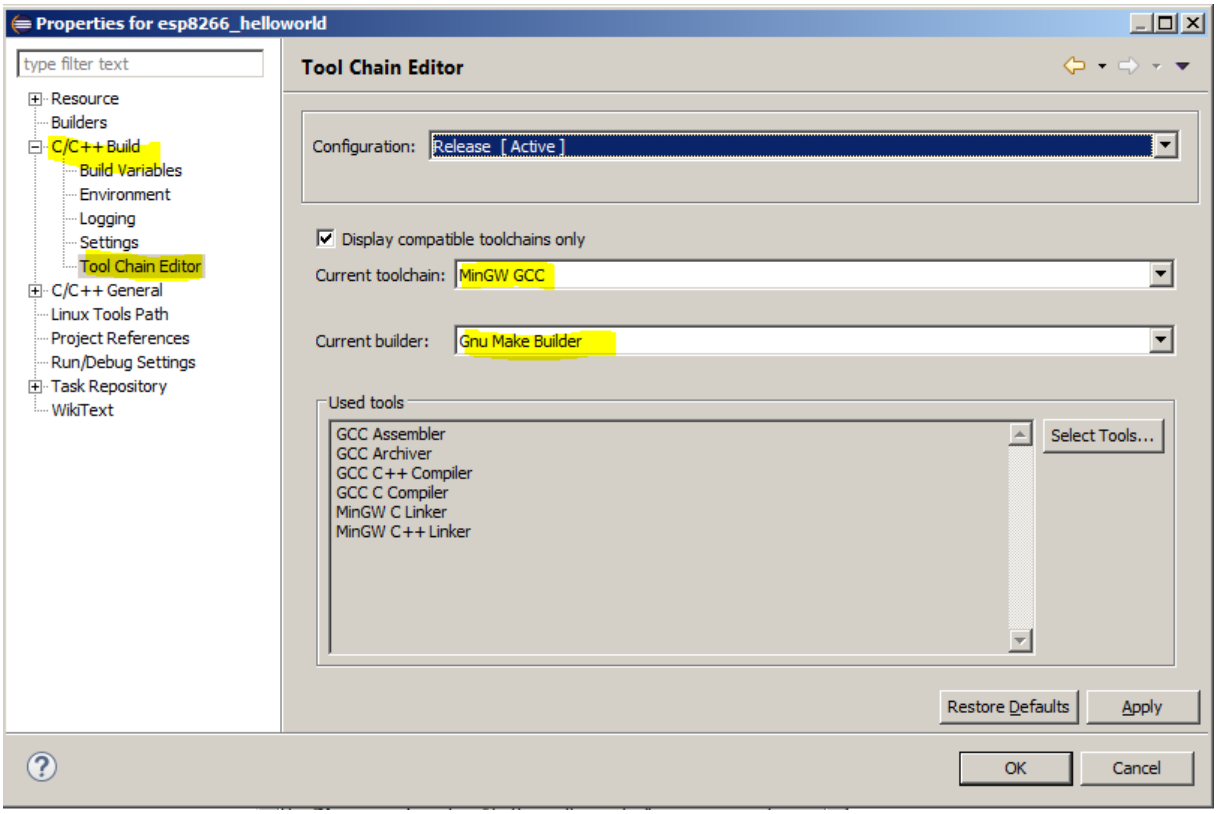
Specify folder for project

Select Empty project type

**Click Next**

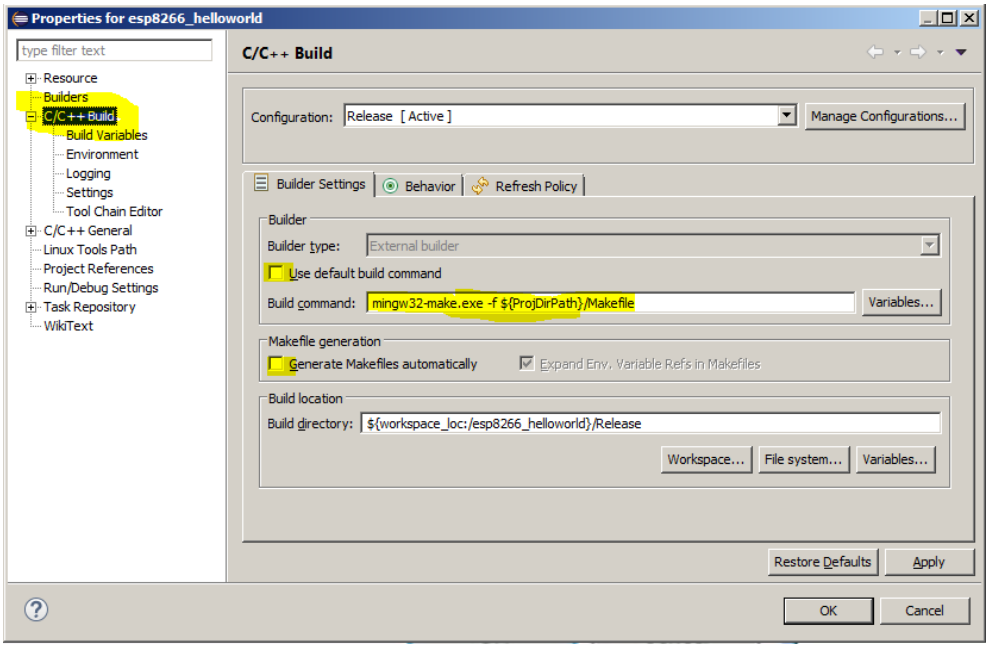



Uncheck Debug and **click Advanced Settings**




Select C/C++ Build -> Tool Chain Editor and select MinGW GCC toolcnain and Gnu Make Builder

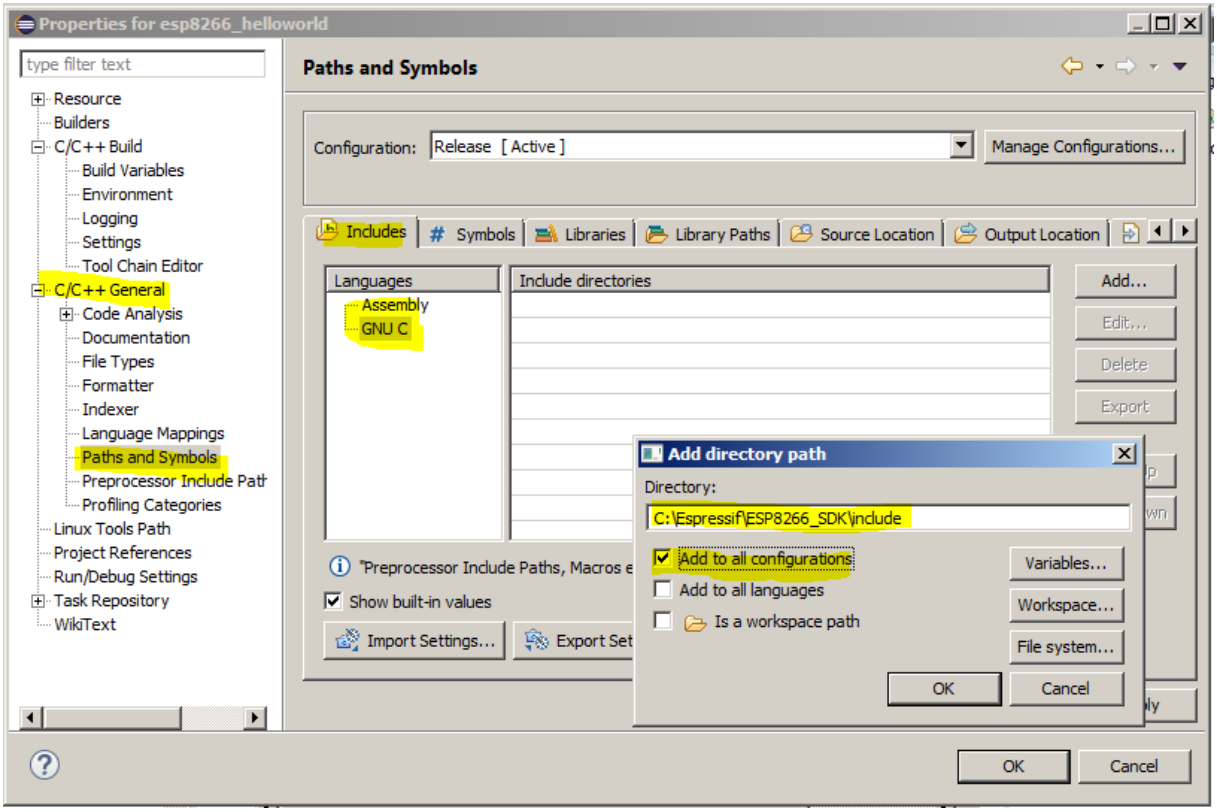
Click **Apply**




Select C/C++ Build and uncheck »Use default build command«

Enter custom build command: mingw32-make.exe -f ${ProjDirPath}/Makefile

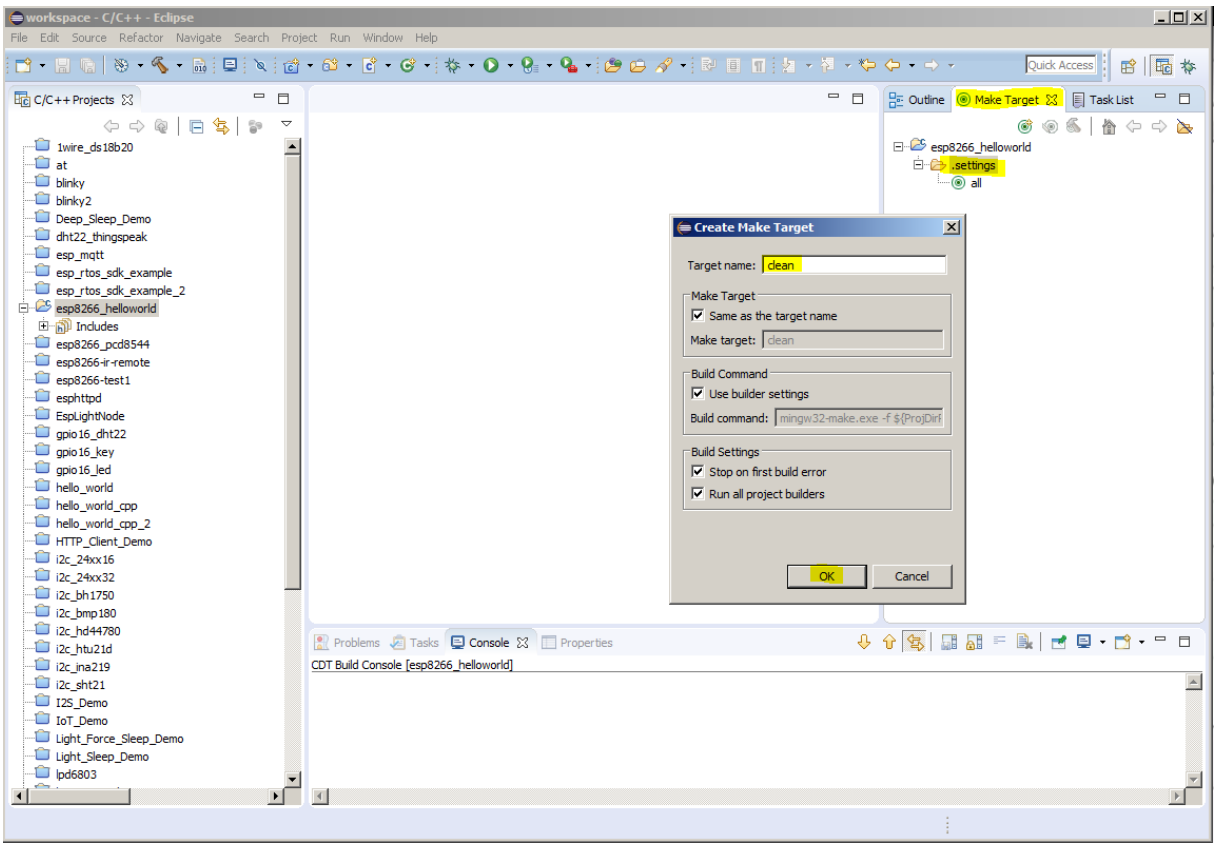
Uncheck »Generate Makefiles automatically«




Select C/C++ General, Paths and Symbols, Select tab »Includes« and click **Add**. Point to C:\Espressif\ESP8266_SDK\include and check »Add to all configurations«

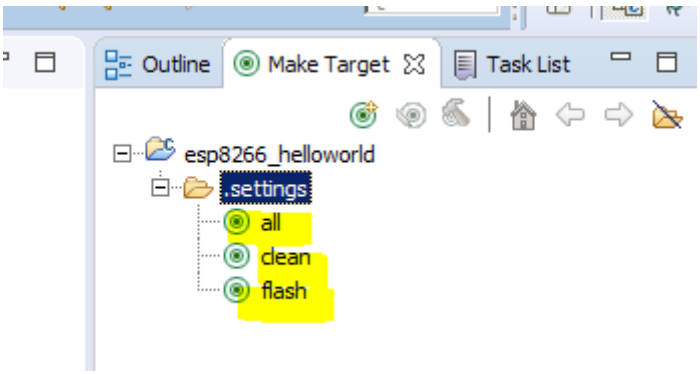
Click **OK**, **OK** and **Finish**.

In workspace there is pane titled »Make target«. Click on your project name and then on .settings:

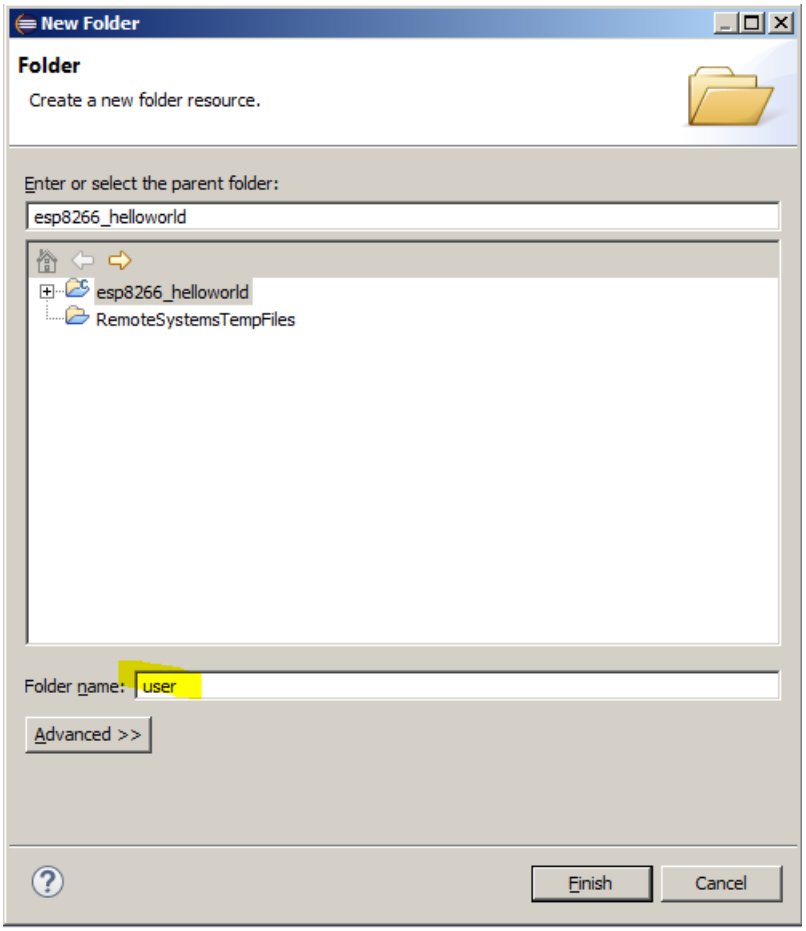



Click on green icon »Create Make Target« and create three new targets:

- all
- clean
- flash

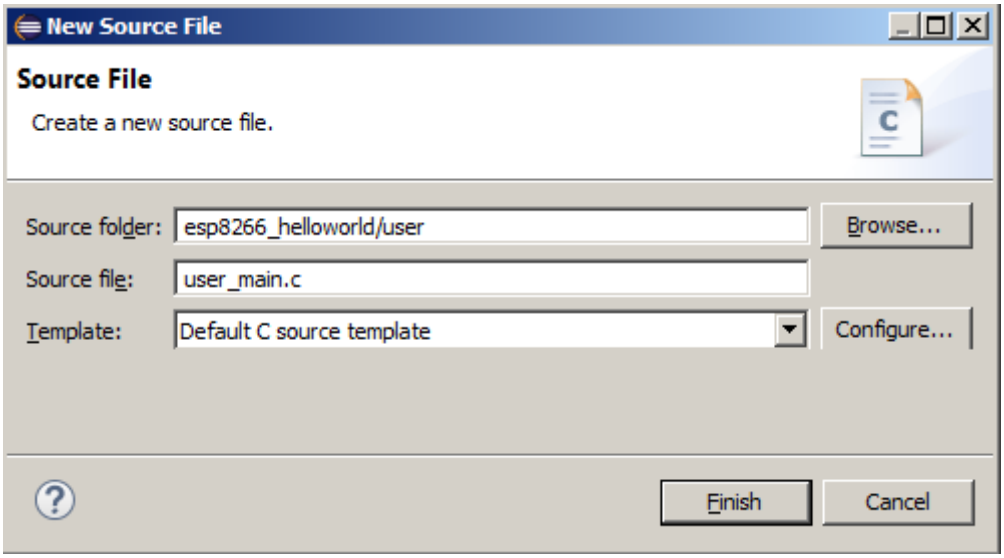


Rightclick on the project in the »Projects« pane on the left and add new folder »user«:




Click **Finish**

Add  new source file to this folder:



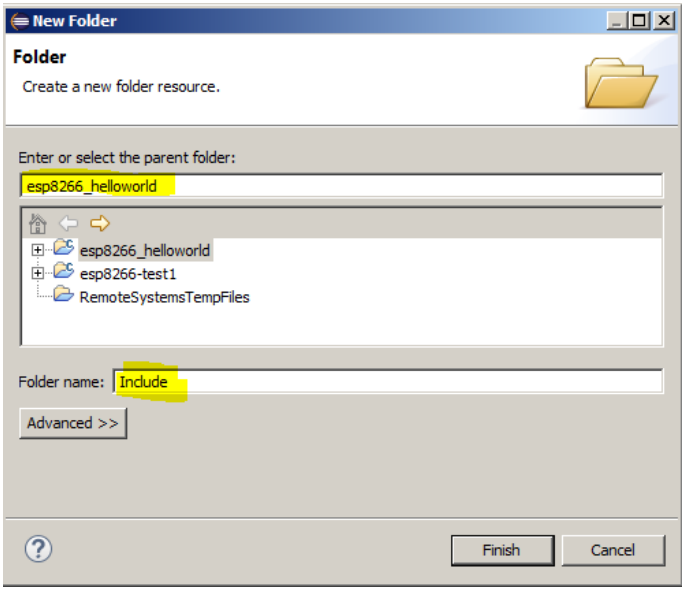

enter file name user_main.c

Click **Finish**.

Write (or copy/paste) »hello world« source code:

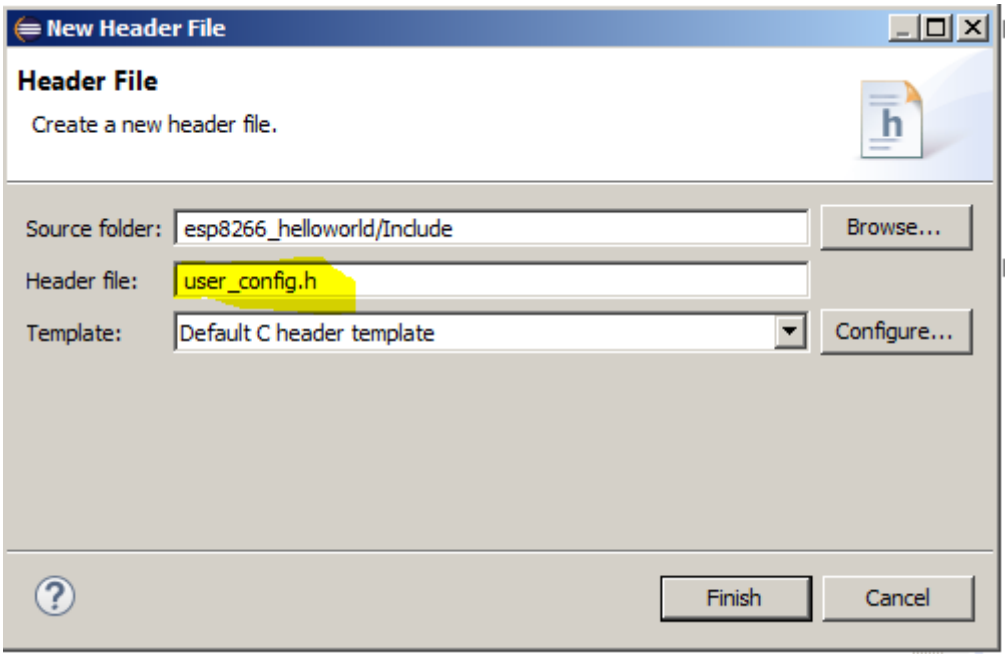
```
/*
 * user_main.c
 *
 *  Created on: 16. sep. 2017
 *      Author: Marko
 */

#include <ets_sys.h>
#include <osapi.h>
#include "user_interface.h"
#include <os_type.h>
#include <gpio.h>
#include "driver/uart.h"

#define DELAY 1000 /* milliseconds */

LOCAL os_timer_t hello_timer;
extern int ets_uart_printf(const char *fmt, ...);

LOCAL void ICACHE_FLASH_ATTR hello_cb(void *arg)
{
       ets_uart_printf("Hello World!\r\n");
}

uint32 ICACHE_FLASH_ATTR user_rf_cal_sector_set(void)
{
    enum flash_size_map size_map = system_get_flash_size_map();
    uint32 rf_cal_sec = 0;

    switch (size_map) {
        case FLASH_SIZE_4M_MAP_256_256:
            rf_cal_sec = 128 - 8;
            break;

        case FLASH_SIZE_8M_MAP_512_512:
            rf_cal_sec = 256 - 5;
            break;

        case FLASH_SIZE_16M_MAP_512_512:
        case FLASH_SIZE_16M_MAP_1024_1024:
            rf_cal_sec = 512 - 5;
            break;

        case FLASH_SIZE_32M_MAP_512_512:
        case FLASH_SIZE_32M_MAP_1024_1024:
            rf_cal_sec = 1024 - 5;
            break;

        default:
            rf_cal_sec = 0;
            break;
    }

    return rf_cal_sec;
}

void ICACHE_FLASH_ATTR user_rf_pre_init(void)
{
}

void ICACHE_FLASH_ATTR user_init(void)
{
       // Configure the UART
       uart_init(BIT_RATE_115200, BIT_RATE_115200);
       // Set up a timer to send the message
       // os_timer_disarm(ETSTimer *ptimer)
       os_timer_disarm(&hello_timer);
       // os_timer_setfn(ETSTimer *ptimer, ETSTimerFunc *pfunction, void *parg)
       os_timer_setfn(&hello_timer, (os_timer_func_t *)hello_cb, (void *)0);
       // void os_timer_arm(ETSTimer *ptimer,uint32_t milliseconds, bool repeat_flag)
       os_timer_arm(&hello_timer, DELAY, 1);
}
```

Create new project folder »Include«:



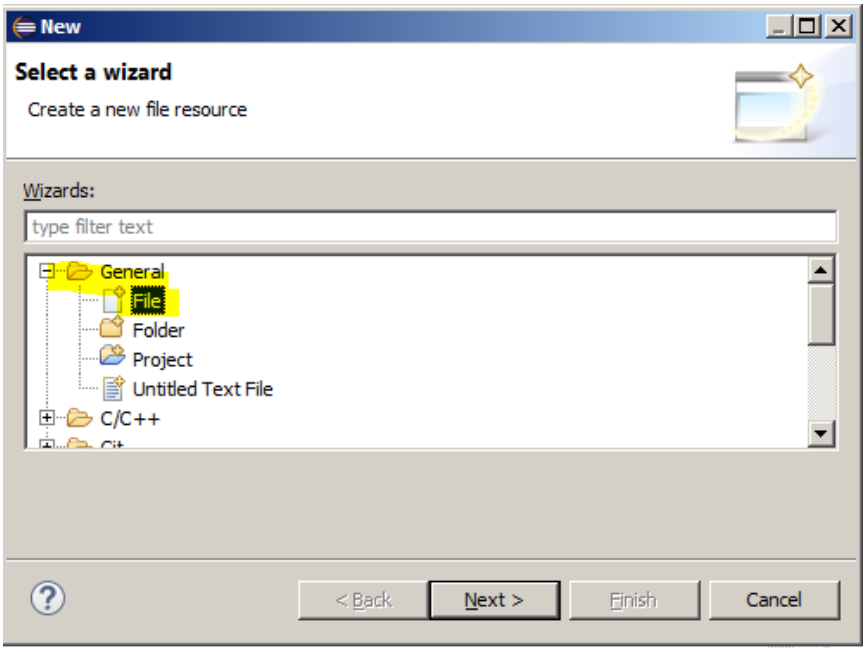

Click **Finish**.

Add new header file to the folder »Include«, name it user_config.h:

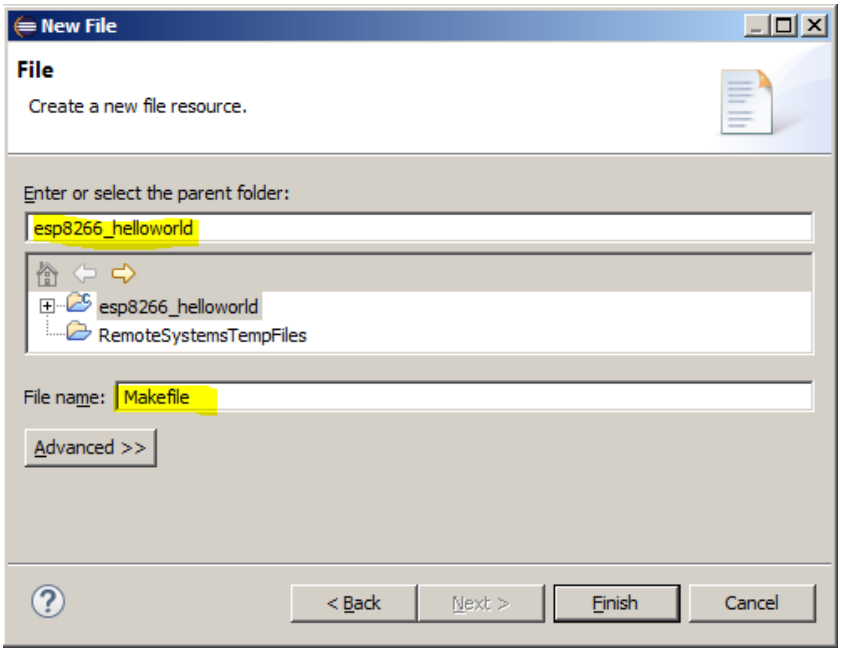



Click **Finish**.

Rightclick on project in a project tree and select New->Other, then select General->File and click **Next**.




Enter filename »Makefile«:




Click **Finish**.

Paste this make file:

```
# Main settings -------------
# build directory
BUILD_BASE    = build

# firmware directory
FW_BASE             = firmware

# name for the target project
TARGET       = app

# Base directory for the compiler
XTENSA_TOOLS_ROOT = c:/Espressif/xtensa-lx106-elf/bin

# base directory of the ESP8266 SDK package, absolute
SDK_BASE     = c:/Espressif/ESP8266_SDK
SDK_TOOLS    = c:/Espressif/utils/ESP8266

# Extra libs, include and ld file
EXTRA_BASE   = c:/Espressif/extra

# esptool path and port
ESPTOOL             = $(SDK_TOOLS)/esptool.exe
ESPPORT             = COM6

# Baud rate for programmer
ESPBAUD             = 115200

# BOOT = none
# BOOT = old - boot_v1.1
# BOOT = new - boot_v1.2+
BOOT = none
# APP = 0 - eagle.flash.bin + eagle.irom0text.bin
# APP = 1 - user1.bin
# APP = 2 - user2.bin
APP = 0
# SPI_SPEED = 40, 26, 20, 80
SPI_SPEED = 40
# SPI_MODE: QIO, QOUT, DIO, DOUT
SPI_MODE = QIO
# SPI_SIZE_MAP
# 0 : 512 KB (256 KB + 256 KB)
# 1 : 256 KB
# 2 : 1024 KB (512 KB + 512 KB)
# 3 : 2048 KB (512 KB + 512 KB)
# 4 : 4096 KB (512 KB + 512 KB)
# 5 : 2048 KB (1024 KB + 1024 KB)
# 6 : 4096 KB (1024 KB + 1024 KB)
SPI_SIZE_MAP = 4


# Basic project settings
MODULES      = driver user
LIBS   = c gcc hal phy pp net80211 lwip wpa main crypto

# Root make -------------
############################################################
#
```

```
# Root Level Makefile
#
# Version 2.0
#
# (c) by CHERTS <sleuthhound@gmail.com>
#
#############################################################

ifeq ($(BOOT), new)
    boot = new
else
    ifeq ($(BOOT), old)
        boot = old
    else
        boot = none
    endif
endif

ifeq ($(APP), 1)
    app = 1
else
    ifeq ($(APP), 2)
        app = 2
    else
        app = 0
    endif
endif

ifeq ($(SPI_SPEED), 26.7)
    freqdiv = 1
    flashimageoptions = -ff 26m
else
    ifeq ($(SPI_SPEED), 20)
        freqdiv = 2
        flashimageoptions = -ff 20m
    else
        ifeq ($(SPI_SPEED), 80)
            freqdiv = 15
            flashimageoptions = -ff 80m
        else
            freqdiv = 0
            flashimageoptions = -ff 40m
        endif
    endif
endif

ifeq ($(SPI_MODE), QOUT)
    mode = 1
    flashimageoptions += -fm qout
else
    ifeq ($(SPI_MODE), DIO)
      mode = 2
      flashimageoptions += -fm dio
    else
        ifeq ($(SPI_MODE), DOUT)
            mode = 3
            flashimageoptions += -fm dout
        else
            mode = 0
```

```
            flashimageoptions += -fm qio
        endif
    endif
endif

addr = 0x01000

ifeq ($(SPI_SIZE_MAP), 1)
  size_map = 1
  flash = 256
  flashimageoptions += -fs 2m
else
  ifeq ($(SPI_SIZE_MAP), 2)
    size_map = 2
    flash = 1024
    flashimageoptions += -fs 8m
    ifeq ($(app), 2)
      addr = 0x81000
    endif
  else
    ifeq ($(SPI_SIZE_MAP), 3)
      size_map = 3
      flash = 2048
      flashimageoptions += -fs 16m
      ifeq ($(app), 2)
        addr = 0x81000
      endif
    else
      ifeq ($(SPI_SIZE_MAP), 4)
       size_map = 4
       flash = 4096
       flashimageoptions += -fs 32m
        ifeq ($(app), 2)
          addr = 0x81000
        endif
      else
        ifeq ($(SPI_SIZE_MAP), 5)
          size_map = 5
          flash = 2048
          flashimageoptions += -fs 16m
          ifeq ($(app), 2)
            addr = 0x101000
          endif
        else
          ifeq ($(SPI_SIZE_MAP), 6)
            size_map = 6
            flash = 4096
            flashimageoptions += -fs 32m
            ifeq ($(app), 2)
              addr = 0x101000
            endif
          else
            size_map = 0
            flash = 512
            flashimageoptions += -fs 4m
            ifeq ($(app), 2)
              addr = 0x41000
            endif
          endif
```

```
        endif
      endif
    endif
  endif
endif

EXTRA_INCDIR    = include $(EXTRA_BASE)/include

# compiler flags using during compilation of source files
CFLAGS       = -Os -g -O2 -std=gnu90 -Wpointer-arith -Wundef -Werror -Wl,-EL -fno-
inline-functions -nostdlib -mlongcalls -mtext-section-literals -mno-serialize-
volatile -D__ets__ -DICACHE_FLASH
CXXFLAGS     = -Os -g -O2 -Wpointer-arith -Wundef -Werror -Wl,-EL -fno-inline-
functions -nostdlib -mlongcalls -mtext-section-literals -mno-serialize-volatile -
D__ets__ -DICACHE_FLASH -fno-rtti -fno-exceptions

# linker flags used to generate the main object file
LDFLAGS            = -nostdlib -Wl,--no-check-sections -u call_user_start -Wl,-
static

# linker script used for the above linkier step
LD_SCRIPT    = eagle.app.v6.ld

ifneq ($(boot), none)
ifneq ($(app),0)
    ifeq ($(size_map), 6)
      LD_SCRIPT = eagle.app.v6.$(boot).2048.ld
    else
      ifeq ($(size_map), 5)
        LD_SCRIPT = eagle.app.v6.$(boot).2048.ld
      else
        ifeq ($(size_map), 4)
          LD_SCRIPT = eagle.app.v6.$(boot).1024.app$(app).ld
        else
          ifeq ($(size_map), 3)
            LD_SCRIPT = eagle.app.v6.$(boot).1024.app$(app).ld
          else
            ifeq ($(size_map), 2)
              LD_SCRIPT = eagle.app.v6.$(boot).1024.app$(app).ld
            else
              ifeq ($(size_map), 0)
                LD_SCRIPT = eagle.app.v6.$(boot).512.app$(app).ld
              endif
            endif
             endif
           endif
         endif
       endif
       BIN_NAME = user$(app).$(flash).$(boot).$(size_map)
       CFLAGS += -DAT_UPGRADE_SUPPORT
endif
else
    app = 0
endif

# various paths from the SDK used in this project
SDK_LIBDIR   = lib
SDK_LDDIR    = ld
SDK_INCDIR   = include include/json
```

```
# select which tools to use as compiler, librarian and linker
CC           := $(XTENSA_TOOLS_ROOT)/xtensa-lx106-elf-gcc
CXX          := $(XTENSA_TOOLS_ROOT)/xtensa-lx106-elf-g++
AR           := $(XTENSA_TOOLS_ROOT)/xtensa-lx106-elf-ar
LD           := $(XTENSA_TOOLS_ROOT)/xtensa-lx106-elf-gcc
OBJCOPY             := $(XTENSA_TOOLS_ROOT)/xtensa-lx106-elf-objcopy
OBJDUMP             := $(XTENSA_TOOLS_ROOT)/xtensa-lx106-elf-objdump

SRC_DIR             := $(MODULES)
BUILD_DIR    := $(addprefix $(BUILD_BASE)/,$(MODULES))

SDK_LIBDIR   := $(addprefix $(SDK_BASE)/,$(SDK_LIBDIR))
SDK_INCDIR   := $(addprefix -I$(SDK_BASE)/,$(SDK_INCDIR))

SRC          := $(foreach sdir,$(SRC_DIR),$(wildcard $(sdir)/*.c*))
C_OBJ        := $(patsubst %.c,%.o,$(SRC))
CXX_OBJ             := $(patsubst %.cpp,%.o,$(C_OBJ))
OBJ          := $(patsubst %.o,$(BUILD_BASE)/%.o,$(CXX_OBJ))
LIBS         := $(addprefix -l,$(LIBS))
APP_AR       := $(addprefix $(BUILD_BASE)/,$(TARGET)_app.a)
TARGET_OUT   := $(addprefix $(BUILD_BASE)/,$(TARGET).out)

LD_SCRIPT    := $(addprefix -T$(SDK_BASE)/$(SDK_LDDIR)/,$(LD_SCRIPT))

INCDIR       := $(addprefix -I,$(SRC_DIR))
EXTRA_INCDIR := $(addprefix -I,$(EXTRA_INCDIR))
MODULE_INCDIR       := $(addsuffix /include,$(INCDIR))

V ?= $(VERBOSE)
ifeq ("$(V)","1")
Q :=
vecho := @true
else
Q := @
vecho := @echo
endif

vpath %.c $(SRC_DIR)
vpath %.cpp $(SRC_DIR)

define compile-objects
$1/%.o: %.c
	$(vecho) "CC $$<"
	$(Q) $(CC) $(INCDIR) $(MODULE_INCDIR) $(EXTRA_INCDIR) $(SDK_INCDIR)
$(CFLAGS)  -c $$< -o $$@
$1/%.o: %.cpp
	$(vecho) "C+ $$<"
	$(Q) $(CXX) $(INCDIR) $(MODULE_INCDIR) $(EXTRA_INCDIR) $(SDK_INCDIR)
$(CXXFLAGS)  -c $$< -o $$@
endef

.PHONY: all checkdirs clean flash flashboot flashinit rebuild

all: checkdirs $(TARGET_OUT)

$(TARGET_OUT): $(APP_AR)
	$(vecho) "LD $@"
```

```
	$(Q) $(LD) -L$(SDK_LIBDIR) $(LD_SCRIPT) $(LDFLAGS) -Wl,--start-group $(LIBS)
$(APP_AR) -Wl,--end-group -o $@
	$(vecho) "Run objcopy, please wait..."
	$(Q) $(OBJCOPY) --only-section .text -O binary $@ eagle.app.v6.text.bin
	$(Q) $(OBJCOPY) --only-section .data -O binary $@ eagle.app.v6.data.bin
	$(Q) $(OBJCOPY) --only-section .rodata -O binary $@ eagle.app.v6.rodata.bin
	$(Q) $(OBJCOPY) --only-section .irom0.text -O binary $@
eagle.app.v6.irom0text.bin
	$(vecho) "objcopy done"
	$(vecho) "Run gen_appbin.exe"
ifeq ($(app), 0)
	$(Q) $(SDK_TOOLS)/gen_appbin.exe $@ 0 $(mode) $(freqdiv) $(size_map) $(app)
	$(Q) mv eagle.app.flash.bin $(FW_BASE)/eagle.flash.bin
	$(Q) mv eagle.app.v6.irom0text.bin $(FW_BASE)/eagle.irom0text.bin
	$(Q) rm eagle.app.v6.*
	$(vecho) "No boot needed."
	$(vecho) "Generate eagle.flash.bin and eagle.irom0text.bin successully in
folder $(FW_BASE)"
	$(vecho) "eagle.flash.bin-------->0x00000"
	$(vecho) "eagle.irom0text.bin---->0x10000"
else
    ifneq ($(boot), new)
	$(Q) $(SDK_TOOLS)/gen_appbin.exe $@ 1 $(mode) $(freqdiv) $(size_map) $(app)
	$(vecho) "Support boot_v1.1 and +"
    else
	$(Q) $(SDK_TOOLS)/gen_appbin.exe $@ 2 $(mode) $(freqdiv) $(size_map) $(app)
	ifeq ($(size_map), 6)
	      $(vecho) "Support boot_v1.4 and +"
	  else
	      ifeq ($(size_map), 5)
	       $(vecho) "Support boot_v1.4 and +"
	      else
	       $(vecho) "Support boot_v1.2 and +"
	      endif
	  endif
    endif
	$(Q) mv eagle.app.flash.bin $(FW_BASE)/upgrade/$(BIN_NAME).bin
	$(Q) rm eagle.app.v6.*
	$(vecho) "Generate $(BIN_NAME).bin successully in folder $(FW_BASE)/upgrade"
	$(vecho) "boot.bin------->0x00000"
	$(vecho) "$(BIN_NAME).bin--->$(addr)"
endif
	$(vecho) "Done"

$(APP_AR): $(OBJ)
	$(vecho) "AR $@"
	$(Q) $(AR) cru $@ $^

checkdirs: $(BUILD_DIR) $(FW_BASE)

$(BUILD_DIR):
	$(Q) mkdir -p $@

$(FW_BASE):
	$(Q) mkdir -p $@
	$(Q) mkdir -p $@/upgrade

flashboot:
ifeq ($(app), 0)
```

```
      $(vecho) "No boot needed."
else
    ifneq ($(boot), new)
      $(vecho) "Flash boot_v1.1 and +"
      $(ESPTOOL) -p $(ESPPORT) -b $(ESPBAUD) write_flash $(flashimageoptions)
0x00000 $(SDK_BASE)/bin/boot_v1.1.bin
    else
      ifeq ($(size_map), 6)
            $(vecho) "Flash boot_v1.5 and +"
            $(ESPTOOL) -p $(ESPPORT) -b $(ESPBAUD) write_flash
$(flashimageoptions) 0x00000 $(SDK_BASE)/bin/boot_v1.6.bin
        else
            ifeq ($(size_map), 5)
            $(vecho) "Flash boot_v1.5 and +"
            $(ESPTOOL) -p $(ESPPORT) -b $(ESPBAUD) write_flash
$(flashimageoptions) 0x00000 $(SDK_BASE)/bin/boot_v1.6.bin
            else
            $(vecho) "Flash boot_v1.2 and +"
            $(ESPTOOL) -p $(ESPPORT) -b $(ESPBAUD) write_flash
$(flashimageoptions) 0x00000 $(SDK_BASE)/bin/boot_v1.2.bin
            endif
        endif
    endif
endif

flash: all
ifeq ($(app), 0)
      $(ESPTOOL) -p $(ESPPORT) -b $(ESPBAUD) write_flash $(flashimageoptions)
0x00000 $(FW_BASE)/eagle.flash.bin 0x10000 $(FW_BASE)/eagle.irom0text.bin
else
ifeq ($(boot), none)
      $(ESPTOOL) -p $(ESPPORT) -b $(ESPBAUD) write_flash $(flashimageoptions)
0x00000 $(FW_BASE)/eagle.flash.bin 0x10000 $(FW_BASE)/eagle.irom0text.bin
else
      $(ESPTOOL) -p $(ESPPORT) -b $(ESPBAUD) write_flash $(flashimageoptions)
$(addr) $(FW_BASE)/upgrade/$(BIN_NAME).bin
endif
endif

# ===============================================================
# From http://bbs.espressif.com/viewtopic.php?f=10&t=305
# master-device-key.bin is only need if using espressive services
# master_device_key.bin 0x3e000 is not used , write blank
# See 2A-ESP8266__IOT_SDK_User_Manual__EN_v1.1.0.pdf
# http://bbs.espressif.com/download/file.php?id=532
#
# System parameter area is the last 16KB of flash
# 512KB flash - system parameter area starts from 0x7C000
#     download blank.bin to 0x7E000 as initialization.
# 1024KB flash - system parameter area starts from 0xFC000
#     download blank.bin to 0xFE000 as initialization.
# 2048KB flash - system parameter area starts from 0x1FC000
#     download blank.bin to 0x1FE000 as initialization.
# 4096KB flash - system parameter area starts from 0x3FC000
#     download blank.bin to 0x3FE000 as initialization.
# ===============================================================

# FLASH SIZE
flashinit:
```

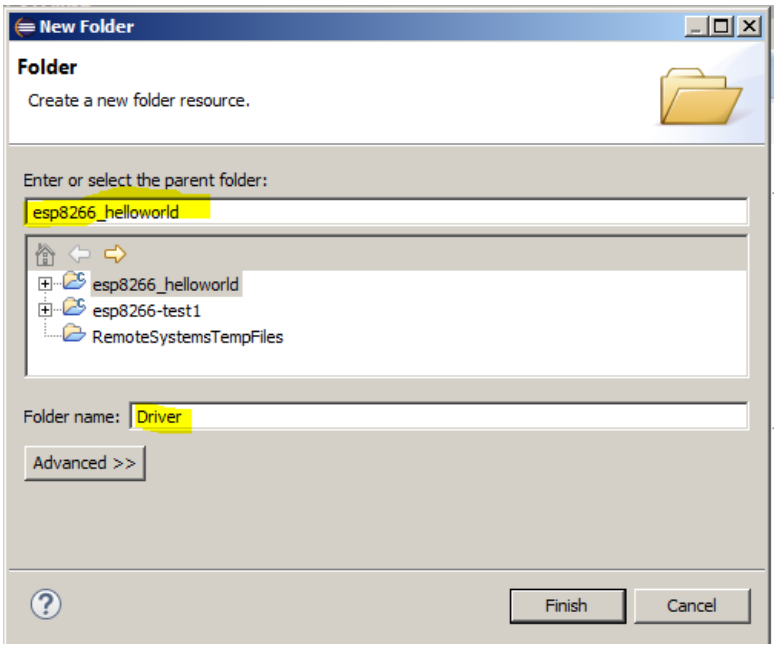
```
	$(vecho) "Flash init data default and blank data."
	$(ESPTOOL) -p $(ESPPORT) write_flash $(flashimageoptions) 0x7c000
$(SDK_BASE)/bin/esp_init_data_default.bin 0x7e000 $(SDK_BASE)/bin/blank.bin

rebuild: clean all

clean:
	$(Q) rm -f $(APP_AR)
	$(Q) rm -f $(TARGET_OUT)
	$(Q) rm -rf $(BUILD_DIR)
	$(Q) rm -rf $(BUILD_BASE)
	$(Q) rm -rf $(FW_BASE)

$(foreach bdir,$(BUILD_DIR),$(eval $(call compile-objects,$(bdir))))
```

In the source code there are some errors:

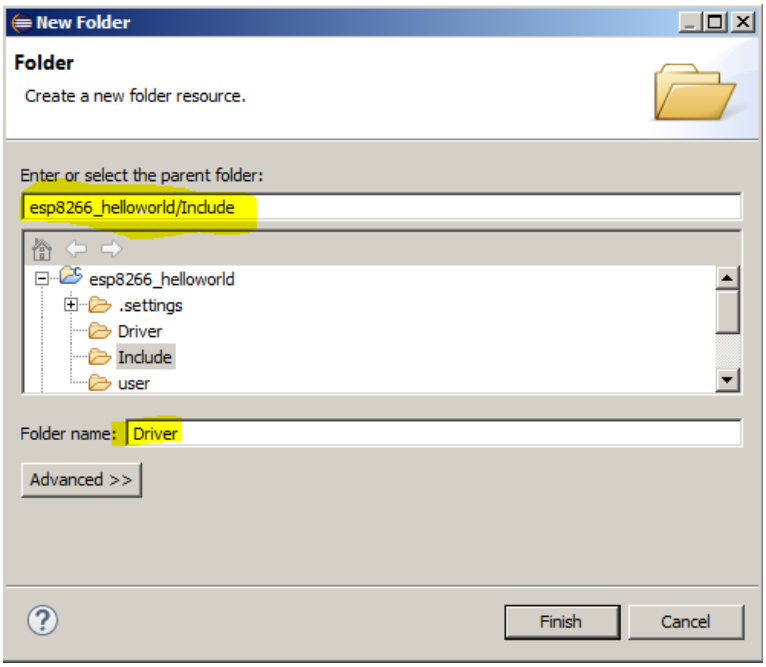
```
 8  #include <ets_sys.h>
 9  #include <osapi.h>
10  #include "user_interface.h"
11  #include <os_type.h>
12  #include <gpio.h>
13  #include "driver/uart.h"
14
```

This is beacuse there is no source code for the »drivers«. It's not necessary to write everything from scratch. Drivers for e.g. UARTs are there, we have to add them to the project. There are several methods to do this. We will use the simplest one: Copy driver folder to our project sub-folder and remove the unused drivers. Even easier is just to drag and drop the files to the proper folder.

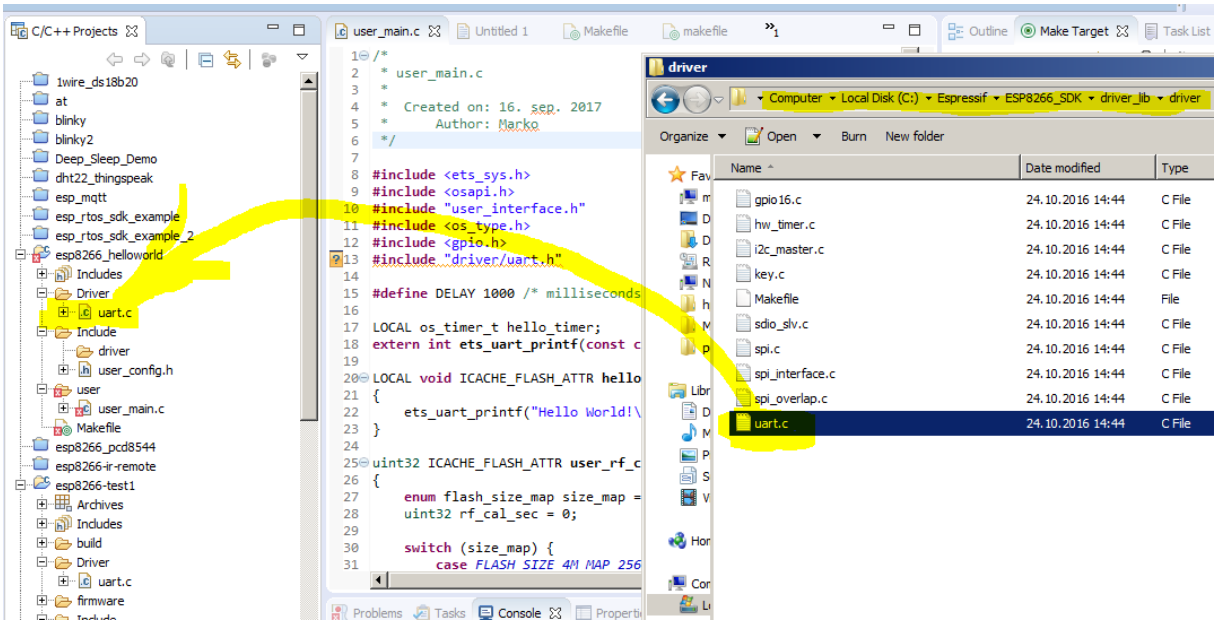
Let's make driver folders in the project. One for source and one for includes:



and second inside include folder:




Now open file explorer in windows and point to C:\Espressif\ESP8266_SDK\driver_lib\driver




Drag and drop file »uart.c« to Driver folder.

Repeat same with files »uart.h« and »uart_register.h« from folder C:\Espressif\ESP8266_SDK\driver_lib\include\driver

Now it's finally the time to compile the project. Double-click on »all« in make targets. If everything is OK, the output console should indicate success:

```
**** Build of configuration Release for project esp8266-helloworld ****
mingw32-make.exe -f D:/!work/xprojects/esp8266/my/ESPtests/Makefile all
CC driver/uart.c
CC user/user_main.c
AR build/app_app.a
LD build/app.out
Run objcopy, please wait...
objcopy done
Run gen_appbin.exe
No boot needed.
Generate eagle.flash.bin and eagle.irom0text.bin successully in folder firmware
eagle.flash.bin-------->0x00000
eagle.irom0text.bin---->0x10000
Done

23:48:37 Build Finished (took 1s.425ms)
```

Find following lines in the makefile:

```
# esptool path and port
ESPTOOL             = $(SDK_TOOLS)/esptool.exe
ESPPORT             = COM6

# Baud rate for programmer
ESPBAUD             = 115200
```

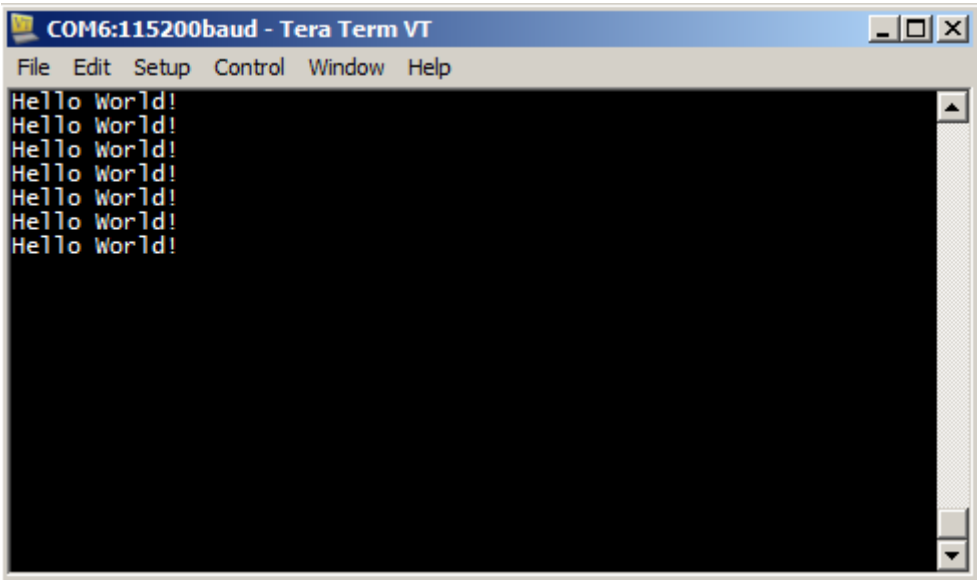
and make corrections if necessary for ESPPORT and ESPBAUD.

Now connect your module in flash load mode and double click on »flash« make target. The console will indicate the flashing process:

```
**** Build of configuration Release for project esp8266-helloworld ****
mingw32-make.exe -f D:/!work/xprojects/esp8266/my/ESPtests/Makefile flash
c:/Espressif/utils/ESP8266/esptool.exe -p COM6 -b 115200 write_flash -ff 40m -fm
qio -fs 32m 0x00000 firmware/eagle.flash.bin 0x10000 firmware/eagle.irom0text.bin
WARNING: Flash size arguments in megabits like '32m' are deprecated.
Please use the equivalent size '4MB'.
Megabit arguments may be removed in a future release.
esptool.py v2.0.0
Connecting....
Detecting chip type... ESP8266
Uploading stub...
Running stub...
Stub running...
Attaching SPI flash...
Configuring flash size...
Flash params set to 0x0040
Compressed 28560 bytes to 21386...

Writing at 0x00000000... (50 %)
```

```
Writing at 0x00004000... (100 %)
Wrote 28560 bytes (21386 compressed) at 0x00000000 in 1.9 seconds (effective 121.1
kbit/s)...
Hash of data verified.
Compressed 194596 bytes to 143947...

Writing at 0x00010000... (11 %)
Writing at 0x00014000... (22 %)
Writing at 0x00018000... (33 %)
Writing at 0x0001c000... (44 %)
Writing at 0x00020000... (55 %)
Writing at 0x00024000... (66 %)
Writing at 0x00028000... (77 %)
Writing at 0x0002c000... (88 %)
Writing at 0x00030000... (100 %)
Wrote 194596 bytes (143947 compressed) at 0x00010000 in 12.7 seconds (effective
123.0 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting...

23:53:41 Build Finished (took 16s.728ms)
```

Now connect the terminal and observe your first application on ESP8266 in action:




For finish double-click the »clean« target to tidy up the project folders.