STAPLER

1 Revision History

This document details the changes made to STAPLER.

Build details: 1.7.3
Version: STAPLER_REL_1.7.3

Version	Release Date	Description
1.7.3	23/02/2011	Fixed BriSec2724 - compilation warnings. Fixed BriSec2837 - STAPLER_SemaphoreWait returns -1 upon timeout.
1.7.2	24/11/2010	Changed STCOMMON dependency mechanism. To remove dependency, define STAPLER_NO_STCOMMON_HANDLES.
1.7.1	22/11/2010	Removed dependency on STCOMMON for legacy use-cases. BriSec2670
1.7.0	18/10/2010	Added STAPLER_InterruptMap function for 7108 DCP interrupt mapping. Feature request BriSec2567. Fixed Linux makefile clean target (BriSec2590)
1.6.0	10/09/2010	Adding UnmapCached and UnmapUncached addresses Polled interrupts under STOS
1.5.0	31/08/2010	Fixed BRISec2499 - STAPLER_HandleAlloc, STAPLER_HandleFree not exported in Linux kernel space Enhancement BRISec2508 - Added STAPLER_MutexTryLock function Change request BRISec2514 - Increased maximum supported semaphore handles from 16 to 64
1.4.0	12/08/2010	Added STAPLER_HandleAlloc and STAPLER_HandleFree functions (BRISec2486)
1.3.2	04/08/2010	Fixed BRISec1814 - Bare mode STAPLER_TimerWait hangs indefinitely Fixed BRISec2309 - Fixed Linux end-user makefile ignored error Fixed BRISec2462 - 5197 bare support and bare compilation
1.3.1	22/04/2010	Fixed BRISec2298 - Linux module now exports STAPLER_BuildMode Re-introduced STx5107 and STx5162 support

Version	Release Date	Description
1.3.0	11/03/2010	Added support for STx7108 Added Queue support
1.2.2	25/02/2010	Added support for STx7106
1.2.1	30/11/2009	Added support for STx5289/ STx5206
1.2.0	23/03/09	Added sypport for TaskCreate, TaskDelete and TaskWait and fixed typo in stapler_core_exports.c. Also support for Mutexes was also added.
1.1.0	06/03/09	Added mutex support. Fixed OS20 support issues
1.0.8	24/02/09	Fixed user space Linux build. Additions to firmware_defs.h
1.0.7	03/02/09	Replaced reference to STOS_Semaphore_t with semaphore_t to work with STOS versions before 2.3.2
1.0.6	03/02/09	Fixed UncachedToCached and CachedToUncached memory mapping functions
1.0.5	23/01/09	Added firmware_defs.h file into release package
1.0.4	23/01/09	Added firmware definitions file for use by STLoad driver
1.0.3	08/01/09	Fixed some ST20 build issues
1.0.2	11/12/08	Updated bare makefiles
1.0.1	05/12/08	Fixed bug with interrupt functions
1.0.0	03/12/08	Initial release

1.1 Introduction

The ST Abstraction Platform LayER, STAPLER, provides an interface between security drivers and the host runtime environment. It was developed to remove driver dependancy on the implementation of STOS and to allow security drivers to be used within the limited runtime environemt of boot loaders. STAPLER provides the greatest benefit to security driver implementation, by reducing the number of required driver variants, but this also has knock on benefits forthe customer, reducing the number of driver variants that have to be manage and reducing the effort required to integrate STAPI updates.

Three runtime environemnts are supported.

- STOS (STAPI) mode for use in STAPI driver trees.
- BARE mode for use in boot loader and minimal systems where no operating system is available.
- REALBARE mode for use in minimal systems where no operating system is available and no standard C library is available to link against.

Note, BARE modes do not use interrupts and are not intended to work in a multi-threaded system.

Unlike most security drivers, STAPLER is released as source code and so can be freely modified. If requesting support for issues related to STAPLER, please include and describe any modifications made.

1.2 Version Numbers

Security drivers and related software products are individually versioned using a numeric, {major}.{minor}.{patch} scheme. The three components are defined as follows,

- A **patch** level change indicates a minor feature or bugfix. Methods **MAY** have been added to the API, but none have been removed.

eg. Customer code written for version 1.0.0 will work on version 1.0.9

- A **minor** level change indicates a change to the API which may require some code changes. API changes are minor, but are necessary for bug fixes or new features.

eg. Customer code written for version 1.0.3 **MAY** require changes for 1.1

- A **major** level indicates big changes to the API or considerable extra functionality. Developers can expect significant changes to bring their code up to date.

eg. Customer code written for version 1.1 is unlikely to work under 2.0

1.3 Supported SoC's

SoC	CHIP Name
STx7108	7108
STx7109	7109
STx7200	7200
STx7111	7111
STx7105	7105
STx7141	7141
STx5206	5206
STx5289	5206
STx5197	5197
STx7106	7106
STx7108	7108
STx5162	5162
STx5107	5107

Table 1

1.4 Building STAPLER

1.4.1 Building for STAPI platforms

STAPLER will be delivered as part of the STAPI Secure SDK and is integrated into the STAPI build system.

1.4.2 Building in BARE mode

Prerequisites for building STAPLER in BARE mode:

- ST40 Toolset on host path.
- Environment variable CHIP set to the numeric component of your target SoC name. For example, CHIP=7109 or CHIP=5206. See section 1.3, Table 1 above for the list of supported SoC's.

To build libstapler.a, call `make` from the STAPLER top level directory. For example, to build STAPLER for a 5206,

```
make CHIP=5206
```

If BARE mode is required inside a STAPI tree the environment variable STAPLER_MODE=bare must be set to force BARE mode.

1.4.3 Building in REALBARE mode

Prerequisites for building STAPLER in REALBARE mode:

- ST40 Toolset on host path.
- Set environment variable CHIP to the numeric component of your target SoC name. For example, CHIP=7109 or CHIP=5206. See section 1.3, Table 1 above for the list of supported SoC's.
- Set environment variable STAPLER_MODE=realbare.

To build libstapler.a, call make from the STAPLER top level directory. For example, to build STAPLER for a 5206,

```
make STAPLER_MODE=realbare CHIP=5206
```