Software Development for the Material Science Laboratory on ISS by Automated Generation of Real-time Software from Datasheet-based Inputs

'DASIA 2000'

- Data Systems in Aerospace -

Montreal, Canada

May 22-26, 2000

Michael Birk, Uwe Brammer, Michael Ziegler, Klaus Lattner

Kayser-Threde GmbH

Rainer Gerlich

BSSE System and Software Engineering

KAYSER-THREDE

BSSE System and Software Engineering

Software Development for the Material Science Laboratory on ISS by Automated Generation of Real-time Software from Datasheet-based Inputs

Michael Birk, Uwe Brammer, Michael Ziegler, Klaus Lattner

Rainer Gerlich

Abstract: To meet the technical and managerial challanges of the MSL project it became necessary to find a highly efficient and flexible software development approach. This paper describes this generic approach chosen for MSL software development and the benefits gained by using tools for automatic software generation to an utmost extent. The outputs of the system engineering tasks as e.g. data input – output relations, control commands and software architectural design are maintained in data sheets. With software tools like the ISG from these data sheets the infrastructure and other important parts (e.g. command dispatcher and database) of the software system are automatically generated. This enables the software development team to put emphasis on the development of the lower level, project specific functionality. Changes are performed by updating the datasheets and rerunning the generation.

Keywords: automation of development, incremental development, reusable software, early system validation, standardisation, software interfaces, software integration, distributed systems, software development process, formal system definition, real time software

1. INTRODUCTION

The Material Science Laboratory (MSL) is an ESA funded experiment payload to be operated onboard the International Space Station. It is a system of high complexity as it consists of a variety of subsystems that need to be controlled and which are developed and manufactured by a industrial consortium comprising some 5 companies.

To meet the technical and managerial challanges of the MSL project it is imperative to find a highly efficient and flexible sofware development approach. The contradictory requirements of allowing for changes and modifications in system configuration and also for changes of requirements and of remaining within the budget envelop led us to a generic development approach. The generic approach involves direct generation of most parts of the software from high-level system engineering definitions, from inputs which are available in general and which in this project only needed to be sligthly extended to cover the needs of automated software generation.

In recent projects the systemwide maintenance and usage of knowledge-bases turned out to be a real advantage and had a productive influence on the improvement of generic datasheet driven software development methods.

This led to the idea to use the ISG (Instantaneous System and Software Generation) tool for MSL development and to encourage its upgrade with additional functionality for generation of database, command dispatcher and low level signal conditioning functions .

2. THE MSL PROJECT

The Material Science Laboratory is a furnace that shall allow for melting metal or semiconductor material samples and to investigate the cristallization or solidification process. By means of particular subsystems like a Magnetic Field generator, Peltier Elements, and mechanisms that allow for quenching the samples, the process can be influenced.

The MSL system consists of two SPLC computers, the FCU (Furnace Control Unit), the PSU (Power Supply Unit), and several subsystems controlled either via RS 422 serial interfaces or via I²C bus like serial digital interfaces.

The subsystems controlled by means of a RS 422 interfaces are :

- mass spectrometer to detect toxic contaminations
- magnetic field generator to influence solidification
- turbo pump 1 and 2 to produce high vacuum
- pyrometer to measure heater temperatures
- water pump to provide water cooling
- ultrasonic diagnostics device for experiment diagnostics
- experiment dedicated electronics (e.g. Seebeck Measurement Unit), for experiment diagnostics

Subsystems controlled via digital I²C bus interface are :

- Furnace drive , shear cell drive and regulation valve
- Quench drive
- DC/DC converters
- Power distribution unit
- Heater Current supply

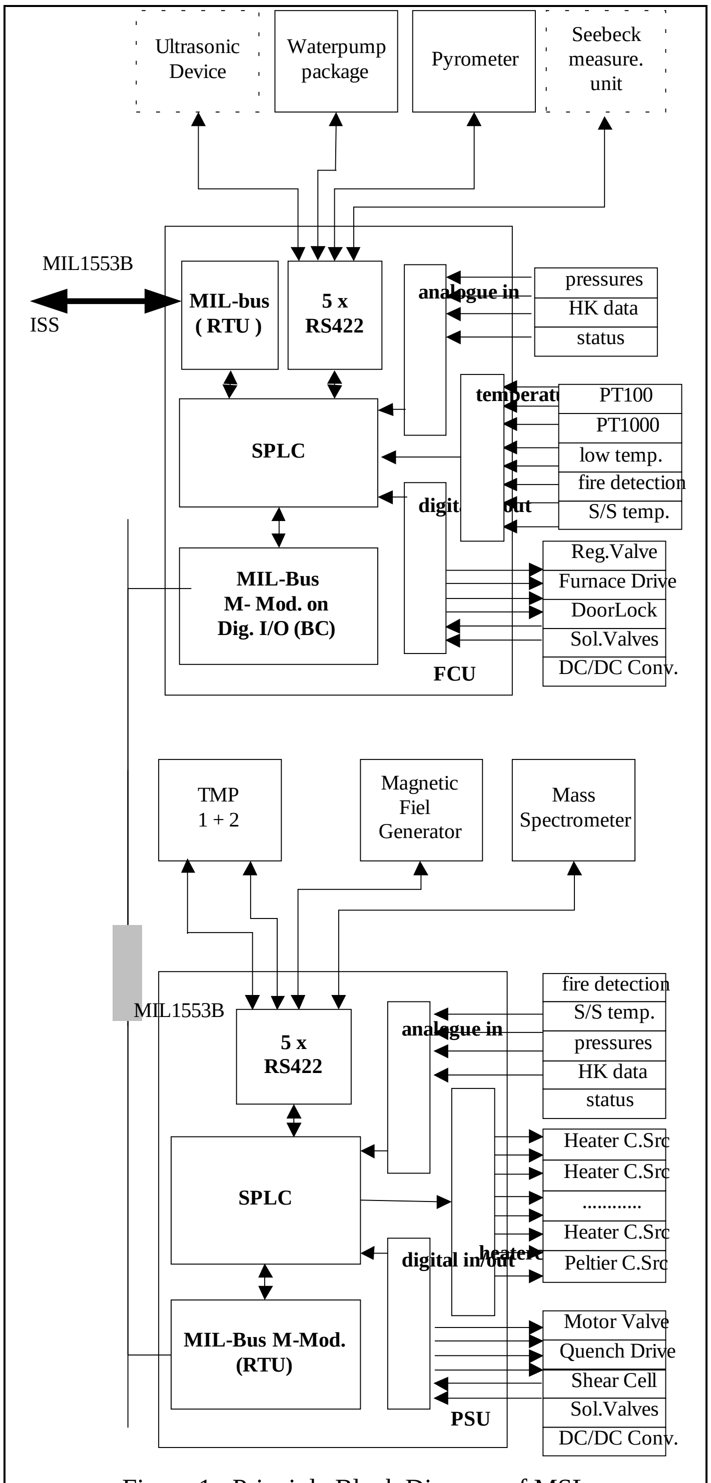

Figure 1 : Principle Block Diagram of MSL

Additionally, single discrete digital I/O bit allow to control devices and subsystems like valves and actuators and to acquire digital status bit. Analogue signals are input by means of analogue to digital converters.

The MSL S/W shall acquire the analogue and digital data - temperatures, pressures, status signals – and shall use these for telemetry and to control the sample processing and to detect critical situations.

The commands that control the MSL S/W functions are uploaded to the MSL S/W by means of the MIL-STD 1553 B communication between Material Science Research Rack (MSRR) Master Controller of the ISS and the FCU. On the same interface telemetry data are downlinked.

The communication between the FCU and the PSU is performed by means of a second internal MIL-STD 1553 B bus for which the FCU is the Bus Controller.

The MSL S/W shall allow that its functionality can be distributed and can be allocated on one of the two SPLC units, either the FCU or the PSU. A TCP/IP communication protocol is established between the FCU and the PSU on the MIL bus.

The block diagram in Figure 1 gives an overview of the configuration of MSL.

Some 600 data items either from digital or analogue or RS422 data sources have to be acquired and processed. The acquisition rate of the signal ranges from 100/s to 1/s. The processing cycle is based on 500 ms and 1000 ms intervals. The temperature control is performed with a period of 500 ms.

According to the Architectural Design, the MSL software consists of about 40 process types and 50 process instances distributed over two SPLC processors. Some 250 commands control the system. Circa 150 external commands exist of which about 120 are constrained by a time-out condition. In total, about 45 different states are defined and some 350 state transitions are executed.

Approximately 450 stubs are automatically generated into which the specific MSL functionality can be plugged-in. About 650 functions are automatically generated for the database, telemetry, data and, command processing software (DTDC software) dealing with data acquisition, database handling, housekeeping and telemetry handling.

3. RATIONALE FOR THE CHOSEN APPROACH

For a rather long time during the system definition and specification phase of the project the functional requirements and subsystem specifications were subjected to farreaching changes. It is not possible to start detailed software design and coding on the basis of only temporarily valid specifications and fluctuating requirements.

On the other hand for MSL it was necessary to gain information on processing budgets at a very early stage of the project. Only vague performance figures of the SPLC platform were available that had to be proofed in order to be able to initiate in time possible corrective actions.

As a way of escape from this situation a new approach for the software development process was chosen. The aim was to achieve executable specifications that allow an incremental development that is tolerant against changes. In the software project for MSL, emphasis was put on the development of software that automatically generates application software code from the specification rather than manually transcribing the specifications to software code.

With this approach the change of requirements and configurations has less impact on the software development process.

4. ORGANISATION OF THE PROJECT'S DATABASE AND ISG INPUTS

The fundamental specifications to be provided for a particular system comprise the following :

- input data and signals and output data (telemetry) description
- specification of the processing algorithms of the data and signals (conversions, calibrations, corrections)
- the architectural design of the software reflecting the functional requirements
- definition of the set of operator commands (telecommands) reflecting the major part of the operational requirements

Figure 2 gives an overview of the tables and datasheet structure in MSL project.

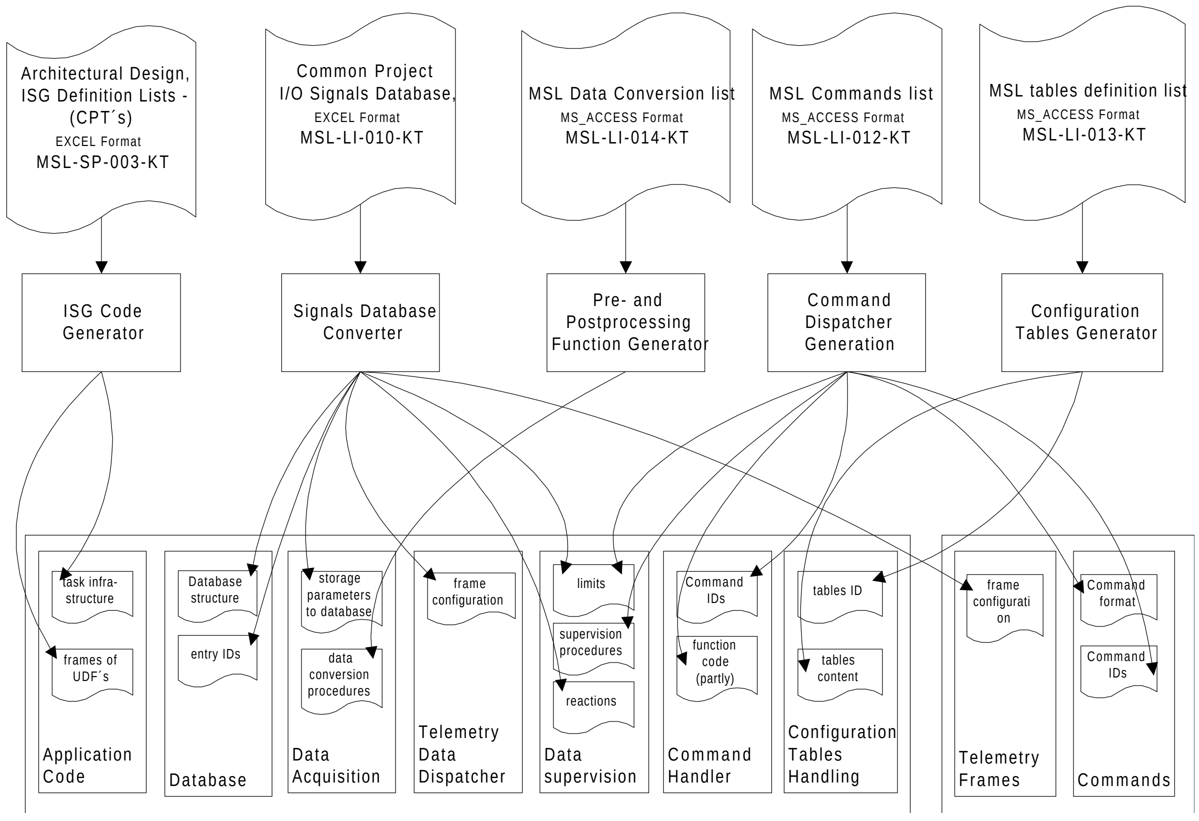
Architectural Design, ISG Definition Lists - (CPT´s) EXCEL Format MSL-SP-003-KT
Common Project I/O Signals Database, EXCEL Format MSL-LI-010-KT
MSL Data Conversion list MS_ACCESS Format MSL-LI-014-KT
MSL Commands list MS_ACCESS Format MSL-LI-012-KT
MSL tables definition list MS_ACCESS Format MSL-LI-013-KT
ISG Code Generator
Signals Database Converter
Pre- and Postprocessing Function Generator
Command Dispatcher Generation
Configuration Tables Generator
task infra-structure
frames of UDF´s
Application Code
Database structure
entry IDs
Database
storage parameters to database
data conversion procedures
Data Acquisition
frame configuration
Telemetry Data Dispatcher
limits
supervision procedures
reactions
Data supervision
Command IDs
function code (partly)
Command Handler
tables ID
tables content
Configuration Tables Handling
frame configuration
Telemetry Frames
Command format
Command IDs
Commands

Figure 2 : Overview of the Datasheet Structure

The content of all of the tables is generated by system engineering processes on a rather high level of definition. As the project progresses more and more detailed information can be added (e.g. the I/O signals database contains now data describing the harness, color of cables). Changes on System level – e.g. adding or deleting an input signal – are reflected by according changes of the database and are immediately available to all engineering departments.

By means of particular converters and code generators parts of the applications software are generated from these datasheets.

5. GENERATION AND INTEGRATION OF THE MSL APPLICATION SOFTWARE

The system database, holding and managing all acquired and internally generated data, is described in the I/O signals database with the name of the signal, its generating source, the acquisition rate, the data format (e.g. float, integer,bit), and its valid range. From these raw data the application system database including the access functions can now easily be generated. Also parts of the data acquiring functions, like A/D converter interrupt service routines can be generated based on the tables.

The functions for data preprocessing, calibration, correction, and conversion are specified and assigned to the corresponding data items in the I/O signals database. The code generator produces automatically the code for sequential processing of a certain data item beginning with its acquisition from the data source and ending with its correct storage in the database. Part of the processing is also the limitcheck of each data item.

Changes, to the configuration - adding, deleting, and modifying signals or data items – are performed in the datasheet. After a re-run of the converters a new consistent software is available.

The datasheets from which the infrastructure for the application code is generated are based on the architectural design. The software is decomposed into tasks, representing the active components. The behavior of each component is described as a finite state machine (FSM). To each of the state transistions user defined functions (UDF) can be assigned. Additionally the performance characteristics and the processor allocation for each particular function are defined. These data are the major inputs in the CPT (Command Processing Tables) datasheets as basis for the ISG tool and allow for the automatic generation of the software infrastructure. (The ISG tool is presented in more details in a separate paper)

6. CONCLUSIONS

The chosen development approach had the following advantages :

- the system turnover time for changes and modifications is significantly decresaed and the risk of inconsistencies is diminished.
- the incremental development approach allows for the planning and scheduling of development activities according to their complexity and their potential technical risk
- the ISG tool allows for the verification of the software architectural design and the control flow based on the specification.
- it is possible to gain credible figures for memory and performance budgets at a very early stage of the project