GStreamer on Symbian using Carbide C++

19 Sep 2010

Introduction

GStreamer is a multimedia plug-in based framework. The main platform audience of GStreamer is the Linux OS what made GStreamer the official multimedia framework of Maemo and the new platform Meego. However, as GStreamer is totally based on Glib/GObject libraries, it is possible to port this framework to Symbian OS using its OpenC libraries. That porting was done and is available in the GStreamer project, and is already merged in the Symbian OSS repository (should be available in the next releases of Symbian^3 SDK).

This article has the objective to explain in a simple and directly way how to setup your environment using Carbide C++ and Nokia S60 SDK (3rd Ed. FP2 and 5th Ed) to get started with GStreamer on Symbian platform. In this article is shown the steps to setup GStreamer in your Emulator. Future articles will explain how to setup it in your S60 phone.

Setup your Environment

1. First download and Setup your Symbian C++/Nokia S60 environment
   1. Install Nokia S60 3rd Edition FP2 SDK from Nokia Developer
   2. Install Carbide C++ IDE from Nokia Developer: How do I start programming for Symbian OS?
   3. Install Open C/C++ plugins in your SDK from Nokia Developer: Open C/C++ Release History
2. Download the GStreamer for Symbian source code/binaries from GStreamer Project
   1. GStreamer Project
   2. GStreamer for Symbian source code
   3. GStreamer for Symbian binaries
3. Extract the GStreamer packages in your preferred location

Tip: Do not extract the source code in a folder with “blank” spaces, such as **C:/Document and Settings/** . Some Carbide C++ environments do not handle well this kind of folder

Get Started

Setup GStreamer Libraries and plugins

Here we have two ways: • Use pre-compiled binaries (simple way). Here you can go directly to the application development stuff. • Try to compile the source code (funniest way). Here you can know how to setup your environment (Carbide C++) to help and contribute with the GStreamer source code.

Use pre-compiled binaries

After extract the gstreamer_binaries package, you should have something like this:

```
Gstreamer_binaries/
|------------------ epoc32/
|-------------------sf/
|------------------readme.txt
```

Then you should follow:

1. Copy and merge the **epoc32/** folder into your S60 SDK. This folder contains all pre-compiled DLL and EXE files for the emulator and phone platforms. For example: copy and merge the epoc32/ folder into your S60 3rd Edition FP2 SDK -> **C:\S60\devices\S60_3rd_FP2_SDK_v1.1**

2. The sf/ (more specifically **sf\mw\gstreamer\include\gstreamer**) folder contains all necessary include files (C header files), necessary to develop GStreamer applications. You can use it in two ways:

a) When developing your applications, you can copy these header files into your application folder OR refer into your applications where these files are placed using the USERINCLUDE keyword in the MMP file:

Pre-Compiled sample MMP file 1

```
//Here we have to use “..” to correctly referencing gstreamer includes
USERINCLUDE  ../../../include/gstreamer
USERINCLUDE  ../../../include/gstreamer/gst
USERINCLUDE  ../../../include/gstreamer/gst/base
USERINCLUDE  ../../../include/gstreamer/gst/controller
USERINCLUDE  ../../../include/gstreamer/gst/dataprotocol
USERINCLUDE  ../../../include/gstreamer/gst/net
```

b) You can copy this header files directly into your SDK include folder. Therefore compiling the content of specifically

(C) Copyright Nokia 2013. All rights reserved.

sf\mw\gstreamer\include\gstreamer to **C:\S60\devices\S60_3rd_FP2_SDK_v1.1\epoc32\include\gstreamer** folder. That way you can use SYSTEMINCLUDE keywords in your MMP file:

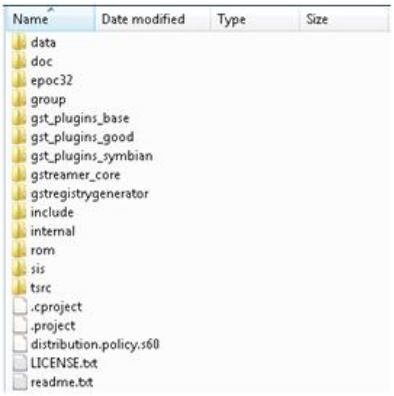
- **Pre-Compiled sample MMP file 2**

```
//Here we can directly the headers in the SDK
USERINCLUDE  epoc32/include/gstreamer
USERINCLUDE  epoc32/include/gstreamer/gst
USERINCLUDE  epoc32/include/gstreamer/gst/base
USERINCLUDE  epoc32/include/gstreamer/gst/controller
USERINCLUDE  epoc32/include/gstreamer/gst/dataprotocol
USERINCLUDE  epoc32/include/gstreamer/gst/net
```

Now you can develop GStreamer applications in Symbian. In one of the next topics it will be shown how to compile a sample application that uses GStreamer.

Compile GStreamer

After extract the gstreamer_source package, you should have the following folder structure:

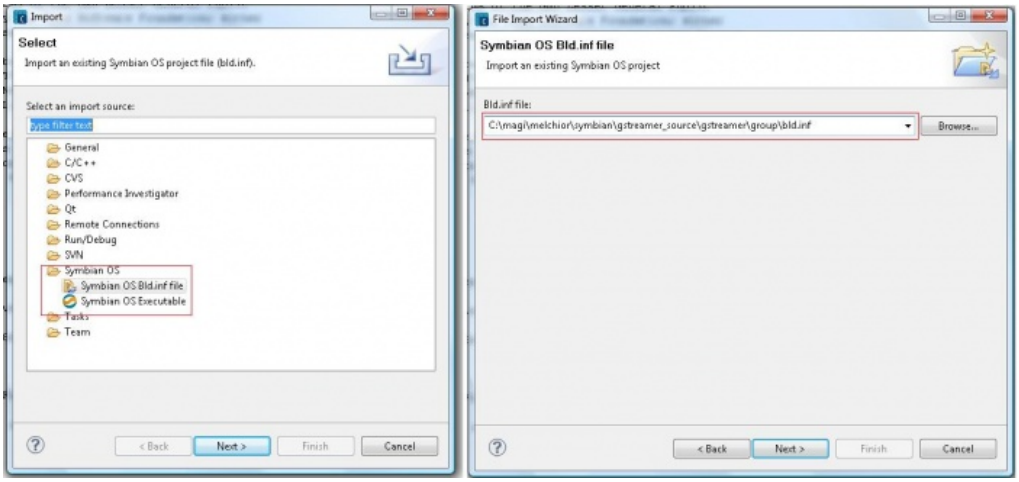


Important folders:

- **Data** folder: contains important configuration files. Also, it has some pre-compiled DLL and LIB files, which need external dependencies to compile.
- **Doc** folder: contains GStreamer documentation.
- **Epoc32** folder: contains DLL and LIB files (dependencies) required to compile the code.
- **Group** folder: Contains the project files to be imported from Carbide C++.
- **Gst*** folders: Contains GStreamer ported code.
- **Include** folder: Contain GStreamer include header files (As explained in item 2 of the previous topic)
- **Internal** folder: Contains a sample application and useful scripts to setup the environment

Compile with Carbide C++

First, import the GStreamer project with Carbide C++. Go to **File->Import->Symbian OS->Symbian OS bld.inf** file. Then import the group/bld.inf file.



After that, choose you SDK and choose ALL MMP files of the bld.inf file. At the end, you should have the same project folder organization in your Carbide C++ as in your source folder.

When compiling GStreamer you are really compiling various projects. Some of these DLL projects (inside the **gst_plugins_symbian/** folder) need external libraries and headers that are not available in the Nokia S60 SDK provided by Nokia Developer. However, GStreamer source code distribution provides those DLL already compiled and ready to use. These binaries are available in the **data/dependent_binaries** folder.

In the 3rd Edition FP2 SDK, you should copy the files in the **data/dependent_binaries/3_2/winscw folder into the <SDK folder>/epoc32/release/winscw/udeb/** folder. The same procedure can be used with 5th Edition SDKs. After that you can remove these projects from

(C) Copyright Nokia 2013. All rights reserved.

the BUILD environment editing the **group/bld.inf** file as follow:

group/bld.inf

```
PRJ_MMPFILES

#include "../gstreamer_core/group/bld.inf"
#include "../gst_plugins_base/group/bld.inf"
#include "../gst_plugins_good/group/bld.inf"
#include "../gstregistrygenerator/group/bld.inf"
//  can't build due to lack of dependencies.
//#include "../gst_plugins_symbian/group/bld.inf"  <<<<<<<< Comment this LINE
```

After that, you can COMPILE the whole project. After compilation the plugins should be available at the <sdk_folder>/epoc32/release/winscw/udeb/ (look for libgst* files).

Post Compilation Script

The source code package provides the internal/miscfiles/copyplugins.bat script that aims to copy all created plugins into a required folder for GStreamer. However, this script requires the setup of some environment variables to work properly. Here it is provided a simple way to use this script. First, create a copyplugins.bat file in the root of the <sdk_folder>, for example: **C:\S60\devices\S60_3rd_FP2_SDK_v1.1\copyplugins.bat**

Use the following content in this script:

copyplugins.bat

```
md epoc32\release\winscw\udeb\z\sys\bin\plugins

COPY epoc32\release\winscw\udeb\libgstcoreelements.dll epoc32\release\winscw\udeb\z\sys\bin\plugins\libgstcoreelements.dll
COPY epoc32\release\winscw\udeb\libgstcoreindexers.dll epoc32\release\winscw\udeb\z\sys\bin\plugins\libgstcoreindexers.dll
COPY epoc32\release\winscw\udeb\libgstwavparse.dll epoc32\release\winscw\udeb\z\sys\bin\plugins\libgstwavparse.dll
COPY epoc32\release\winscw\udeb\libgstwavenc.dll epoc32\release\winscw\udeb\z\sys\bin\plugins\libgstwavenc.dll
COPY epoc32\release\winscw\udeb\libgstdevsoundsink.dll epoc32\release\winscw\udeb\z\sys\bin\plugins\libgstdevsoundsink.dll
COPY epoc32\release\winscw\udeb\libgstdevsoundsrc.dll epoc32\release\winscw\udeb\z\sys\bin\plugins\libgstdevsoundsrc.dll
COPY epoc32\release\winscw\udeb\libgstaudioconvert.dll epoc32\release\winscw\udeb\z\sys\bin\plugins\libgstaudioconvert.dll
COPY epoc32\release\winscw\udeb\libgstaudioresample.dll epoc32\release\winscw\udeb\z\sys\bin\plugins\libgstaudioresample.dll
COPY epoc32\release\winscw\udeb\libgstaudiotestsrc.dll epoc32\release\winscw\udeb\z\sys\bin\plugins\libgstaudiotestsrc.dll
COPY epoc32\release\winscw\udeb\libgstdecodebin.dll epoc32\release\winscw\udeb\z\sys\bin\plugins\libgstdecodebin.dll
COPY epoc32\release\winscw\udeb\libgstdecodebin2.dll epoc32\release\winscw\udeb\z\sys\bin\plugins\libgstdecodebin2.dll
COPY epoc32\release\winscw\udeb\libgstplaybin.dll epoc32\release\winscw\udeb\z\sys\bin\plugins\libgstplaybin.dll
COPY epoc32\release\winscw\udeb\libgsttypefindfunctions.dll epoc32\release\winscw\udeb\z\sys\bin\plugins\libgsttypefindfunctions.dll
COPY epoc32\release\winscw\udeb\libgstqueue2.dll epoc32\release\winscw\udeb\z\sys\bin\plugins\libgstqueue2.dll
COPY epoc32\release\winscw\udeb\libgstaudiorate.dll epoc32\release\winscw\udeb\z\sys\bin\plugins\libgstaudiorate.dll
COPY epoc32\release\winscw\udeb\libgstautodetect.dll epoc32\release\winscw\udeb\z\sys\bin\plugins\libgstautodetect.dll
COPY epoc32\release\winscw\udeb\libgstapp.dll epoc32\release\winscw\udeb\z\sys\bin\plugins\libgstapp.dll
COPY epoc32\release\winscw\udeb\libgstsubparse.dll epoc32\release\winscw\udeb\z\sys\bin\plugins\libgstsubparse.dll
COPY epoc32\release\winscw\udeb\libgstgdp.dll epoc32\release\winscw\udeb\z\sys\bin\plugins\libgstgdp.dll
COPY epoc32\release\winscw\udeb\libgstadder.dll epoc32\release\winscw\udeb\z\sys\bin\plugins\libgstadder.dll
COPY epoc32\release\winscw\udeb\libgstvolume.dll epoc32\release\winscw\udeb\z\sys\bin\plugins\libgstvolume.dll
COPY epoc32\release\winscw\udeb\libgsttcp.dll epoc32\release\winscw\udeb\z\sys\bin\plugins\libgsttcp.dll
COPY epoc32\release\winscw\udeb\libgstmulaw.dll epoc32\release\winscw\udeb\z\sys\bin\plugins\libgstmulaw.dll
COPY epoc32\release\winscw\udeb\libgstalaw.dll epoc32\release\winscw\udeb\z\sys\bin\plugins\libgstalaw.dll
COPY epoc32\release\winscw\udeb\libgstauparse.dll epoc32\release\winscw\udeb\z\sys\bin\plugins\libgstauparse.dll
```

Save the script and run it the SDK root folder. After that verify if all plugins (libgst* files) are in **epoc32\release\winscw\udeb\z\sys\bin\plugin**. After that you GStreamer environment is ready.

Compile Sample Application

After setup (or compile) the necessary GStreamer libraries and plugins, you can use an application provided in the gstreamer_source package, the GSTPlayer application. It is necessary to compile this application together with your SDK.

For that, using Carbide C++, import the project from gstreamer_source/internal/gstplayer/group /bld.inf as explained in the previous section.

Before compiling the project, it is necessary to copy some needed files for the compilation. These files are available at: **gstreamer_source/epoc32 folder** (mmfdevsound files). Just copy and merge this folder into the SDK root, eg. **C:\S60\devices\S60_3rd_FP2_SDK_v1.1**

Then, you can compile the project. After compiling the project you can run this application using Nokia S60 SDK Emulator.

(C) Copyright Nokia 2013. All rights reserved.

PS. The application just support play AU, RAW and WAV files. Also, it takes some time to open, therefore, be patient :)

Current Status

Currently GStreamer on Symbian supports the following formats:

- **On Phone**
  - Recording formats : RAW, WAV, AMR-NB, G711, G729, ILBC.
  - Playback formats : AU, RAW, WAV, AMR-NB, G711, G729, ILBC.
- **On Emulator**
  - Recording formats : RAW, WAV.
  - Playback formats : AU, RAW, WAV.

More Information

Documentation

- Most of GStreamer documentation is provided in its website.
- More details about how to compile and use GStreamer on Symbian platform, please refer the **readme.txt** file provided in the source package.

External Links

- GStreamer web site

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.