Create a simple hello-world application

Status: [illegible] Revision: [illegible]

A quality version of this page, approved on *17 November 2020*, was based off this revision.

1 Overview

This stage explains how to create, build and execute a simple C code application using the freshly installed SDK.

2 Code

- Create a directory that will host your source codes

```
For ecosystem release  v2.1.0 ⓘ :
PC $> mkdir $HOME/STM32MPU_workspace/STM32MP15-Ecosystem-v2.1.0/Developer-Package
/stm32mp1-openstlinux-20-11-12
PC $> mkdir $HOME/STM32MPU_workspace/STM32MP15-Ecosystem-v2.1.0/Developer-Package
/stm32mp1-openstlinux-20-11-12/sources
For ecosystem release  v2.0.0 ⓘ :
PC $> mkdir $HOME/STM32MPU_workspace/STM32MP15-Ecosystem-v2.0.0/Developer-Package
/stm32mp1-openstlinux-20-06-24
PC $> mkdir $HOME/STM32MPU_workspace/STM32MP15-Ecosystem-v2.0.0/Developer-Package
/stm32mp1-openstlinux-20-06-24/sources
```

- Create a directory for your user space example

```
For ecosystem release  v2.1.0 ⓘ :
PC $> mkdir $HOME/STM32MPU_workspace/STM32MP15-Ecosystem-v2.1.0/Developer-Package
/stm32mp1-openstlinux-20-11-12/sources/gtk_hello_world_example
PC $> cd $HOME/STM32MPU_workspace/STM32MP15-Ecosystem-v2.1.0/Developer-Package/stm32mp1-
openstlinux-20-11-12/sources/gtk_hello_world_example
For ecosystem release  v2.0.0 ⓘ :
PC $> mkdir $HOME/STM32MPU_workspace/STM32MP15-Ecosystem-v2.0.0/Developer-Package
/stm32mp1-openstlinux-20-06-24/sources/gtk_hello_world_example
PC $> cd $HOME/STM32MPU_workspace/STM32MP15-Ecosystem-v2.0.0/Developer-Package/stm32mp1-
openstlinux-20-06-24/sources/gtk_hello_world_example
```

- Create the source code file for your user space example: *gtk_hello_world.c*

```
#include <gtk/gtk.h>

static void
print_hello (GtkWidget *widget,
             gpointer   data)
{
  g_print ("Hello World\n");
}

static void
activate (GtkApplication *app,
          gpointer        user_data)
{
  GtkWidget *window;
  GtkWidget *button;
  GtkWidget *button_box;

  window = gtk_application_window_new (app);
  gtk_window_set_title (GTK_WINDOW (window), "Window");
  gtk_window_set_default_size (GTK_WINDOW (window), 200, 200);

  button_box = gtk_button_box_new (GTK_ORIENTATION_HORIZONTAL);
  gtk_container_add (GTK_CONTAINER (window), button_box);
```

```
  button = gtk_button_new_with_label ("Hello World");
  g_signal_connect (button, "clicked", G_CALLBACK (print_hello), NULL);
  g_signal_connect_swapped (button, "clicked", G_CALLBACK (gtk_widget_destroy), window);
  gtk_container_add (GTK_CONTAINER (button_box), button);

  gtk_widget_show_all (window);
}

int
main (int    argc,
      char **argv)
{
  GtkApplication *app;
  int status;

  app = gtk_application_new ("org.gtk.example", G_APPLICATION_FLAGS_NONE);
  g_signal_connect (app, "activate", G_CALLBACK (activate), NULL);
  status = g_application_run (G_APPLICATION (app), argc, argv);
  g_object_unref (app);

  return status;
}
```

3 Build

- Create the makefile for your user space example: *Makefile*

Information

All the indentations in a makefile are tabulations

```
PROG = gtk_hello_world
SRCS = gtk_hello_world.c

CLEANFILES = $(PROG)

# Add / change option in CFLAGS and LDFLAGS
CFLAGS += -Wall $(shell pkg-config --cflags gtk+-3.0)
LDFLAGS += $(shell pkg-config --libs gtk+-3.0)

all: $(PROG)

$(PROG): $(SRCS)
	$(CC) -o $@ $^ $(CFLAGS) $(LDFLAGS)

clean:
	rm -f $(CLEANFILES) $(patsubst %.c,%.o, $(SRCS))
```

- Cross-compile the project

```
PC $> make
```

4 Deploy and execute

- Push this binary into the board (Ethernet connection needed)

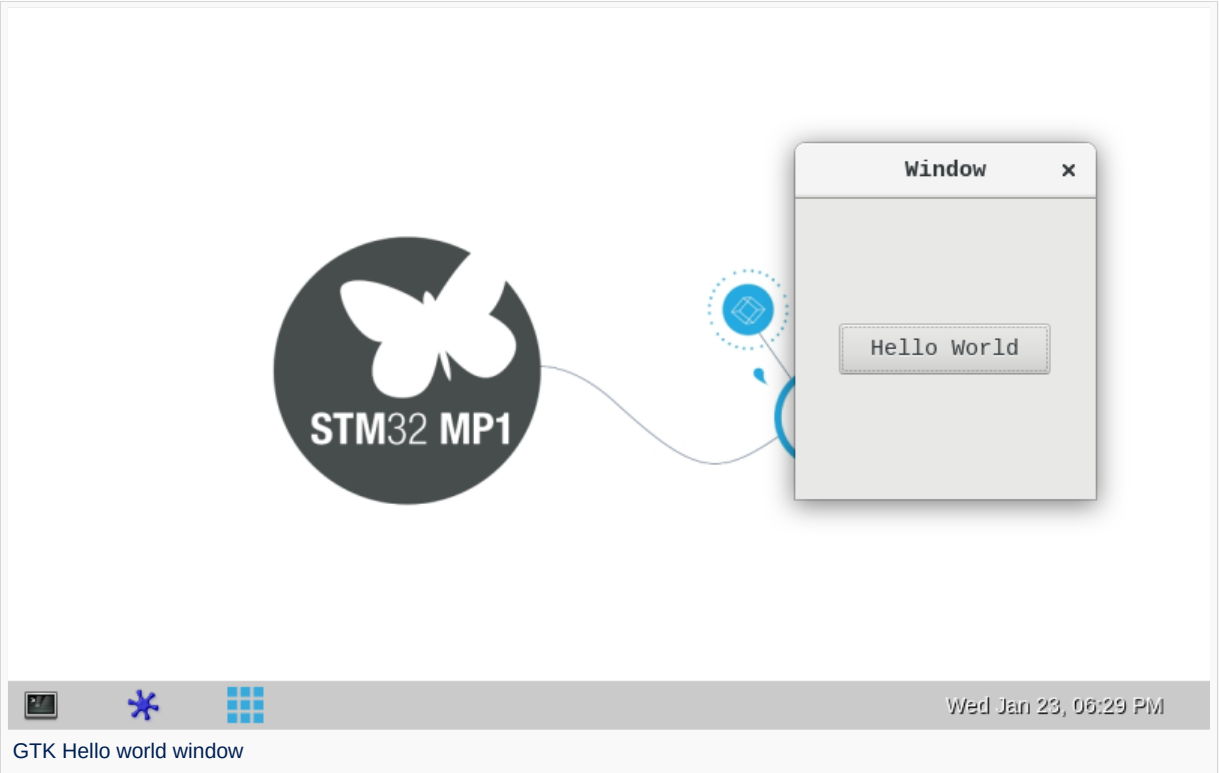
```
PC $> scp gtk_hello_world root@<board ip address>:/usr/local
```

- Execute your user space example

```
Board $> cd /usr/local/
Board $> ./gtk_hello_world


(gtk_hello_world:6370): dbind-WARNING **: 18:17:49.914: Error retrieving accessibility
bus address: org.a11y.Bus.Error: Failed to execute chi)
```

- A GTK windows is displayed



GTK Hello world window

- Click on the "Hello world" button to close the window. *Hello world* is displayed in the console:

```
Hello world
```

Software development kit (A programming package that enables a programmer to develop applications for a specific platform.)