**Module-description variables:**

The following variables are used to describe your module to the build system. You should define some of them between an 'include $(CLEAR\_VARS)'

and an 'include $(BUILD\_XXXXX)'. As written previously, $(CLEAR\_VARS) is a script that will undefine/clear all of these variables, unless explicitly

noted in their description.

LOCAL\_PATH

This variable is used to give the path of the current file.

You MUST define it at the start of your Android.mk, which can be done with:

LOCAL\_PATH := $(call my-dir)

This variable is \*not\* cleared by $(CLEAR\_VARS) so only one definition per Android.mk is needed (in case you define several

modules in a single file).

LOCAL\_MODULE

This is the name of your module. It must be unique among all module names, and shall not contain any space. You MUST define

it before including any $(BUILD\_XXXX) script.

By default, the module name determines the name of generated files,

e.g. lib<foo>.so for a shared library module named <foo>. However you should only refer to other modules with their 'normal'

name (e.g. <foo>) in your NDK build files (either Android.mk or Application.mk)

You can override this default with LOCAL\_MODULE\_FILENAME (see below)

LOCAL\_MODULE\_FILENAME

This variable is optional, and allows you to redefine the name of generated files. By default, module <foo> will always generate a

static library named lib<foo>.a or a shared library named lib<foo>.so, which are standard Unix conventions.

You can override this by defining LOCAL\_MODULE\_FILENAME, For example:

LOCAL\_MODULE := foo-version-1

LOCAL\_MODULE\_FILENAME := libfoo

NOTE: You should not put a path or file extension in your LOCAL\_MODULE\_FILENAME, these will be handled automatically by the

build system.

LOCAL\_SRC\_FILES

This is a list of source files that will be built for your module.

Only list the files that will be passed to a compiler, since the build system automatically computes dependencies for you.

Note that source files names are all relative to LOCAL\_PATH and you can use path components, e.g.:

LOCAL\_SRC\_FILES := foo.c \

toto/bar.c

NOTE: Always use Unix-style forward slashes (/) in build files. Windows-style back-slashes will not be handled properly.

LOCAL\_CPP\_EXTENSION

This is an optional variable that can be defined to indicate

the file extension of C++ source files. The default is '.cpp' but you can change it. For example:

LOCAL\_CPP\_EXTENSION := .cxx

Since NDK r7, you can list several extensions in this variable, as in:

LOCAL\_CPP\_EXTENSION := .cxx .cpp .cc

LOCAL\_CPP\_FEATURES

This is an optional variable that can be defined to indicate

that your code relies on specific C++ features. To indicate that

your code uses RTTI (RunTime Type Information), use the following:

LOCAL\_CPP\_FEATURES := rtti

To indicate that your code uses C++ exceptions, use:

LOCAL\_CPP\_FEATURES := exceptions

You can also use both of them with (order is not important):

LOCAL\_CPP\_FEATURES := rtti exceptions

The effect of this variable is to enable the right compiler/linker

flags when building your modules from sources. For prebuilt binaries,

this also helps declare which features the binary relies on to ensure

the final link works correctly.

It is recommended to use this variable instead of enabling -frtti and

-fexceptions directly in your LOCAL\_CPPFLAGS definition.

-------