This Document describes the design of Slog

DESIGN DOCUMENT

Document ID:

Slog

Slog User Guide

TABLE OF CONTENTS

1 What is Slog?

It is the debug/trace/logging tool provides rich set of features as mentioned below. The log messages by application's is called as events in this modules

- Traces the events module wise
- Almost 10 different trace levels for each module.
- Compile time and runtime enable/disable traces for every module & every level.
- Reduce foot print of target binary by removing debug prints directly from binary.
- Configurable support for Logging to console, Logging to buffer, etc…
- Timestamp all the events
- Origin (File path & line no) of the event

2 Directory Structure

- Slog_XX_XX_XX_XX		Slog root directory, Also has interface files
	+ docs	Documents
	+ samples	Sample applications
	+ src	Slog implementation code, makefile, library

3 How to Build & Run

3.1 Build Slog library

- Install code sourcery for Linux.
  - Code Sourcery: http://www.codesourcery.com/sgpp/lite/arm/portal/release1600
  -
- Add the path of bin folder of MinGW or Code Sourcery installation to the PATH environment variable.
- Change the COMPILER_PREFIX variable in makefile file appropriately.
- Execute "make". This creates a library(archive) file slog.a in the src folder.

3.2 Build sample application

- Go to the samples
- Change the COMPILER_PREFIX variable in makefile file appropriately.
- Execute make. This creates executable of each samples under respective directories
- Execute exe file generated.

Each sample applications can be built using the above procedure.

3.3 Run Sample applications

The application built above can be executed as below

./HelloWorld_static.exe

4 How to integrate with applications

4.1 Static Mode

Application file changes

Step1: Add the below lines in the beginning of each C file.

```
#define MYTESTMODULEID (0x8000)
#define Module__MID MYTESTMODULEID
```

These statements indicate to the Logger module this file belongs to the module indicated by the module id.

Note: If these lines are not present in the C file then this file becomes part of the default module called as “main module”. If these lines are being put below the included Log.h file, then use #undef Module__MID.

Note: User must ensure that correct module id is used for each file. (ie. Multiple files can be part of same module, but it must be ensured that given c file is not passed with different module id.

Step2: Include below files in each C file

```
#include <Log.h>
#include <LoggerSys.h> ← Include LoggerBuf.h in case Logging to buffer mode is used
```

Note: For the applications already using RTSC package, ensure that the above include follows all the RTSC specific include files. Otherwise you will get build errors similar to the one below

```
/xdc/std.h:185:0: warning: "Void" redefined
/xdc/std.h:187:25: error: redefinition of typedef 'Char'
/xdc/std.h:188:25: error: redefinition of typedef 'UChar'
…
```

Step3: Below table is optional to be to be defined by Application. If exists this table would be used, otherwise default table with NULL entry would be used.

```
Diags_DictElem Diags_dictElems[n] ={
        {"module1", MOD1MODID, MOD1RUNON, MOD1ALSOFF, MOD1ALSON},
            .
            .
        {"modulen-1",MODn-1MODID,MODn-1RUNON, MODn-1ALSOFF, MODn-1ALSON},
        {NULL, 0, 0, 0, 0}
        };
```

XXXRUNON: Diags mask corresponding to the Runtime On.
XXXALSOFF: Diags mask corresponding to the Always Off.
XXXALSON: Diags mask corresponding to the Always On value in RTSC
XXXMODID: Unique module id, should start from 1. 0 is the reserved id which indicates the end of the records in the table.

Note: The table must have {NULL, 0, 0, 0, 0} as the last entry, otherwise the result is undefined.

Step4: link libraries slog.a and slog_modtbl.a to application. The library slog_modtbl.a must be defined in linker order such that it listed after the application defined module table.

4.2 Dynamic Mode

Application file changes

Step1: Add the below lines in the beginning of each C file.

```
#define MYTESTMODULEID (Registry_findByName("ModName")->id)
#define Module__MID MYTESTMODULEID
ModName- Module to which the file is desired to be inserted.
```

These statements indicate to the Logger module this file belongs to the module indicated by the module id.

Note: If these lines are not present in the C file then this file becomes part of the default module called as "main module". If these lines are being put below the included Log.h file, then use #undef Module__MID.

Note: User must ensure that correct module id is used for each file. (ie. Multiple files can be part of same module, but it must be ensured that given c file is not passed with different module id.

Step2: Include below files in each C file

```
#include <Registry.h>
#include <Log.h>
#include <LoggerSys.h> ← Include LoggerBuf.h in case Logging to buffer mode is used
```

Note: Registry.h must be the first file to get included.

Step3: Application containing main must declare Diags_dictElems variable providing Null values like

```
Diags_DictElem Diags_dictElems[1] ={{NULL, 0, 0, 0, 0}};
```

Step4: Application must call below API for each module passing the module name & trace masks

```
Registry_Desc desc;
Registry_addModule(&desc, ModName", R_ON_F, A_OFF_F, A_ON_F, R_OFF_F);

A_ON_F: Diags mask corresponding to the Always On value in RTSC.
A_OFF_F: Diags mask corresponding to the Always Off value in RTSC.
R_ON_F: Diags mask corresponding to the Runtime On value in RTSC
R_OFF_F: Diags mask corresponding to the Runtime Off value in RTSC
descName: An instance of  Registry_Desc structure
```

Note: Main should compulsorily create a main module besides other modules.
See Examples in Phonebook_Registry folder in samples

5 Sample Code

5.1 Registry Mode

Refer to $(SLOG_INSTALL_DIR)\samples\HelloWorld\HelloWorld_Registry.c

5.2 Static mode

Refer to $(SLOG_INSTALL_DIR)\samples\HelloWorld\HelloWorld_static.c

6 Version History

Revision Number	Date	Description
0.1	17-Jun-11	Initial draft

«««« § »»»»