CrossCore® Development Tools by Analog Devices

Release Notes for CrossCore® Embedded Studio 2.9.4

1 Table of Contents

2 Introduction

This document describes the changes for CrossCore Embedded Studio (CCES) 2.9.4. You can find the release notes for older releases in the `docs` sub-directory of your CCES installation as well as an Installation Guide which will help you install this release.

3 New and Noteworthy

3.1 New -pgo-hw API for setting the flush API interval

A new runtime library API to specify the interval in milliseconds between PGO data flushes has been added.

```
#include <pgo_hw.h>
pgo_hw_status_t pgo_hw_set_flush_interval(unsigned msec);
```

3.2 New byteswap4 SHARC compiler builtin function

A new SHARC compiler builtin function has been added to enable optimal changing of data from big-endian to little-endian format.

```
#include <builtins.h>
int byteswap4(int in);
```

3.3 securebootsim support for ADSP-2156x ECDSA-256

A new build of securebootsim, `System/securebootsim_cfgP256CP0sw`, has been added to enable testing of ECDSA-256 signed boot image loader files.

3.4 elfloader -MaxImageSize switch

The elfloader utility supports a new `-MaxImageSize` switch that takes a numeric argument. This switch is intended for use when you have a limited amount of flash memory space available for your boot image, and you want to ensure that the boot image created will fit within that space. The switch does not change the size of the boot image itself, but will issue an error in the case that its size exceeds the specified maximum. For more information, see the *Loader and Utilities Manual* in the Online Help.

3.5 __STDC_LIB_EXT1__ compatible safe string functions

The run-time libraries for Blackfin and SHARC introduce support for the safe _s variants of the string functions defined in the __STDC_LIB_EXT1__ extension to C11. These functions provide additional constraint checking and will call a constraint handler in addition to returning an error code when a violation is detected. These are available by default but can be excluded by defining `__STDC_WANT_LIB_EXT1__` to 0.

The new APIs introduced are:

```
  errno_t memcpy_s(void *__restrict s1, rsize_t s1max, const void *__restrict s2,
rsize_t n);
  errno_t memmove_s(void *s1, rsize_t s1max, const void *s2, rsize_t n);
  errno_t memset_s(void *dest, rsize_t smax, int c, rsize_t n);
  errno_t strcpy_s(char *__restrict s1, rsize_t s1max, const char *__restrict s2);
  errno_t strncpy_s(char *__restrict s1, rsize_t s1max, const char *__restrict s2,
rsize_t n);
  errno_t strcat_s(char *__restrict s1, rsize_t s1max, const char *__restrict s2);
  errno_t strncat_s(char *__restrict s1, rsize_t s1max, const char *__restrict s2,
```

```
rsize_t n);
  char *strtok_s(char *__restrict s1, rsize_t *__restrict s1max, const char
*__restrict s2,char **__restrict ptr);
  rsize_t strnlen_s(const char *str, rsize_t size);
  errno_t strerror_s(char *buf, rsize_t bufsz, errno_t errnum);
  size_t strerrorlen_s(errno_t errnum);
```

Note that for SHARC+ cores, the char-size-8 and char-size-32 implementations invoke constraint handlers in the appropriate addressing space and should be registered separately.