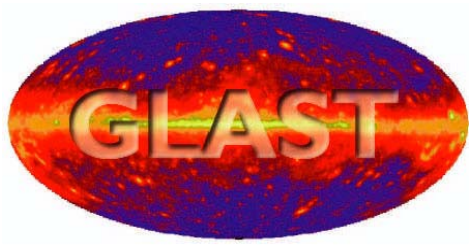

LAT Flight Software

SBC Tools User Manual

Type:	User Manual
Version:	V1-0-0
Author:	D.L. Wood
Created:	8 June 2004
Updated:	14 June 2004
Printed:	14 June 2004

Manual for the GLAST LAT Flight Software Secondary Boot Code support tools.

Contents

0 Introduction

The LAT flight software Secondary Boot Code requires a set of ground-based tools to produce the file products used by the flight portion of the software. These file products are transferred to the LAT on-board storage by the normal file upload process. The tools to produce these file products are contained in the same CMX package, *SBC*, as the flight side target software which consumes these products. In general, two types of file products are needed to operate the flight software secondary boot code process: application software module files and application configuration database files.

0.0 Application Software Modules

One of the primary responsibilities of the flight software secondary boot code is the loading and initialization of the application level software object modules. These fall into three categories:

1. The VxWorks RTOS executable
2. The secondary boot executable
3. The application libraries and executables

The first two types are really handled by the boot process, and are somewhat mislabeled as “application”. On the ground side, however, they share many of the same features as the true application modules. Therefore, they are handled by the same set of tools. The *SBC* tools take the binary output of the CMX build process for a flight constituent and convert it into a form suitable for use by the flight secondary boot module loader and the file upload process.

0.1 Application Database File

The flight side secondary boot code for each flight CPU requires that an application configuration database has been prepared for that target. The secondary boot code will parse this database file to know which application software configuration to run. In general, the application software configuration for a target consists of two items:

1. A list of software object modules to load from the on-board EEPROM file systems
2. A list of application initialization functions to call and the parameters that should be passed to them.

The flight secondary boot code expects to find the application database in a compact binary format on the target. The *SBC* tools allow users to create this binary format file from an XML source specification.

0.2 Reference Documents

Secondary Boot Code, LAT Flight Software Design Description Document.

Primary Boot Code, LAT Flight Software Design Description Document.

ZLIB User Manual, LAT Flight Software User Manual.

FILE User Manual, LAT Flight Software User Manual.

CMX User Manual, LAT Flight Software User Manual.

1 Application Object Module Tools

The script *sbc_make* in package *SBC* takes the binary products of the CMX build process and outputs a set of files which is both ready for the LAT flight file upload process and is ready for loading by the secondary boot code module loader.

1.0 Command Syntax

The *sbc_make* tools supports a small number of commands. The command name is always the first parameter to the tool.

1.0.0 sbc_make file

The sbc_make file command is the primary command of the tool. It takes a CMX constituent and converts the binary object file into a form suitable for use in the secondary boot code. It also produces a set of intermediate files, all of which are saved in the output directory along with the actual upload product file.

```
sbc_make file <package> <constituent> [--name=<mnemonic>]
      [--key=<key_num>]
```

The *package* and *constituent* parameters name the CMX constituent to work with. The CMX environment settings are queried to determine the current branch to use, so to work with a version other than the site production version, issue a “cmx set branch” command before running the *sbc_make* tool. The *rad750* tag for the constituent is always used. The behavior of the tool differs depending on whether the package branch is public or private. The optional *key* and *name* qualifiers allow the user to set these members in the file header which is prefixed to the output file.

A set of four files is produces when this command is run. The four files are:

1. A copy of the constituent object module file as produced by the CMX build process.
2. A stripped version of the object module. For RTOS images, this is the straight binary representation of the object. For all other constituents, this is an ELF object module file which has been stripped of all unnecessary sections and symbols.
3. A compressed version of the stripped object module. The simple ZLIB compression format is used, where the entire file is compressed as one block.
4. The compressed module with a LAT file header inserted at the beginning.

Only the last file of the set is needed for loading on the flight CPU target; the other files are collected in the directory for reference.

1.0.1 sbc_make link

The sbc_make “link” command allows users to set the link for a constituent in the public output directory tree (see Section 1.1.0).

```
sbc_make link <package> <constituent> --version=Vx-x-x|dev
```

The *package* and *constituent* parameters determine which constituent link will be modified. The required *version* qualifier indicates to which public version of the constituent the new link should refer. This command allows users to work with an older version of a package without re-generating the file set.

1.1 Output Directories

Besides producing the upload product files, the *sbc_make* tool provides a simple directory tree structure for organizing and maintaining the file sets.

1.1.0 Public Directory Structure

When a public branch of the CMX package is in effect for the user environment, the file output set is output to a directory tree in the *${SBC_C_SBD}* directory. The location of the directory is completely governed by the *SBC_C_SBD* environement variable setting.

${SBC_C_SBD} Directory

|----------Package

|----------Package Version

|----------Constituent

The *${SBC_C_SBD}* directory will also contain a subdirectory named “links”. This directory contains a set of symbolic links to the upload product files. Each links is named after the constituent which it represents. When running the *sbc_make* “file” command, the link for a constituent is set to the latest output file. Running the *sbc_make* “link” command will set the link to one of the other versions for that constituent, if they exist.

It is intended that the *${SBC_C_SBD}* directory be global for a CMX site. This directory would then contain the upload file sets for the site’s public branches of packages. For testing and other purposes, however, users may override the *SBC_C_SBD* environment setting and point the file output to some private location.

1.1.1 Private Directory Structure

When a private (test) branch of the CMX package is in effect for the user environment, the file output set is output to a directory tree in the *${HOME}/SBC* directory.

${HOME}/SBC Directory

|----------Package

|--------------------Constituent

|-------------------------Timestamp

The “timestamp” directory is created using the host current time. By using a timestamp in the directory path, it is possible to accumulate multiple product sets for the same constituent. The *${HOME}/SBC* directory will also contain a subdirectory named “links”. This directory contains a set of symbolic links to the upload product files. Each links is named after the constituent which it represents. Unlike the public links directory, users are responsible for setting the link themselves if they want to run a back version of the constituent.

1.2 Continuing Evolution

- The *sbc_make* tool stops one step short of producing the code module file format which will be handed to the MOC. This is fairly straightforward to do, but requires explicit knowledge of the flight target storage location where the file will be loaded.

2 Application Configuration Database Tools

The script *sbx* in package *SBC* takes an XML specification of the application configuration as input and produces a binary database file as output. The output file is suitable for use with the flight software secondary boot code.

2.0 Command Syntax

The *sbx* command line syntax is very straightforward.

```
sbx <xml_file> <bin_file> [--name=<mnemonic>] [--key=<key_num>]
```

The *xml_file* parameter refers to the input XML file, whose format is described in Section 2.1. The *bin_file* parameter gives the output path and name of the binary database file. In total, three files will be output along side one another:

1. A binary application database file.
2. A compressed version of the binary database file. The simple ZLIB compression format is used, where the entire file is compressed as one block.
3. The compressed binary database with a LAT file header inserted at the beginning.

The optional *name* and *key* qualifiers allows users to set these members in the file header which is prefixed to the final file of the output set.

2.1 XML Elements

This section describes the XML element hierarchy which is assumed for the XML input files to the *sbx* tool. At the top level, the XML must contain the following elements:

```
<SBCdoc  modified="date" created="date">
  <application_database version="1">
        <modules>
              .  .  .
              .  .  .
        </modules>

        <functions>
              .  .  .
              .  .  .
        </functions>
  </application_database>
</SBCdoc>
```

The *version* attribute in the *<application_database>* element refers to the binary format version number which will be inserted into the output database file. Currently, only version '1' is supported.

Each *<modules>* section may contain one or more *<module>* elements (see Section 2.1.0). Each *<functions>* section may contain one or more *<function>* elements (see Section 2.1.1).

2.1.0 Module Elements

Each *<module>* record describes one application object module to be loaded by the secondary boot code. Each object module is assumed to correspond to exactly one CMX constituent. The format of each *<module>* record is shown below.

```
<module key="n">
  <package> PKG  </package>
  <constituent> constit </constituent>
  <version> vers </version>
  <device> dev </device>
  <directory> dir </directory>
  <file> fil </file>
</module>
```

The <*device*>, <*directory*>, and <*file*> elements describe completely the storage location of the object module file on the target CPU. The <*device*> value should be either '*ee0*' or '*ee1*' to indicate in which non-volatile flight file system the file is stored.

The <*package*>, <*constituent*>, and <*version*> elements list the CMX attributes of the constituent to use. The *key* attribute value should provide the file master key number from the ground LAT file database. Note that this information is not actually compiled into the binary database output since it is of no actual use to the secondary boot code. The elements are required, however, to make sure the XML record of the application configuration is precise.

2.1.1 Function Elements

Each <*function*> record describes one application initialization function to be called by the secondary boot code. The function symbol name must be defined in the target CPU system symbol table. Possible sources of the functions are one of the public VxWorks RTOS system functions, one of public secondary boot code executable functions, or one of the public application module functions. The format of each <*module*> record is shown below.

```
<function name="NAME">
  <parameter number="n"> param </parameter>

  . . .

</function>
```

The *name* attribute of the <*function*> element provides the symbol name of the function to call. Up to four <*parameter*> elements may be provided. Each one is a decimal 32-bit value which is passed to the application initialization function. The *number* attribute should be '0' through '3', and is the position index in the function parameter list. If no parameter value is provided for one of the four slots, a value of '0' will be compiled into the binary database output.

2.2 XML Example

The example below shows an sbx XML file which was used for testing at one point.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE SBCdoc SYSTEM "D0-0-0.dtd">


<SBCdoc modified="4/16/04" created="4/16/04">

  <application_database version="1">

        <modules>

              <module key="1">
                    <package> CMX </package>
                    <constituent> cmx_asBuiltSpy </constituent>
                    <version> V2-0-8 </version>
                    <device> 2 </device>
                    <directory> 1 </directory>
                    <file> 0 </file>
              </module>

              <module key="2">
                    <package> PBS </package>
                    <constituent> pbs </constituent>
                    <version> V2-2-0 </version>
                    <device> 2 </device>
                    <directory> 1 </directory>
                    <file> 1 </file>
              </module>

              <module key="3">
                    <package> MSG </package>
                    <constituent> msg_mt </constituent>
                    <version> V1-1-4 </version>
                    <device> 2 </device>
                    <directory> 1 </directory>
                    <file> 2 </file>
              </module>

              <module key="4">
                    <package> MSG </package>
                    <constituent> msg_print </constituent>
                    <version> V1-1-4 </version>
                    <device> 2 </device>
                    <directory> 1 </directory>
                    <file> 3 </file>
              </module>

              <module key="5">
                    <package> CCSDS </package>
                    <constituent> ccsds_pkt </constituent>
                    <version> V3-2-1 </version>
                    <device> 2 </device>
                    <directory> 1 </directory>
                    <file> 4 </file>
              </module>
```

```
                <module key="6">
                      <package> CTDB </package>
                      <constituent> co1553_rt </constituent>
                      <version> V4-0-2 </version>
                      <device> 2 </device>
                      <directory> 1 </directory>
                      <file> 5 </file>
                </module>

                <module key="7">
                      <package> CTDB </package>
                      <constituent> sumt_rt_sib </constituent>
                      <version> V4-0-2 </version>
                      <device> 2 </device>
                      <directory> 1 </directory>
                      <file> 6 </file>
                </module>

                <module key="8">
                      <package> FILE </package>
                      <constituent> file_upl </constituent>
                      <version> V1-0-1 </version>
                      <device> 2 </device>
                      <directory> 1 </directory>
                      <file> 7 </file>
                </module>

                <module key="9">
                      <package> FILE </package>
                      <constituent> file_path </constituent>
                      <version> V1-0-1 </version>
                      <device> 2 </device>
                      <directory> 1 </directory>
                      <file> 8 </file>
                </module>

                <module key="10">
                      <package> FILE </package>
                      <constituent> fule_sys </constituent>
                      <version> V1-0-1 </version>
                      <device> 2 </device>
                      <directory> 1 </directory>
                      <file> 9 </file>
                </module>

                <module key="11">
                      <package> LFS </package>
                      <constituent> lfs_rt </constituent>
                      <version> V0-0-0 </version>
                      <device> 2 </device>
                      <directory> 1 </directory>
                      <file> 10 </file>
                </module>

          </modules>
```

```
        <functions>

              <function name="CMX_asBuiltPrint">

              </function>

              <function name= "rt_init">
                    <parameter number="0"> 1048576 </parameter>

              </function>

              <function name="rt_start">

              </function>

              <function name="upl_info">

              </function>

              <function name="rt_info">

              </function>

        </functions>

  </application_database>


</SBCdoc>
```

2.3 Continuing Evolution

- The *sbx* tool stops one step short of producing the application database file format which will be handed to the MOC. This is fairly straightforward to do, but requires explicit knowledge of the flight target storage location where the file will be loaded.
- Currently, the XML source input file must be created either by hand or by some other tool. It remains to be seen whether it is advantageous for the *sbx* tool itself to be able to provide a user-friendly interface to accomplish this task.