Programming possibilities using MATLAB simulink embedded coder on the example of data analysis from ahrs module.

S Elert
Military Institute of Armament Technology, Prymasa St. Wyszyńskiego 7 Street, 05-220 Zielonka, Poland
elerts@witu.mil.pl

Abstract. This paper shows capabilities to program Texas Instruments TMS320F2839xD DSP (Digital Signal Processor) using MATLAB Simulink Embedded Coder on the example of receiving and processing data from VN-100 AHRS module. The first part of article described specification of C2000 Delfino MCU F28379D LaunchPad development kit and AHRS module. Next part presents physical connection VN-100 module with microcontroller and configuration method to setup MATLAB Simulink and Embedded Coder for selected microcontroller. Presented program in MATLAB Simulink shows communication between AHRS module, VN-100 and microcontroller treated as receiver. Also It shows how to generate code and described the basic functions that were used to run application. The designed system is explained as a photos from MATLAB and real tests. The last part of paper demonstrates graphic preview of microcontroller load possible during operation and advantages of programming using MATLAB Simulink Embedded Coder.

1. Introduction
Modern control systems are used in complex systems, which makes their designing process complicated and lengthy. One example of the application of those systems is precision-guided munition, which in addition to control elements supports many subsystems responsible for the location and nature of operation of the entire actuating system. In order to simplify and reduce working time during the design phase, programming in Embedded Coder MATLAB Simulink environment is possible. Embedded Coder generates compact code in C and C++ for embedded processors such as Texas Instruments. Owing to fast and automatic code generation and its optimization, it is possible to quickly compare operation of the program in many configurations. Owing to extensive possibilities of programming DSP signal processors in the MATLAB Simulink environment, the TMS320F2839D processor was programmed, with data received from the VN-100 sensor that provides information about the controlled system’s current position. This article aims to present the process in the following order:
- hardware configuration,
- MATLAB Simulink,
- model simulation with code generation,
- real-time monitoring of program operation
Following code generation, processor performance analysis was carried out, which allows for its optimization if the workload is known. This option allows for effective modernization of control systems of objects that require high CPU timing and multithreading.

2. Technical configuration
A low-cost device was used to perform the analysis. The Delfino TMS320F2837xD is a powerful 32-bit floating-point microcontroller unit (MCU) designed for advanced closed-loop control applications. The dual real-time control subsystems are based on TI’s 32-bit C28x floating-point CPUs, which provide 200 MHz of signal processing performance in each core. [2] The microcontroller is supported in the MATLAB Simulink environment through a dedicated toolbox, TI C2000 Support for Embedded Coder which generates C/C++ code that can be compiled on C2000 microcontrollers using Code Composer Studio IDE. [1] It is possible to easily and quickly configure MATLAB Simulink code generation and optimization. In the described experiment it was used USB connected isolated XDS100v2 JTAG debug probe for real-time debug and flash programming and one Serial Communication Interface module for communication with the AHRS module.

![Figure 1. Connection between the microcontroller(left) and the VN-100(right).](image)

To measure the position was used Inertial measuring device VN-100 which is a miniature Inertial Measurement Unit and Attitude Heading Reference System. The VN-100 is based on the latest MEMS sensor technology and combines 3-axis accelerometers, 3-axis gyroscopes, 3-axis magnetometers, a barometric pressure sensor and a 32 bit processor. [3] In this measurement was used the Rugged version which contains the VN-100 enclosed in a clamshell precision anodized aluminium enclosure. Interfacin with the module is made trough a locking 10-pin connector and communication interface Serial RS-232 and TTL. This example uses TTL serial communication.

3. Code generation and analysis

3.1. Configuration
The first step in Matlab SIMULINK is system configuration (Configuration Parameters), which includes the microcontroller peripherals and definition of its type. The order of bytes is important: in TI C2000 processors, Little Endian is set by default, i.e. the format in which the LSB bit (the least significant) is placed first; besides, TI C2000 processors have the smallest addressable cell of 16 bits.
Thus, 8-bit values take up 16 bits. First, the serial port (SCI-C) was configured to identify the correct data frame during transmission. *Hardware Implementation>* *Target hardware resources*

- SCI_C – baud rate 115200 bit/sec.
  
  The VN-100 is configured by default with the following parameters:
  - Transmission speed 115200 b/s;
  - Operating frequency 40 Hz;
  - Measurement data type: ASCII.

An example data frame is shown below:

$VNYMR,-161.791,-007.279,-017.792,-02.1412,+00.5506,+00.8598,-00.831,+02.666,-09.374,-00.058151,-00.469756,-00.283905*6B<CR><LF>

![Figure 2. Hardware configuration.](image)

![Figure 3. Data transmission on the TX board (preview by oscilloscope).](image)
Data is transmitted every 25 ms and each frame consists of 122 8-bit characters. The sampling rate in Simulink depends on data transmission frequency. The SCI RCV block defines the interface module (SCI C), the type of data received, the frame length (figure 5). Sampling time should be defined as 25 ms or inherent (-1) if the $Ts$ of the whole system is AHRS frequency compatible. Received data in the form of uint8 is treated by the TI processor as 16-bit values, so one more 8-bit value is added to each 8-bit character in the form of 0 for alignment. After conversion to 16-bit type, the zeros are not taken into account. To receive data correctly, the hardware interrupt block is used (figure 4), the output of which is a function call, i.e. data receive block. The interruption is represented by four parameters [4]:

- CPU interrupt numbers;
- Peripheral Interrupts Expansion interrupts numbers;
- Task priorities;
- Preemption flags.

In the presented model, there is one hardware interruption supporting transmission from the VN-100 module.

**Figure 4.** Using of hardware interrupt.

**Figure 5.** Serial Communication Interface configuration.
The received frame in the vector form, containing 122 characters, is divided into individual smaller vectors, each value consists of eight 8-bit characters. The Selector block was used to separate the individual measurement values (figure 7). Following the above separate operations, data in the form of [1x8] vectors is converted from ASCII to decimal values by means of a function for correct reading:

```matlab
function [yaw, pitch, roll] = fcn(u1, u2, u3)
    U1=u1.;
    U2=u2.;
    U3=u3.;
    
    ch1=char(U1);
    ch2=char(U2);
    ch3=char(U3);
    
    str1=string(ch1);
    str2=string(ch2);
    str3=string(ch3);
    
    yaw=real(str2double(str1));
    pitch=real(str2double(str2));
    roll=real(str2double(str3));
end
```

**Figure 6.** Conversion data types from ASCII to double.

**Figure 7.** Separating the individual measurement values.
Figures 8 and 9 show examples of measurements from the IMU sensor reading the angles of pitch, roll and yaw and acceleration respectively. Based on the above graphs it can be noticed that data frames were received correctly; an additional confirmation of error-free transmission is the data reception status in the SCI Receive block, which informs about possible errors during incorrect data transmission. The “0” status means no errors during data reception.

3.2. Performance
The next step was to check the generated code in real time. Shown below is the procedure of identification of tasks and execution time of application functions in real time. Based on that information, computer performance can be optimized. The analysis was carried out for two program variants:

- Variant I - Receiving data without using the interrupt function.
- Variant II - Receiving data with interrupt.

The operation was carried out in the following order:
configuration of real-time data collection parameters
- building and running the model
- reading of collected data
- data visualization

First, the *Measure function execution times* was enabled in *Configuration Parameters > Code Generation > Verification*; additionally, *Save options > All data* was selected for real-time collection of all data. Following configuration, the model was built and started by selecting *Build, Deploy & Start*. Within the next step, data was called out in the command window for upload into the Matlab workspace. [5]

```
codertarget.profile.getData('RegistrationVN100')
```

Then generated the report (figure 10-11) and timeline for the current session (figure 12-13) by the following commands:
```
>> executionProfile.report
>> executionProfile.timeline
```

Below are reports for the two models, which make it possible to compare the number of operations performed by the processor and their execution time. Within the first step, the program is initialized; next, the individual tasks are performed sequentially according to their assigned priorities.

| Section                | Maximum Turnaround Time in ns | Average Turnaround Time in ns | Maximum Execution Time in ns | Average Execution Time in ns | Calls |
|------------------------|-------------------------------|--------------------------------|-----------------------------|-----------------------------|-------|
| (-) registrationVN100_initialize SCI Rx-C | 8610                          | 8610                           | 8610                         | 8610                         | 1     |
| (-) registrationVN100_step [0.025] SCI Rx-C | 205060820                     | 20498260                       | 205060820                    | 20498260                    | 4     |
|                       | 20432040                      | 20390290                       | 20432040                    | 20390290                    | 4     |
| registrationVN100_str2double | 5970                          | 5970                           | 5970                         | 5970                         | 4     |
| registrationVN100_str2double | 5950                          | 5950                           | 5950                         | 5950                         | 4     |
| registrationVN100_str2double | 5950                          | 5950                           | 5950                         | 5950                         | 4     |
| registrationVN100_str2double_h | 7195                          | 7195                           | 7195                         | 7195                         | 4     |
| registrationVN100_str2double_h | 7195                          | 7195                           | 7195                         | 7195                         | 4     |
| registrationVN100_str2double_h | 7195                          | 5396                           | 7195                         | 5396                         | 4     |
| registrationVN100_str2double_h | 7200                          | 7200                           | 7200                         | 7200                         | 3     |
| registrationVN100_str2double_g | 6405                          | 6405                           | 6405                         | 6405                         | 3     |
| registrationVN100_str2double_g | 6955                          | 6955                           | 6955                         | 6955                         | 3     |
| registrationVN100_str2double_g | 6925                          | 6925                           | 6925                         | 6925                         | 3     |
| registrationVN100_str2double_g | 6420                          | 6420                           | 6420                         | 6420                         | 3     |
| registrationVN100_str2double_g | 6405                          | 6405                           | 6405                         | 6405                         | 3     |
| registrationVN100_str2double_g | 6385                          | 6385                           | 6385                         | 6385                         | 3     |

*Figure 10. Code execution profiling report for variant I.*

Following analysis of data from the report, durations of individual tasks on the generated timeline were compared. The difference between the two programs in terms of duration can be observed. The models’ time step has been set at 0.025s, which means that the full program cycle must be completed within that time. In variant I, total duration of each task takes about 80% of a single cycle, which means that the processor is heavily loaded during operation of the program. In variant II, where the interrupt function is used when receiving data, the tasks are executed in 0.00012s, so that the processor is only loaded for 0.48% of cycle duration. Based on the report, the generated code can be precisely analyzed and optimized it by adjustment of settings of individual blocks or functions. In this case, the optimum processor performance was achieved by addition of a hardware interrupt, which only triggered the function at the moment of frame arrival.
Figure 11. Code execution profiling report for variant II.

Figure 12. Estimated execution timeline for variant I.
4. Conclusion
The article presents the possibilities of programming Texas Instruments processors with MATLAB Simulink Embedded Coder, based on the example of data receiving and reading from the AHRS module. The analysis shows that:
(1) Modern tools enable fast code generation and optimization without interfering with its structures, which saves time and reduces costs.
(2) Many tests can be run in a short period of time, especially those used for testing designed control systems.
(3) The generation of reports enables analysis of the examined model and the generated code, which makes its quick optimization possible.
(4) If the effect of the generated code is unsatisfactory, C code and own library can be added.
(5) Unfortunately, the generated code also contains many extended automatic functions, which increase the processor’s working time and cause lack of full control by the user.

The example described above shows how individual blocks and their configurations affect the functioning of the whole program. Owing to the generated reports, it can be checked whether a given code meets the requirements of real-time operation. This makes it possible to customize the program to the needs of the user, who can assess whether the program is sufficiently efficient and functional. Owing to time savings and the possibility to run many tests at low cost, these methods are used in the designing of control systems, among others in the space industry, where it is particularly important to achieve the highest possible computer performance in many complex and interrelated operations such as navigation and guidance, communication and control.

References
[1] Elrajoubi A and Simon S Ang 2017 TMS320F28335DSP Programming using MATLAB Simulink Embedded Coder: Techniques and Advancements IEEE 18th Workshop on Control and Modeling for Power Electronics (COMPEL)
[2] Texas Instruments 2019 TMS320F2837xD Dual-Core Delfino Microcontrollers Technical
Reference Manual

[3] VectorNav Technologies 2017 VN-100 User Manual, rev 2.43

[4] https://uk.mathworks.com/help/supportpkg/texasinstrumentsc2000/index.html?s_tid=CRUX_lfinav, Embedded Coder Support Package for Texas Instruments C2000 Processors MathWorks

[5] https://uk.mathworks.com/help/supportpkg/armcortexa/examples/real-time-code-execution-profiling.html, Real-Time Code Execution Profiling MathWorks