SPNU595 describes the Flash API V1.53. But the current version of Flash API is V1.54. SPNU595 is not updated yet to reflect the details of Flash API V1.54. TI will update SPNU595 soon. Until then, this document can be referred by users to know the details of V1.54.

Below are the changes in Flash API V1.54 compared to that of V1.53.

1. Enhancement: Fapi_issueProgrammingCommand() function is updated to block programming ECC for the link-pointer locations in F2837xD devices. If user chooses the Fapi_AutoEccGeneration or Fapi_DataAndEcc programming mode for programming link-pointer locations, Fapi_issueProgrammingCommand() function will modify the command as Fapi_DataOnly mode. If user chooses the Fapi_EccOnly programming mode for programming link-pointer locations, then the Fapi_issueProgrammingCommand() function will return an error (Fapi_Error_FeatureNotAvailable). For programming link-pointer locations, users should only use Fapi_DataOnly programming mode.

2. Enhancement: Flash API V1.54 is compiled with build option "Place each function in a separate subsection". Hence, when the application is compiled with the Flash API V1.54, only the API functions that are used by the application will be included in the coff file and hence users might notice smaller coff file size compared to that of using V1.53.

3. Bug fix: In V1.53, the Fapi_issueProgrammingCommand() function when used with Fapi_DataAndEcc and Fapi_EccOnly programming modes allows only 128-bits and/or its corresponding ECC to be programmed in F2837xD devices. In V1.54, Fapi_issueProgrammingCommand() function is updated to support programming even 64-bits of main-array data and/or corresponding ECC when using Fapi_DataAndEcc and Fapi_EccOnly programming modes. It means that V1.54 allows to either program 64-bits/ECC at a time or 128-bits/ECC at a time for F2837xD devices. For Concerto devices, this feature already exists.

Below are few other clarifications that will be included in the next version of API reference guide.

1. EALLOW has to be executed before using Flash API functions for C28x devices.

2. Flash bank-width in F2837xD and Concerto devices is 128 data-bits+ 16 ECC-bits.

3. Fapi_CalculateEcc function needs a byte address (left shift the C28x Flash address by 1 position before passing it to this function).

4. A note regarding Fapi_AutoEccGeneration mode usage: Fapi_AutoEccGeneration mode will program the supplied data portion in Flash along with automatically generated ECC. The ECC is calculated for the data width of the bank and data not supplied is treated as 0xFFFF. Note that there are practical implications of this when writing a custom programming utility that streams in the output file of a code project and programs the individual sections one at a time into flash. If a 64-bit word spans more than one section (i.e., contains the end of one section, and the start of another), values of 0xFFFF cannot be assumed for the missing data in the 64-bit word when programming the first section. When you go to program the second section, you will not be able to program the ECC for the first 64-bit word since it was already (incorrectly) computed and programmed using assumed 0xFFFF for the missing values. One

way to avoid this problem is to align all sections linked to flash on a 64-bit boundary in the linker command file for your code project.

For example, like this:

```
SECTIONS
{
  .text    : > FLASH, PAGE = 0, ALIGN(4)
  .cinit   : > FLASH, PAGE = 0, ALIGN(4)
  .const   : > FLASH, PAGE = 0, ALIGN(4)
  .econst  : > FLASH, PAGE = 0, ALIGN(4)
  .pinit   : > FLASH, PAGE = 0, ALIGN(4)
  .switch  : > FLASH, PAGE = 0, ALIGN(4)
}
```

If you do not align the sections in flash, you would need to track incomplete 64-bit words in a section and combine with the words in other sections that complete the 64-bit word. This will be difficult to do so it is recommended to align your sections on 64-bit boundaries.