Usage of Ada in the Gripen Flight Control System

Bo Frisberg
Saab AB
SE-581 88 Linköping
Sweden
Phone: +46 13 18 32 18
Bo.Frisberg@saab.se

1. ABSTRACT

The positive experiences from the usage of Ada in a safety critical flight control system are described in this report. It shows that preemptive scheduling implemented with tasking can be combined with high requirements on reliability and deterministic behavior.

1.1 Keywords

Flight control system, tasking, data consistency, exception handling.

2. INTRODUCTION

The flight control system in the Swedish military aircraft JAS 39 Gripen has been upgraded and is now programmed in Ada 83, which includes tasking and exception handling in a limited way (further described in [1]). Traditionally, these flight control systems have been programmed in assembler with simple cyclic executives.

The test flights with the upgraded flight control system started in the beginning of 1996 (on schedule) and deliveries to the Swedish Air Force in production aircraft started later in the same year.

The system described in this report was the first in the Gripen avionics programmed in Ada. However, today Ada has been selected for other upgrade programs in the avionics.

3. SYSTEM DESCRIPTION

The JAS 39 Gripen is a multi-role (fighter, attack and reconnaissance) aircraft and is produced in both single- and two-seat versions. The differences between these versions are handled within the same software.

The aircraft has an electrical (fly-by-wire) flight control system, which performs input signal conditioning and voting, control law computations, functional monitoring and redundancy management, data recording, etc. Autopilot functions are also provided.

Typical input sources are stick and pedal position sensors, angular rate gyros, accelerometers, buttons for mode selections, etc. Outputs are sent to control surface actuators, indicators and warning lamps, etc. Communication with other subsystems in the aircraft is performed via a 1553 data bus.

The flight control system has three redundant channels. Identical software resides in the channels, and each channel contains the following two processors:

- Primary processor (MC68040).
- Input/Output and backup processor (TMS320C30).

Simplified backup control law computations are included in the Input/Output processor in order to be able to fly home and land safely in case of primary side failures. The I/O & backup processor is programmed in C, and the primary processor is programmed in Ada. Different languages were selected in order to obtain diversity. The software written in C is of significantly smaller size and the number of changes are anticipated to be insignificant, i.e. future development and update of the system is expected on the primary processor side.

The three redundant channels exchange data via a Cross Channel Data Link (CCDL), and the two processors within one channel communicate via a Dual-Port RAM.

4. USAGE OF ADA

The primary processor is programmed in Ada 83 (the compiler is EDS XDAda version 1.3 with the standard run-time system and VAX as host computer). Ada is used both for the flight part of the program and for the Built-In-Test (BIT). Ada has been found reliable and sufficient even for routines with close hardware interaction, and it has been possible to minimize the usage of assembler code (assembler has not been necessary for performance reasons). The strong typing and controlled interfaces are examples of Ada features which support development of reliable software. Abstract Data Types (ADTs) have been found useful for implementation of filters, integrators, faders, etc., which are typical building blocks in this application.

4.1 Tasking and Scheduling

The flight control software in the primary processor is implemented in Ada with five periodic tasks. The harmonic frequencies are 120, 60, 30, 15 and 7.5 Hz, which are common for the total avionic system including the 1553 bus communication. The functions are allocated to the tasks depending on the required data update frequency.

The periodic tasks are assigned fixed priorities according to the rate monotonic scheduling algorithm, i.e. higher priorities to tasks with shorter periods.

The scheduling is implemented by a separate task with the highest priority, which is activated at 120 Hz (the 8.33 ms minor frame) by an interrupt from a real time clock device. This scheduler task activates the periodic tasks in required intervals and also detects if any periodic task has missed its deadline (overrun). As this is a hard real-time application, a missed deadline means that the backup control laws (basic control with only a single periodic thread) in the other processor are made active. It should be noted that it is possible to detect a missed deadline before the time for the next periodic activation of a task, i.e. the allowed execution time can be shorter than the period time (but both the allowed execution and period times have to be multiples of the minor frame).

A background task only measures and stores the computational load of the periodic tasks (the real time clock timer is read when the periodic tasks have finished its execution and the background task is resumed). Furthermore there are two tasks with lower priorities which are only executed during BIT.

Although tasking is allowed in this application, the following restrictions are applicable:

- All tasks are declared at library level and are created at start of the program.
- No task is allowed to terminate, which means that each task contains a non-terminating outer loop. Abort is not allowed.
- All tasks have unique and fixed priorities (no dynamic priorities).
- The Calendar package and the Delay statement are not used (not needed in this application).

4.2 Data Consistency

In a high integrity system where preemptive scheduling is allowed, it is essential to ensure that data exchanged between the tasks are consistent. The harmonic period times make it possible to achieve a fixed scheme of data exchanges via buffers (described more in detail in [1]). The data buffers require some extra RAM space, but the execution time overhead is minimal. The buffering ensures data consistency. Furthermore, it means that the timing of data exchanged between the periodic tasks is independent of the current CPU utilization, i.e. preemptive scheduling can be combined with a deterministic behavior.

4.3 Other Design Characteristics

As there is no direct communication or synchronization between the periodic tasks, the risk for priority inversion, deadlock and other undesired task blocking is eliminated.

All tasking and data buffering mechanisms are handled at the executive level, transparent to the application functions. The sequential parts of the program can be analyzed and tested separately. This is also an advantage from the maintenance point of view. When application functions are modified or new functionality is added, the tasking and communication framework will normally not be affected at all.

4.4 Exception Handling

Exception handling is another Ada feature which has been questioned from safety point of view. In the Gripen flight control system exceptions are handled, but there are no advanced recovery mechanisms designed with exceptions. Whenever the occurrence of an exception can be foreseen, the exception should normally be avoided. But still there may be situations where exceptions are difficult to foresee, e.g. exceptions triggered by hardware failures.

If an exception occurs, the type of error and program location are always recorded for diagnostic purpose. Exception event flags as well as other failure flags are stored in a non volatile memory for inspection after landing. The general action when an exception occurs is then to make the simplified backup control laws in the I/O & backup processor active (similar to the periodic task overrun situation).

A special situation where exception handling has been found appropriate is to make it possible to catch failures in isolated procedures which are not critical to the application. For example, functionality may be added in such procedures only for recording purpose during flight tests. A correct normal function can still be ensured after an exception.

Another special use of the exception mechanism in this application is for exit of the BIT execution on ground when the speed of the aircraft exceeds a certain value (take-off). This is not the ordinary way to exit BIT, and can basically be seen as a form of asynchronous transfer of control.

So far, no exception has been raised in the air.

5. SUMMARY

The project experiences prove that Ada with tasking and exception handling can be used in a safe and efficient way in embedded control systems with strong requirements on reliability and deterministic behavior. A robust software architecture and design is necessary, including restrictions and limitations in the usage of tasking and exception handling.

The flight control system described in this experience report is written in Ada 83, and no transition to Ada 95 is planned today.

6. ACKNOWLEDGEMENT

Thanks to my colleague Kjell Larsson for his comments.

7. REFERENCE

[1] Frisberg, B. Ada in the JAS 39 Gripen Flight Control, System, In Lars Asplund (Ed.) Ada-Europe’98 Conference Proceedings, LNCS 1411, Springer Verlag, 1998.