GLAST Large Area Telescope
Monthly Mission Review

LAT Flight Software Status

October 5, 2007

Jana Thayer

Stanford Linear Accelerator Center
FSW Status

• B1-0-2 progress
  – Remaining effort:
    • Testing of additional housekeeping information
    • Testing of filter statistics for non-gamma filters
    • Miscellaneous IVV4 fixes (new)
    • LIM/GRBP bug fixes (new)
    • Addition of stall after GARC LAM (new)
  – Roll build on 10/12
  – Two weeks of testing (see next slide for details)
  – Ready for upload, 10/26

• B1-0-2 does not contain any large perturbations. Targeted changes included in this build only affect a few packages:
  – Low rate science counters can be routed to SDI
  – Filter statistics added to science stream to monitor performance of onboard event filter
  – Bug fixes to GRB algorithm
  – Tweaks to PIG/LIM to adjust delays and fix bugs
  – Report HSK information on LAT power and LAT physics acquisition
B1-0-2 regression testing (10/12 - 10/26)

- Run FSW unit tests on all FSW packages that have been changed, (LHK, GRBP, etc.), prior to release of code
  - Where changes to code were made, do side-by-side comparison of code performance before and after to verify that output or performance is identical (except where bug fixes change performance)
- Run FQT regression test suite against final build on testbed (3 days)
  - Testing includes all packages, even those that haven’t changed
  - Verifies that all of FSW requirements are met
  - Compare results of B1-0-1 testing to B1-0-2
- Run selected portions of CPT using LICOS on testbed (5 days)
  - Verifies that LICOS Scripts have kept pace with FSW additions/changes
  - Reassurance that there will be no surprises during testing on the LAT
- Use PROCs to perform LEO power up sequence and exercise nominal data taking on testbed (1 day)
  - Verifies that PROC performance unaffected
  - NOTE: Inspection of B1-0-2 changes to dbx shows that PROCs should work identically in B1-0-1 and B1-0-2
- Upload to LAT and run selected portions of CPT in config 1 (~4 hours)
- If anything slips through this testing, it will be caught by the subsequent LAT CPT
JIRA Metrics as of 4 October 2007

- Open issues are divided as follows
  - 29 items planned for B1-0-2 (13 individually tracked housekeeping additions)
  - 1 item planned for B1-0-3
  - 1 awaiting FSW CCB adjudication

- Note: does not include candidate post-launch items (i.e., “Deferred”, “B2-0-0”)
Solaris 9 → Solaris 10 transition

**SCCS upgraded the public Solaris servers from solaris 9 (w/ gcc-3.2.3) to solaris 10 (w/ gcc-3.4.3)**

- As a result, numerous warnings affecting about a dozen packages are produced during FSW builds on the Sun where the majority of software unit testing is done.
- Compilation warnings derive from compilation on Sun, do not affect flight binaries
  - Although the gcc compiler has changed, the cross compiler that actually produces the flight code has not. Consequently, the move to gcc-3.4.3 alone has no direct effect on flyable code.
  - These warnings can be eliminated by rephrasing the affected code.
  - These changes are purely syntactical in nature and have no effect on the coding logic.

**Consequence of ignoring such warnings**

- Warnings and errors during compilation are indications of potential bugs or problems.
- Releasing a FSW build with warnings during compilation is a violation of the rules in the flight software management document
- If the warnings are not eliminated, the noise generated by a build will make it significantly more difficult to identify substantive issues
- New compiler releases may improve optimization of code
Proposed solution for B1-0-2+

- Complete B1-0-2 without making solaris-inspired changes
- Test against testbed, upload to LAT, regression test on LAT as planned
- Make controlled changes to each of the affected packages, as necessary
  - Changes made by experienced programmers, familiar with the code
  - Developers will assess each potential change intelligently and decide how to address the warning
- Perform unit tests on affected packages
- Release changes as B1-0-3
  - Regression test on testbed (FQT, CPT, PROC validation)
  - Compare performance to B1-0-2
  - If performance is identical, make B1-0-3 available for upload
    - Either find the time to upload B1-0-3 to the LAT
    - Wait until the next requested bug fix forces a build (B1-0-4) or B2-0-0 (post-launch) and incorporate/upload the changes as part of that build
Validating code produced with gcc 3.4.3

How do we get assurance that code is functionally identical to previous version? What could go wrong?

- Introduce a syntactical error or typo
  - Most likely caught by compiler, warning becomes an error
- Change functionality/logic of code
  - Error caught by FSW unit tests (target to test changed code)
  - Second line of defense: FQT and/or LICOS CPT
  - Finally, PROCs can be re-validated against new build on testbed, to reassure ourselves that build behavior has not changed

Gain additional reassurance with extended runs on testbed

- Stress test software by running extended nominal data-taking run
- Vary data size and event filter configuration
- Monitor filter performance, CPU load, memory usage, and other parameters to ensure nominal performance
Future transitions

• Cross-compiler that produces flight code is frozen
• ISOC owns 4 Suns (that came with solaris 10) for maintenance and compilation of FSW builds
  – Currently, these machines are maintained by SCCS
  – Patches installed when available
  – FSW/ISOC can work with SCCS on timing of updates
• Avoid being “surprised” by future transitions:
  – Strengthen relationship with SCCS
  – New versions of solaris are released approximately every 2 - 3 years
  – Decide on case-by-case basis whether to change code
• Would such a change trigger the upload of a new FSW build in the future?
  – No, this would not be the driver for a new build
  – However, any approved changes would be incorporated into the next FSW build, triggered by a bug fix or improvement
Implications

• Implication of not moving code forward with OS upgrades
  – Live with warnings
    • Forces FSW to sift through a lot of noise to find real problems
    • Ignore increasingly more sophisticated compiler warnings resulting in un-optimized code
  – If we choose to change the code to eliminate the warnings, as appropriate, we need to ---
    • Unit test the modified code
    • Regression test build on testbed, as usual
    • Upload at next reasonable opportunity

• FSW will not attempt to carry a code branch
  – A code branch defeats all of the tools we have developed to manage code versions
  – No resources to do this bookkeeping
  – Risk uploading the wrong code modules or releasing incompatible versions of packages