UDK 621.452.32.001.57

A. SUMTSOV, K. SZURMAN

UNIS, a.s., Division Aerospace and Advanced Control, Brno, Czech Republic

MODEL BASED DESIGN APPROACH APPLIED ON THE DEVELOPMENT OF THE ENGINE MONITORING MODULE

The development of the aerospace industry nowadays is focused on continuous monitoring of equipment. The outputs of the monitoring algorithms can be used for planning the maintenance with respect to operative modes and conditions. In such systems, the device that monitors particularly the engine is usually referred to as the Engine Monitoring Module (EMM). The Model Based Design (MBD) approach and its application on the development process of EMM are described. Using this approach a dramatic development time saving is acquired. Monitoring algorithms are implemented in Matlab/Simulink environment and the final software is a combination of hand-written and automatically generated C code.

Keywords: *FADEC, engine monitoring, MBD, Matlab, Simulink, code generation.*

Introduction

One of the main differences between conventional preventive maintenance and the application of monitoring systems is the approach to the planning. The former relies on time-based scheduling of maintenance actions, whereas the latter enables condition-based practice. A complex view into the health of each aircraft of the fleet is granted and upcoming component problems are indicated in advance. This means that operators are provided with all the necessary information to make beneficial maintenance-related planning decisions.

Small problems that can be easily fixed like imbalances, faulty installations or starting gear and bearing degradation are indicated. The operator then focuses the maintenance on these easily fixable things and prevents the problem to develop and cause collateral damage to the aircraft.

The monitoring system allows both quick intuitive on-board real-time troubleshooting and on-ground complex in-depth data processing. Trend analysis tools help to detect the root of emerging problems and minimize the troubleshooting time, greatly reducing the no-fault-found component removals.

The Engine Monitoring Module (EMM) development process described in this paper took place during a collaborative aerospace project under Efficient Systems and Propulsion for Small Aircraft (ESPOSA). UNIS, a.s. company contribution part was to develop a Full Authority Digital Electronic Control unit (FADEC) for an aircraft engine AI450-BE2+.

The FADEC itself consists of four modules – two redundant Electronic Control Units (ECUs), the EMM unit and a Vibration Monitoring Module (VMM). ECU units are in charge of engine control where the one is active and the second performs the same functionality but its actuators do not control, the VMM acquires data from vibration sensors around the engine and it also detects overspeed of a free turbine and gas generator. Dangerous vibration and overspeed situations are handled by the ECU algorithms and the EMM carries out the monitoring and diagnostics tasks.

In this paper, the system requirements for the EMM are introduced and the process of development of the EMM with the use of the Model Based Design (MBD) is described. Moreover, implemented software architecture and proposed interface between hand-written and generated MBD tasks are presented. Finally, the process of generating the C source code from Simulink models is presented and benefits of the MBD approach are summarized.

1. Requirements

The EMM is designed to complete the following tasks:

– the module continuously monitors the values of defined engine parameters and signals. It processes these signals and values of parameters according to the algorithms described below and send results to the crew. This monitoring and processing takes place both while the aircraft is on the ground and during the flight;

– the module continuously checks if the monitored parameters are in the defined range. The range for the measured parameters can be defined both statically and dynamically according to the change in flight mode;

– the module detects faults and damage that can affect decisions of the crew concerning the control of the aircraft during the flight and the actions of maintenance crew during the on-ground after-flight service;

– forming messages of the "event" type concerning faults and the damage of the engine and its systems and sending them to the cockpit;

– detection of flight modes for later use in deter-

mining the wear-off of the engine;

– calculating sum flight durations for limited modes and the total sum flight duration for the engine in hours and cycles;

– determining the engine's slow-down time after it is switched off;

– processing the data for on-ground maintenance and trend analysis diagnostics of the flow system of the engine.

The input signals and parameters of the EMM are being previously validated using the reliability criteria defined for the built-in validation check system. If the input signal in a given sample is detected to be faulty, the last validated sample is used for the algorithm processing.

2. Hardware

The EMM module is based on a hardware platform which is used also for the ECU units considering modular approach applied on the FADEC development.

The platform consists of three digital cards: peripheral, control and communication. Each digital card contains except of specific hardware equipment and peripherals given by defined functionality also TI tms320F28335 microcontroller (MCU) which is responsible for execution of implemented software. Digital cards are interconnected through a Controller Area Network (CAN) bus where a data and control parameters are transferred. The control card controls start and operation states of other digital cards.

3. Software

For software development a combined approach was chosen to implement the demanded functionality of the EMM by hand-written source code and source code generated and developed through the MBD approach.

Software was implemented in C programming language according to ANSI C90 standard conforming to software standard MISRA-C 2012 for safety-critical systems.

3.1. Software architecture

Software equipment for all digital cards of the EMM is based on the same architecture. This architecture and its basic layers are shown in Figure 1. The core of the software architecture consists of card framework which implements simple operating system with a scheduler. Low level system initialization and functions for operation with the MCU's peripherals are implemented in the peripheral drivers' layer.

The scheduler periodically executes system and application tasks. Application tasks can be implemented in hand-written software modules or in software modules generated from models developed in Matlab/Simulink during the MBD.

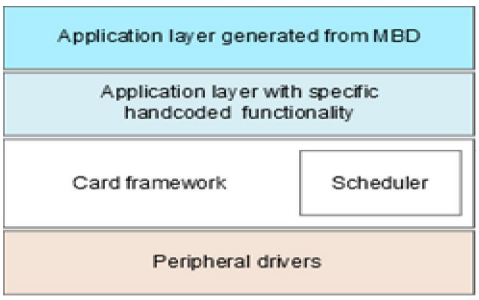

Figure 1. Basic software architecture layers

Before the card framework is started, an initialization sequence and low level configuration of the digital card must be performed. Then, execution of application tasks specific for the given card is started.

3.2. Scheduler

After the initialization sequence is finished, the scheduler starts the periodic execution of scheduled tasks which were added during the initialization sequence. The execution of all tasks is performed in the scheduler's loop.

During the scheduler execution, the timer TIMER2 is counting up to TASK_TIMER_PERIOD_VALUE. When the timer counter is equal to this value, a global counter is incremented in an interrupt routine. The scheduler checks the deadlines of all tasks. A task is executed when its scheduled deadline is met. After a task execution is finished, a new task deadline is calculated.

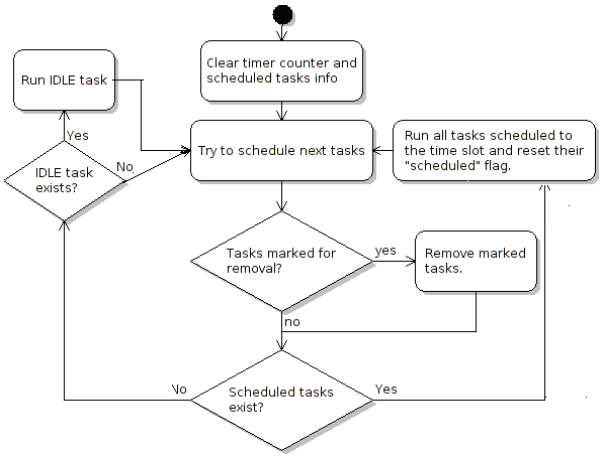

Figure 2. Main scheduler loop

Figure 2 shows the state diagram implemented by the scheduler. The scheduler supports dynamic adding and removing of tasks. Nevertheless, the scheduler executes a static sequence of tasks added during the initialization. An idle task or low power MCU mode is not used.

3.3. Generated source code integration

Generated source code and software modules from MBD are integrated into the software architecture on the two levels. Function which are supposed to be executed periodically are generated in a form of task which can be directly added into the scheduler queue. Then, these application tasks can use functions from the software architecture e.g. for reading of inputs.

Therefore, we defined unified interface which define a function name mapping between software modules implemented by a hand for the card specific functionality and generated software modules. It is formed by `card_gets.h` and `card_sets.h` header files. An illustrative diagram is shown in Figure 3.

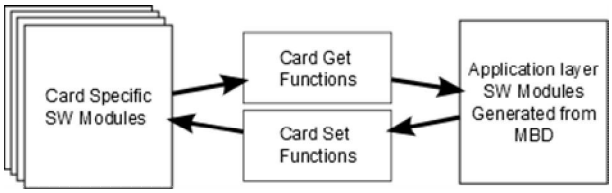

Figure 3. Interface between the card specific functions and generated functions from MBD

4. Model based design and generated tasks

There are three main tasks in the EMM system. Tasks are called according to the card they are run on. These are the peripheral card task, the communication card task and the control card task.

The role of the peripheral and communication cards can be called supportive. The control task is the one that runs the top level – most important from functionality point of view – algorithms.

In this chapter I will briefly go through the peripheral and communication cards tasks, and then have a closer look at algorithms that are run on the control card.

4.1. Peripheral card task

There are raw analogue signals from the sensors voltage and raw discrete input signals entering the algorithm. This task reads raw electrical values, validates them and provides the shavings detection algorithm. The inputs are transferred into the control card through the CAN bus. The outputs of the peripheral card are generated according to demands of the main algorithms in the control card which are received via the CAN bus.

As the name suggests, the card purpose is to manage the communication, signal harvesting and control of the peripheries. This card takes the input signals and according to the sensor specification turns the electrical signal into a physical interpretation. Along with the mentioned conversion process the validation process takes place. The code running in this card also detects and generates the corresponding error flags of electrical, range, gradient and channel errors of the inputs.

4.2. Communication card task

This task runs on 100Hz and services communication with redundant ECU channels through an ARINC interfaces. The task periodically transmits the EMM data and parameters to both of ECU channels.

First, the card task communicates with the VMM through the CAN bus and CANAerospace application protocol and reads acquired data from vibration sensors around the engine and speed of a free turbine and gas generator. These data are sent to the control card to process and evaluate by the main EMM algorithms.

Then, evaluated engine monitoring parameters and other data are received from the control card through the CAN bus. These data are processed and converted to a form compatible with ARINC communication protocol and sent to the ECU units.

Simply put, the communication card takes input signals from other modules, decodes, regroups and prepares again them for other two tasks to be used as inputs, then takes these task's outputs, regroups again them, code them back and sends them to other modules. The most important challenge of this task is to make the coding/decoding receiving/transmitting as efficient as possible.

4.3. Control card task

This task reads data from the VMM module received by the communication card and data from the peripheral card received through the CAN bus. Then, the task executes the EMM functionalities and evaluate acquired parameters of the engine.

This task runs several subsystems – modular algorithms – that will be described closely further into this chapter. These are:

– Emm_counter – it manages the calling of other sub-algorithms;

– AK01 – Engine modes detection;

– AK02 – Gas dynamics parameters monitoring;

– AK03 – Oil system monitoring;

– AK04 – Fuel system monitoring;

– AK05 – Number of starts and duration of start sequence monitoring;

– AK06 – On-ground engine slow-down time monitoring;

– AK07 – Wear-off calculation algorithm;

– AK08 – Data gathering algorithm for the purpose of on-ground maintenance of the flow system.

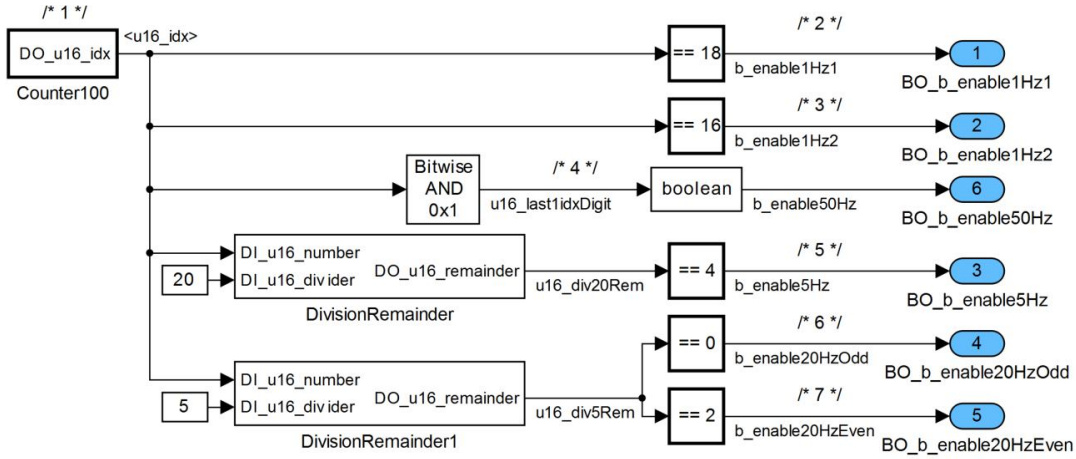

Figure 4. Trigger generating algorithm implemented in Simulink model

The task itself is called periodically at 100 Hz, Different algorithms listed above are supposed to run on frequencies of 1, 5, 20 and 50 Hz. There is subsystem enabling counter algorithm used to call different algorithms at desired frequencies (Figure 4). In fact it is a simple counter which is continuously incremented and is reset when 100 is reached. To generate a certain frequency of triggering the value of the counter is either compared to a number (1 Hz), or is divided with the remainder that is compared to a number (5, 20 Hz), or the last bit of the value of the counter is checked on oddity (50 Hz).

Parameters of this algorithm are chosen to minimize the load on a processor by calling different sub-algorithms non-simultaneously.

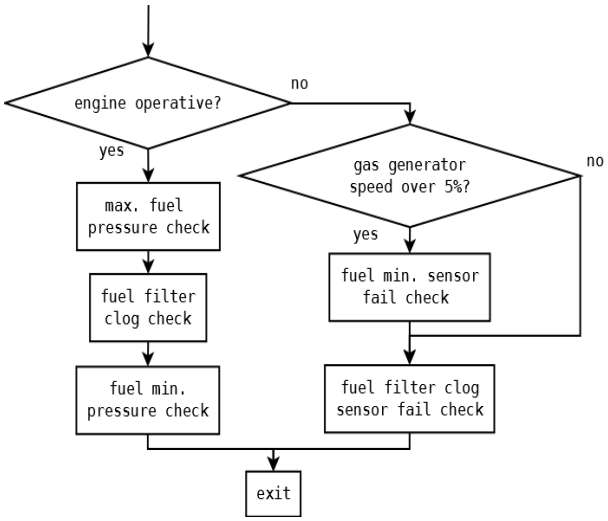

Figure 5. AK04 flowchart diagram

Each sub-algorithm is implemented in Matlab Simulink as a Stateflow diagram. Figure 5 shows the flowchart diagram representing the fuel monitoring algorithm. Every time the subtask is triggered, it is sequentially executed and corresponding signals are sent out.

5. Code generation sequence

In this chapter code generation method used during the project is briefly described. Some suggestions on model adjustments are given and things to avoid when creating Simulink model for the purpose of automatic C code generation are listed.

Each task represents a separate functionality and it is created with a standardized interface. The top level algorithms are generated and are embedded into a hand-written low level microcontroller interface containing drivers and a scheduler. This approach is inevitable in case the target hardware is not supported by the Embedded Coder or if non-standard drivers or scheduler are required.

The approach gives us full control over the code and an unlimited ability to update or improve the code.

5.1. Implementation

During the implementation phase, the module's algorithms are converted from a Simulink model to the target platform source code in C language. This can be done with the use of Matlab Coder and Simulink Coder toolboxes. There is also a possibility of using the Embedded Coder Toolbox to generate the entire code including the scheduler and drivers for a specific microcontroller. This toolbox includes a set of blocks that provide the hardware interface in the model.

The appearance and the behavior of the final generated code is highly influenced by many factors, such as the model architecture, settings and constructions, signal data types or the storage class definitions. Because the target code is highly model-dependent, it is necessary to use certain constructions and keep strict model-designing rules and standards. If we didn't keep them, the generated code would lack legibility, comprehensibility, traceability to the source model and a standard interface for connection with other source code parts.

With Matlab Simulink, the generated C source code can be easily traced back to its origin if it was modelled in appropriate way. From the other side, automatic transformation of the Simulink model into understandable C source code can be quickly checked. Moreover, the Matlab Simulink can be integrated with the Polyspace tool for static code analysis to check compliance with the MISRA-C 2012 rules and for absence of a dead code and runtime errors.

Conclusion

The monitoring systems' introduction was given and requirements were shown for a particular monitoring module. The Model Based Design approach use was shown. The C code automatic generation procedure was suggested and some hints were given.

Overall the automatic generation of the C source code and MBD approach is the current trend in aviation development with its origin in flexible altering the software and its time-saving fundamentals. Also this approach is anchored in RTCA/DO-178C aviation standard. Since 2013 it is used by certification authorities such as FAA, EASA and Transport Canada.

Acknowledgments

This work was supported by FP7 EU project ESPOSA – "Efficient Systems and Propulsion for Small Aircraft", grant agreement no. ACP1-GA-2011-284859.

References

1. UTC Aerospace Systems, Health and Usage Management Systems (HUMS) [Electronic resource]. – Access mode: http://utcaerospacesystems.com/cap/products/Pages/health-usage-management-systems.aspx. – 12.04.2015.

2. US Joint Helicopter Safety Implementation Team, Health and Usage Monitoring Systems Toolkit, 2013 [Electronic resource]. – Access mode: http://www.ihst.org/portals/54/Toolkit_HUMS.pdf. – 12.04.2015.

3. Sumtsov, A. SW development and HIL testing for engine monitoring module [Electronic resource] / A. Sumtsov. – Brno : Brno University of Technology, 2015. – 65 p.– Access mode: https://dspace.vutbr.cz/bitstream/handle/11012/39389/2015_DP_Sumtsov_Artem_108123.pdf?sequence=1&isAllowed=y. – 12.04.2015.

Поступила в редакцию 2.06.2015, рассмотрена на редколлегии 22.06.2015

Рецензент: д-р техн. наук, проф., зав. каф. конструкции авиационных двигателей С. В. Епифанов, Национальный аэрокосмический университет им. Н. Е. Жуковского «Харьковский авиационный институт».

ПРИМЕНЕНИЕ МОДЕЛЬНО-ОРИЕНТИРОВАННОГО ПРОЕКТИРОВАНИЯ ДЛЯ РАЗРАБОТКИ БЛОКА УПРАВЛЕНИЯ САМОЛЁТНОГО ДВИГАТЕЛЯ

А. Сумцов, К. Шурман

Развитие аэрокосмической промышленности в настоящее время сосредоточено на непрерывном мониторинге оборудования. Результаты алгоритмов мониторинга можно использовать для планирования технического обслуживания в соответствии с оперативными режимами и условиями. Устройство, которое контролирует параметры двигателя, обычно называется модулем контроля двигателя. В статье описан подход модельно-ориентированного проектирования (МОП) на примере разработки алгоритмов контроля двигателя. Использование этого подхода позволяет сократить срок разработки. Алгоритмы мониторинга разработаны в среде Matlab/Simulink. Финальное программное обеспечение является сочетанием автоматически генерируемого и написанного вручную кода С.

Ключевые слова: FADEC, контроль двигателя, МОП, Matlab, Simulink, генерирование кода.

ЗАСТОСУВАННЯ МОДЕЛЬНО-ОРІЄНТОВАНОГО ПРОЕКТУВАННЯ ДЛЯ РОЗРОБКИ БЛОКА КЕРУВАННЯ ДВИГУНА ЛІТАКА

А. Сумцов, К. Шурман

Розвиток аерокосмічної промисловості у поточний час зосереджений на безперервному моніторингу обладнання. Результати алгоритмів моніторингу можна використовувати для планування технічного обслуговування відповідно до оперативних режимів і умов. Пристрій, який контролює параметри двигуна, зазвичай має назву модуль контролю двигуна. У статті описано підхід модельно-орієнтованого проектування (МОП) на прикладі розробки алгоритмів контролю двигуна. Використання цього підходу дозволяє скоротити термін розробки. Алгоритми моніторингу розроблено у середовищі Matlab/Simulink. Фінальне програмне забезпечення є сполученням коду С, який генерується автоматично і написаний уручну.

Ключові слова: FADEC, контроль двигуна, МОП, Matlab, Simulink, генерування коду.

Sumtsov Artem – MBD developer at UNIS, a.s., Division Aerospace and Advanced Control, Brno, Czech Republic, e-mail: asumtsov@unis.cz.

Szurman Karel – Embedded software developer at UNIS, a.s., Division Aerospace and Advanced Control Brno, Czech Republic, e-mail: kszurman@unis.cz.