STEREO
IMPACT

IDPU Flight Software Development Plan

IDPU_SDP_A.doc
Version A – 2001-Aug-20

David Curtis, UCB IMPACT Project Manager

Document Revision Record

Rev.	Date	Description of Change	Approved By
A	2001-Aug-20	Preliminary Draft	-

Distribution List

Dave Curtis, UCB
Harry Culver, GSFC

Table of Contents

1. Overview

1.1. *Introduction*

IMPACT consists of a number of instruments connected to the spacecraft via the IDPU. The IDPU contains, amongst other things, a Data Controller Board (DCB) on which there is a microprocessor. This microprocessor controls all data flow between the instruments and the spacecraft, including commands, telemetry, timing, and status. This document describes the UCB plan for developing flight software for this processor.

UCB has had experience with similar flight software tasks on HESSI, Lunar Prospector, FAST, Cluster, Mars Global Surveyor, Wind, Polar, etc.

1.2. *Document Conventions*

In this document, TBD (To Be Determined) means that no data currently exists. A value followed by TBR (To Be Resolved) means that this value is preliminary. In either case, the value is typically followed by a code such as UCB indicating who is responsible for providing the data, and a unique reference number.

1.3. *Applicable Documents*

The following documents include drawings and STEREO Project policies, and are part of the Interface Requirements. In the event of a conflict between this SDP and the following documents, this SDP takes precedence. All documents can be found on the Berkeley STEREO/IMPACT FTP site:

http://sprg.ssl.berkeley.edu/impact/dwc/

1. Specifications/IDPUSoftwareRequirements
2. PLASTIC Software requirements
3. Specifications/IDPUSpec (IDPU Data Controller Board Specification)
4. IMPACT_rev_b_distrib16Aug01 (IMPACT/Spacecraft ICD, on the APL web page)
5. ICD/Impact Serial Interface (Instrument Serial Interface Specification)
6. Plans/STEREO-IMPACT-PAIP (Performance Assurance Implementation Plan)
7. TBD – (IDPU DCB Description)
8. Specifications/IMPACTPerformanceSpec
9. Plans/IMPACTCMPlan (Configuration Management Plan)

1.4. *Acronyms*

CDR	Critical Design Review
CSCI	Computer Software Configuration Item
DCB	Data Controller Board
GSFC	Goddard Space Flight Center
ICD	Interface Control Document
IDPU	Instrument Data Processing Unit
IMPACT	In-situ Measurements of Particles and CME Transients

MAG	Magnetometer instrument
PDR	Preliminary Design Review
PLASTIC	PLasma and SupraThermal Ion Composition instrument
PER	Pre-Environmental Review
SDP	Software Development Plan
SEP	Solar Energetic Particle instrument suite
SSL	Space Sciences Laboratory
STE	SupraThermal Electron instrument
SWAVES	STEREO Waves instrument
SWEA	Solar Wind Electron Analyzer
TBD	To Be Determined
TBR	To Be Resolved
UCB	University of California at Berkeley
UNH	University of New Hampshire

2. System Requirements and Software interfaces

2.1. *System Overview*

The IMPACT instrument suite consists of 9 of instruments connected to the spacecraft via the IDPU as shown in Figure 2.1-1. Note that the IDPU serves as the processing element for the PLASTIC instrument as well as the IMPACT suite. The SEP suite has its own processing element, so the IDPU is primarily a bent pipe for commands and telemetry between SEP and the Spacecraft. The rest of the instruments rely on the IDPU to perform instrument operations and data compression and formatting.

The IDPU hardware is described in reference 3.

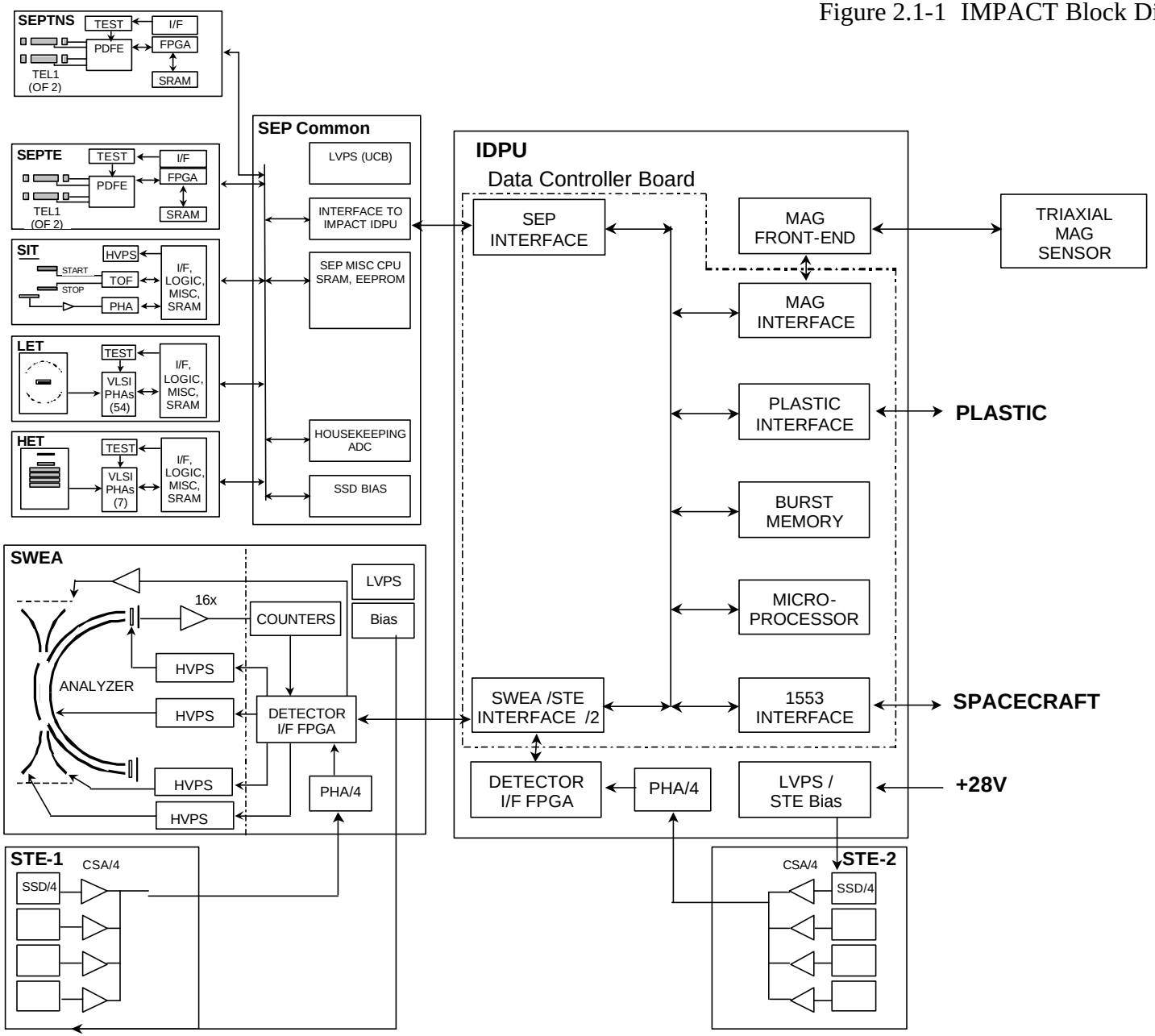
2.2. *Top Level Requirements*

The IMPACT IDPU provides the interface between the STEREO Spacecraft C&DH system and the IMPACT and PLASTIC Instruments. The IDPU software runs on a microprocessor in the IDPU, on the Data Controller Board (DCB). All information transfer between the IMPACT/PLASTIC instruments and the Spacecraft/Ground, including telemetry, commands, and status flow through the IDPU. The IDPU performs instrument control functions, telemetry compression and formatting, and system monitoring/safing functions.

The IDPU Flight Software requirements are described in reference 1 and 2.

There are two pieces of flight code to develop: The boot code resident in PROM and the operational code residing in EEPROM. The boot code operates at processor reset. It determines if good code exists in EEPROM, and then transfers control to it. It shall be capable of reloading the EEPROM from ground command via the spacecraft C&DH. Most of this document applies to both Boot and Operational code, except where called out.

Figure 2.1-1 IMPACT Block Diagram



2.3. *Resorce Requirement Estimates*

Table 2.2-1 lists the estimated software resource requirements extracted from these requirements and based on previous code and/or analysis.

Table 2.2-1 IDPU Flight Software Resource Requirements Estimates

Task	Processor Cycles, %	Code Size, kbytes	Buffer Memory, kbytes
Service Software:			
Operating System	1%	0.3 (b)	32.0
1553 Interface	1%	3.0 (b)	8.0
Command Router	<1%	0.7 (b)	10.0
Telemetry Packet Queue	<1%	0.3 (b)	10.0
Burst System	1%	1.0	1500.0 (*)
Serial Instrument Interfaces	1%	0.5	75.0
Housekeeping	<1%	0.2 (b)	0.6
Beacon Telemetry	<1%	0.2	0.6
Event Logging	<1%	0.3 (b)	0.6
Safing	<1%	0.5 (b)	0.3
MAG Software	1%	0.5	1.2
SEP Software	1%	1.0	2.5
SWEA Software:			
SWEA Moments	20%	1.0	17.0
SWEA Distributions	4%	2.5	10.8
SWEA Misc.	1%	1.0	2.5
STE Software	1%	1.0	3.7
PLASTIC Software:			
PLASTIC Distribution	6%	2.0	185.6
PLASTIC PHA	5%	0.5	46.6
PLASTIC Moments	5%	1.0	45.6
PLASTIC Misc.	1%	1.0	2.5
Total	**55%**	**18.0**	**1955.1**
Available	100%	32 (**)	2048
% Usage	55%	56%	95%

(*) Nominal value. Burst memory will expand to use all unallocated memory space.
(**) 32K is maximum code that can be paged into memory at one time. Paging code in and out is a complication, but possible.
(b) To be included (at least in part) in Boot PROM. PROM = 8kbytes, usage = 66%

2.4. *External Interfaces*

The IDPU interfaces to the Spacecraft via a 1553 bus as described in the Spacecraft / IMPACT ICD (reference 4). The SWAVES interface is also via the spacecraft 1553 interface. The ICD is generated by APL, and will be signed off at PDR.

The IDPU interfaces to the 5 instrument groups are via Serial Instrument Interfaces described in reference 5 (the groups are: MAG, SEP (SIT, HET, LET, SEPT-NS, SEPT-E), SWEA/STE-D, STE-U, and PLASTIC). This interface is defined by UCB and shall be signed off at PDR.

2.5. *Internal Hardware/Software Interfaces*

The IDPU Hardware is described in reference 3. This document is generated by UCB. A lower-level description of the function of each bit of each register is given by Reference 7, generated by the DCB designer (Elf) by 5/02.

2.6. *Interaction between UCB and Instrument Teams*

UCB is responsible for the Serial Instrument Interface spec (reference 5) that describes the hardware interface between the IDPU and the instruments.

UCB shall write the Software Requirements Document that describes the software tasks performed by the IDPU for the following instruments:

MAG (Acuna, GSFC)
SEP (Tycho, GSFC; Cummings, Caltech)
SWEA (Larson, UCB)
STE-U, STE-D (Larson, UCB)

UNH shall write the software requirements Document that describes the software tasks to be performed for the PLASTIC instrument.

A software requirements review shall be held to confirm that the instrument software requirements are adequately described in the documents.

The instrument teams shall be involved in the acceptance testing and subsequent system testing of the software with their instruments.

3. Software Development

3.1. *Requirements Definition and Analysis Phase*

Based on experience with similar instruments flown previously, and after discussions with the lead scientists and engineers from each instrument, UCB has developed a set of requirements documented in Reference 1. Similarly PLASTIC requirements were generated by UNH and iterated with UCB as documented in Reference 2. Software requirements shall relate to both the instrument performance requirements as called out in reference 8, as well as the lower level functional requirements associated with the instrument implementation. The Requirements Documents shall also include derived

requirements related to the hardware and software implementation adequate to fully specify the software to be coded. Based on these requirements, software resource requirements have been analyzed, and are shown in table 2.2-1.

3.2. *Design*

A top-down approach shall be used to partition the code into modules involving minimal interaction. It is desirable that the individual instrument modules shall be as independent as possible to simplify operations. The requirements described in reference 1 and 2 already reflect this approach. Following that point, a more bottoms-up approach will be taken to each module. The systems modules will be defined first, from which the instrument module interfaces will be derived.

No formal design products are planned, but various pseudo-code, flow chart, and other products shall be used as needed, generally documented in the Software Development Log.

3.3. *Coding*

The code is planned to be written in assembly language. A modular, structured approach shall be used. Good coding practices shall be used in the development, such as:

- Software shall be developed by a programmer with detailed system design, hardware design, and science instrument knowledge.
- Code shall be structured into source code modules
- Modules shall be divided into functional subroutines
- Subroutines shall be no longer than a page of two of source code.
- Minimal use shall be made of global variables; interactions between subroutines and modules shall be well defined and use a minimum number of common variables
- Modules and subroutines (and their interfaces) shall be self-documenting
- End conditions on loops shall be well defined

3.4. *Development Environment*

Software shall be developed using the Phyton 80C196 cross-development package on a PC. This package includes an assembler, linker, and simulator; the simulator shall be used for early module-level testing. Once an ETU is available, most testing will take place using it. The final flight software acceptance test will be performed using one of the flight IDPUs.

A PROM-emulator shall be used to allow quick turn-around in downloading and testing code. Later, when the boot PROM is in place, code will be downloaded to the EEPROM either via the spacecraft interface or via a diagnostic serial port.

The APL Emulator shall be used to emulate the spacecraft interface to the ETU and later to the flight IDPU. The emulator shall be attached to a PC running the Command & Telemetry GSE software to send commands and display telemetry data. Later science display GSE shall be added to display more complex science telemetry. Since the APL

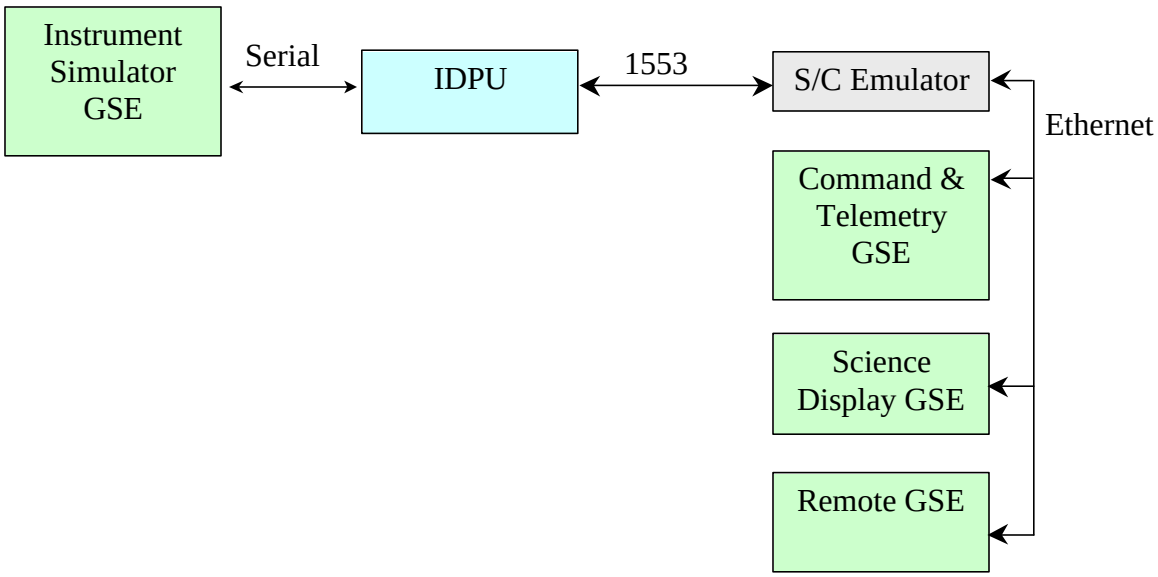
Emulator connects to the GSE via Ethernet, the GSE may be remotely located, facilitating tests with remote instrument teams.

UCB shall develop both an IDPU emulator to help in development of the instruments, and an instrument emulator to aid in testing the IDPU and IDPU flight software. The PLASTIC team plans on providing a better fidelity PLASTIC emulator to aid in IDPU PLASTIC flight software development.

A logic analyzer and digital oscilloscope shall be available for diagnosing subtle problems.

Diagnostics shall be primarily via the telemetry interface – this ensures that the telemetry interface is adequate for in-flight diagnostics. Access to the serial interface for diagnostics shall be limited. Likewise use of a logic analyzer for diagnosing problems shall be limited.

All equipment is available at this time except the APL emulator software, the Command & Telemetry GSE software, and the instrument emulators. These shall be developed during the course of the program.



3.5. ***Product Assurance***

3.5.1. Walkthroughs

A walkthrough of the critical code (boot PROM and safing code) shall be performed during an internal review some months before acceptance testing of the flight code (see

section 4.7 below). Project personnel will be invited to this review, in addition to IMPACT team members not involved in the writing of the code. The intent of the review is to get independent verification that the code and the acceptance test plans are adequate. Only the critical code is targeted for walk-through, since the remaining code can be modified as required after launch.

3.5.2. Test Plan

Software tests will start at the module level. As the code builds up, system-level testing will start, and the majority of the test time will be targeted at the system level.

A formal Acceptance Test will be performed on the final build flight software prior to start of IDPU functional tests. During environmental tests and suite level testing, more experience and test time with the flight software with real instrument data shall be gained, and changes are expected. Any subsequent changes in the flight software will go through the similar acceptance testing. The test shall be designed to verify each of the software functional requirements as called out in the requirements documents. The test plan shall be reviewed internally prior to the start of testing; Project participation at this review is welcome.

Results of all testing will be documented in the Software Development Log.

3.5.3. Configuration Management / Backup Plan

The IMPACT Configuration Management Plan (reference 9) covers the basics of the CM including that used for software. Because of the small development team involved (1 person), formal configuration control of the software shall only start with the first delivery of code to acceptance testing. Prior to that time, the Software Development Log shall be used to track changes, problems, etc.

Once under control, each version of software shall be archived along with its documentation. Changes shall be fully documented in the code as well as in the Software Development Log. Changes will be signed off by all relevant parties (typically the Programmer, PM, and appropriate instrument lead(s), and an acceptance test shall be run prior to installation of any new version in the flight hardware.

Code shall be developed on a computer at SSL that is backed up daily.

3.5.4. Problem Reporting and Tracking

Prior to installation in flight hardware, problems shall be documented and tracked in the Software Development Log. Once the software has passed acceptance tests and is installed in flight hardware, the common problem reporting and tracking system used by the flight hardware will be used, as described in the PAIP (reference 6).

3.5.5. Risk Assessment

The margins shown in table 2.2-1 (based on conservative estimates) are adequate but not excessive, due to cost/mass/power limitations. Should the code grow to where margins

become small, unexpected effort will be required to contain growth (more optimization of coding).

The programmer resources indicated in 4.6 and 4.7 are somewhat lean, but comparable to other recent flight software development efforts. Loss of the single programmer late in the development would have significant schedule impacts.

To remain within resource margins, requirements growth through science creep must be resisted.

3.6. *Software Maintenance*

After the instrument is delivered, the ETU together with the supporting GSE shall be maintained at UCB. Any problems can be recreated and diagnosed in this environment. Code changes shall be run through an acceptance test prior to loading into the flight hardware. Every effort will be used to keep the flight software programmer available (though probably on another program) to make any necessary changes through the life of the program. Software documentation should be adequate to allow another programmer to understand and make changes to the software if necessary.

4. Management Plan

4.1. *Build Plan*

Code shall be developed in three builds. The first build shall be designed to test the ETU processor hardware. The second build will be used to test the instrument suite. It may be missing some of the data products and automatic functions, but shall be adequate to verify correct instrument operations. The boot PROM code shall be derived from the second build. The third build shall be for flight and shall be fully functional.

4.2. *Reviews*

- An informal internal Software Requirements Review shall be held with the science team to ensure that the documented requirements are complete and clear.
- At Instrument CDR, the Software Design shall be covered
- An informal internal Critical Software Walkthrough / Acceptance Test Plan Review shall be held a few months before the Acceptance Tests are scheduled.

The margins shown in Table 2.2-1 are unlikely to change prior to PDR. By CDR, some code shall have been written (Build one), and better estimates shall be available. At that time, more than 75% usage (Code or processing cycles) would indicate a problem. At Acceptance Test, more than 90% usage would be cause for concern, leaving little room for future growth.

4.3. *Documents & Source Code*

The IDPU Flight Software shall be documented by:

- This Software Development Plan (at PDR)
- Software Requirements (at PDR)

- Software Development Log
- Users Manual
- Source Code & Description

4.4. *Heritage & Reuse*

No code shall be directly reused for the IDPU. Much of the IDPU code design and structure shall be copied from FAST, HESSI, Lunar Prospector, or Mars Global Surveyor (about 25% of the code is directly applicable, and another 50% nearly so; the 1553, SEP. STE, and PLASTIC PHA code are largely new development items).

Table 2.2-1 lists the estimated software modules and resource requirements.

4.5. *Manpower*

The manpower allocated to the IDPU software development is:

FY01:	1mm
FY02:	5mm
FY03:	6mm
FY04:	3mm

Maintenance following delivery of the IMPACT suite to the spacecraft is on an as-needed basis, not included above.

4.6. *Staffing Plan*

The IDPU flight software shall be designed, coded, and tested by David Curtis at UCB. The Data Controller Board hardware is designed by Dorothy Gordon of Elf to a specification developed by David Curtis

4.7. *Schedule*

Software Development Plan	PDR (9/01)
Software Requirements Document(s)	PDR (9/01)
Software Requirements Review	12/01
Software Design Complete, start Coding	1/02
First Build Complete, ETU Available	8/02
Software Design Review (part of CDR)	CDR (11/02)
Critical Code Walkthrough / Acceptance Test Review	6/03
Boot PROM Acceptance Test	8/03
Boot PROM Install in DCB of FM1	9/03
Second Build Complete	9/03
Third Build Complete, Acceptance Test	3/04