



MathWorks | Book Program



Optymalizacja deploymentu funkcji ze środowiska MATLAB na platformy Cortex-M4



Dr hab. inż. Adam Jabłoński, prof. AGH
Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie
AGH University of Krakow

MATLAB EXPO, Warszawa, 04.06.2024 r.

Opis problemu

my_signal_processing.m

```
function out = my_signal_processing(x)
% my code goes here
out = fft(x);
% my code goes here
end
```



Urządzenia
wbudowane



ARM Cortex znajdują zastosowanie w szerokiej gamie urządzeń elektronicznych, takich jak smartfony, tablety, smartwatche, nowoczesny sprzęt AGD i samochody.



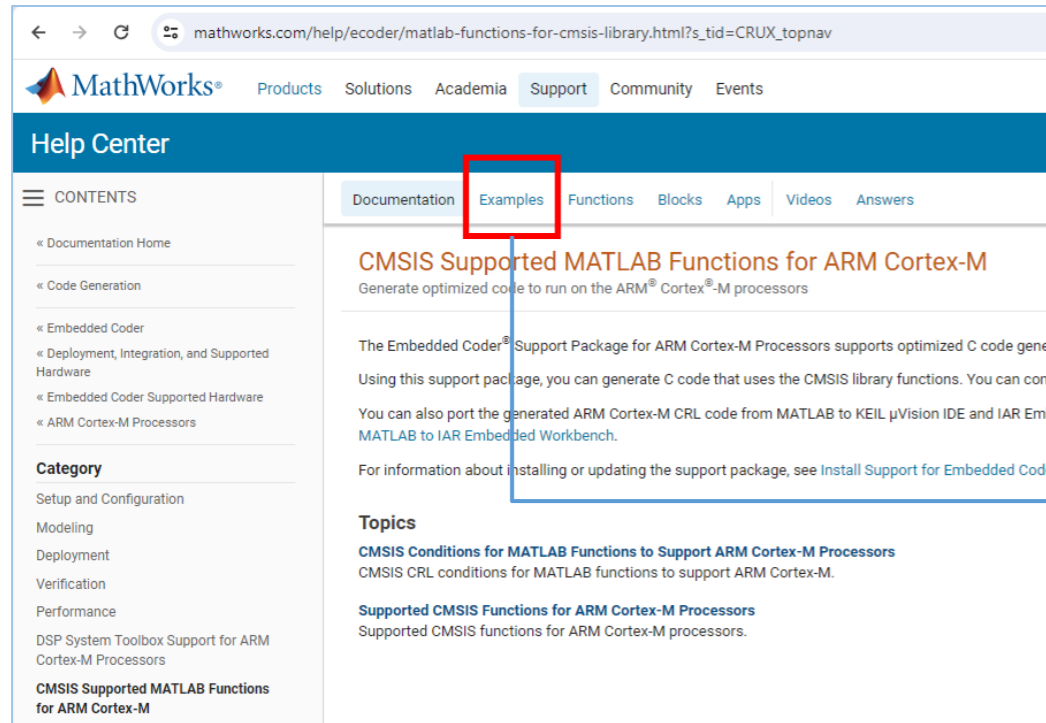
Signal Processing



Output

Dzięki narzędziu MATLAB CRL oraz dostosowaniu *m-kodu* do CMSIS DSP, możliwe jest automatyczny deployment algorytmów

Motywacja tematu prezentacji



mathworks.com/help/ecoder/matlab-functions-for-cmsis-library.html?s_tid=CRUX_topnav

MathWorks® Products Solutions Academia Support Community Events

Help Center

CONTENTS

- « Documentation Home
- « Code Generation
- « Embedded Coder
 - « Deployment, Integration, and Supported Hardware
 - « Embedded Coder Supported Hardware
 - « ARM Cortex-M Processors

Category

- Setup and Configuration
- Modeling
- Deployment
- Verification
- Performance
- DSP System Toolbox Support for ARM Cortex-M Processors
- CMSIS Supported MATLAB Functions for ARM Cortex-M**

Documentation Examples Functions Blocks Apps Videos Answers

CMSIS Supported MATLAB Functions for ARM Cortex-M

Generate optimized code to run on the ARM® Cortex®-M processors

The Embedded Coder® Support Package for ARM Cortex-M Processors supports optimized C code generation. Using this support package, you can generate C code that uses the CMSIS library functions. You can compile the generated C code for ARM Cortex-M processors.

You can also port the generated ARM Cortex-M CRL code from MATLAB to KEIL µVision IDE and IAR Embedded Workbench.

For information about installing or updating the support package, see Install Support for Embedded Coder.

Topics

- CMSIS Conditions for MATLAB Functions to Support ARM Cortex-M Processors**
CMSIS CRL conditions for MATLAB functions to support ARM Cortex-M.
- Supported CMSIS Functions for ARM Cortex-M Processors**
Supported CMSIS functions for ARM Cortex-M processors.



Documentation Examples Functions Blocks Apps Videos Answers

CMSIS Supported MATLAB Functions for ARM Cortex-M – Examples

No results in the CMSIS Supported MATLAB Functions for ARM Cortex-M category.

Related categories with results:

- [ARM Cortex-M Processors \(10\)](#)
- [Modeling \(1\)](#)
- [Verification \(1\)](#)
- [Performance \(1\)](#)
- [DSP System Toolbox Support for ARM Cortex-M Processors \(7\)](#)

Do dziś (27.05.2024 r.) w zakładce „Examples” MATLAB (2024a) strona *MathWorks* wyświetla: „No results”.

Rozwiązanie problemu

Krok 1: analiza „*MATLAB conditions*” dla używanych funkcji

Krok 2: instalacja komponentów (EC, HSP, CMSIS, compiler)

Krok 3: analiza bibliotek producenta (weryfikacja krzyżowa)

Krok 4: przygotowanie skryptu demo

Krok 5: obsługa procedury Wizarda

Krok 6: weryfikacja via Coder Report Viewer (CRV)

Krok 7: pakowanie zip





Rozwiązanie problemu

Krok 1: analiza „*MATLAB conditions*”

Krok 2: instalacja komponentów

Krok 3: analiza bibliotek producenta

Krok 4: przygotowanie skryptu demo

Krok 5: obsługa procedury Wizarda

Krok 6: weryfikacja via CRV

Krok 7: pakowanie zip

CMSIS Conditions for MATLAB Functions to Support ARM Cortex-M Processors R2024a

Each MATLAB[®] function that can be used with the Support Package for ARM[®] Cortex[®]-M processors requires specific conditions to allow code replacement by using the CMSIS Library. You use this code replacement when code.

If you do not meet the specific requirements, then the generated C code runs on the ARM Cortex-M processors. However, this generated code does not use CMSIS library support.

The CMSIS library supports these MATLAB functions only when you set specific properties, as indicated in this table:

MATLAB Function	Supported Function Signatures	Input Requirements	Parameter Requirements
fft	$Y = \text{fft}(X)$ $Y = \text{fft}(X, n)$ $Y = \text{fft}(X, n, \text{dim})$	<ul style="list-style-type: none"> Real or Complex Values Multichannel Input (Multiple columns) single data type 	Transform Length n: <ul style="list-style-type: none"> Real Input: n must be in the range [9 2048] U {4096} - {16}. Complex Input: n must be in the range [5 2048] U {4096} - {8}. Dimension to operate along dim: dim must be a compile time constant.



Rozwiązanie problemu

Krok 1: analiza „MATLAB conditions”

Krok 2: instalacja komponentów

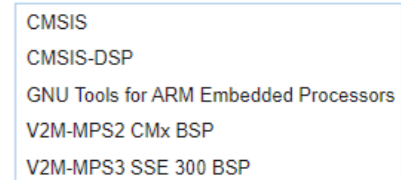
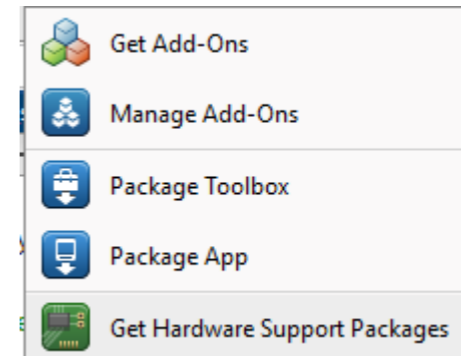
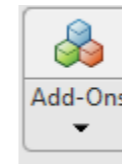
Krok 3: analiza bibliotek producenta


Krok 4: przygotowanie skryptu demo

Krok 5: obsługa procedury Wizarda

Krok 6: weryfikacja via CRV

Krok 7: pakowanie zip




1)  Arm_Compiler_Support_Package

2)  Embedded Coder

3) **MATLAB Support for MinGW-w64**

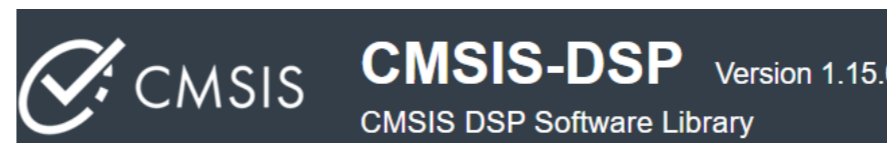
4)  **CMSIS** **CMSIS-DSP** Version 1.15.0
CMSIS DSP Software Library

5)  **Embedded Coder Support Package for ARM Cortex-M Processors** by MathWorks Embedded Coder Team **STAFF**
Generate code optimized for **Cortex-M** processors.
Embedded Coder® Support Package for ARM® **Cortex**®-M Processors lets you generate optimized code for math operations using the CMSIS library. Use this generated code for ARM **Cortex-M** processors. For



Rozwiązanie problemu

Krok 1: analiza „MATLAB conditions”



Krok 2: instalacja komponentów

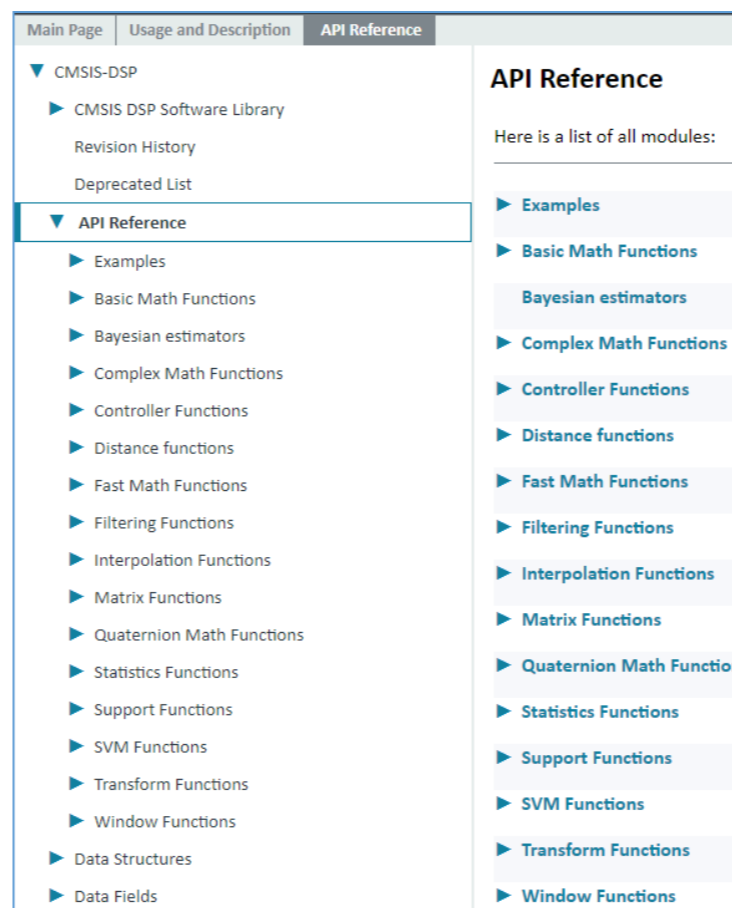
Krok 3: analiza bibliotek producenta

Krok 4: przygotowanie skryptu demo

Krok 5: obsługa procedury Wizarda

Krok 6: weryfikacja via CRV

Krok 7: pakowanie zip



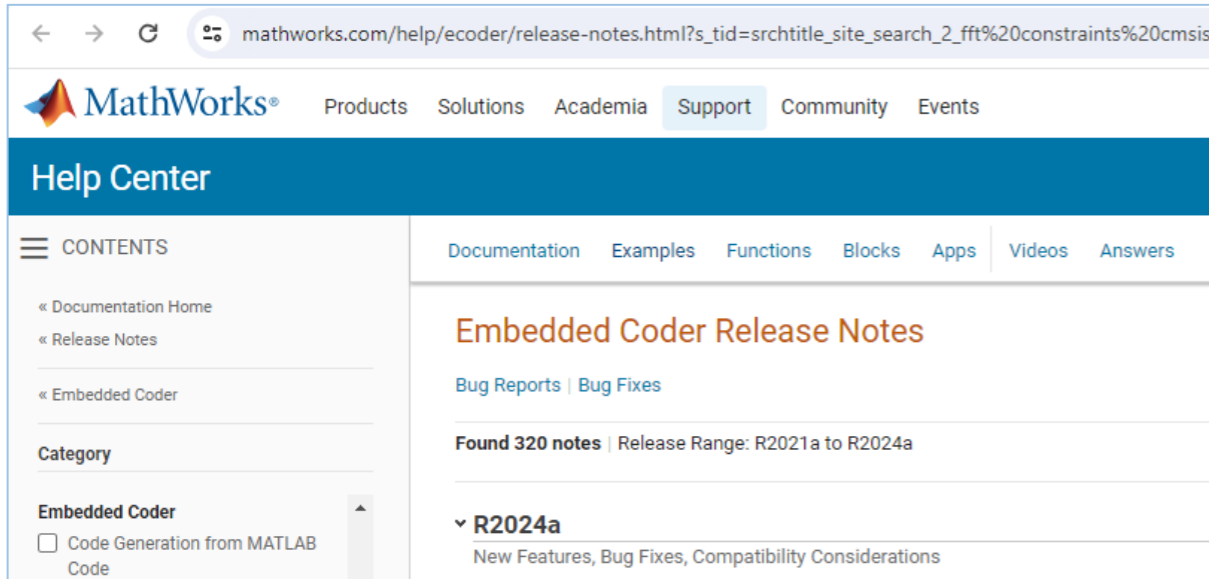
```

▼ Real FFT F32 Functions
arm_rfft_fast_f32

arm_rfft_fast_init_1024_f32
arm_rfft_fast_init_128_f32
arm_rfft_fast_init_2048_f32
arm_rfft_fast_init_256_f32
arm_rfft_fast_init_32_f32
arm_rfft_fast_init_4096_f32
arm_rfft_fast_init_512_f32
arm_rfft_fast_init_64_f32
arm_rfft_fast_init_f32

void arm_rfft_fast_f32 ( const arm_rfft_fast_instance_f32 * S,
                        float32_t * p,
                        float32_t * pOut,
                        uint8_t ifftFlag
                      )
    
```

Krok 3: analiza bibliotek producenta - BTW...



mathworks.com/help/ecoder/release-notes.html?s_tid=srchtitle_site_search_2_fft%20constraints%20cmsis

MathWorks® Products Solutions Academia Support Community Events

Help Center

CONTENTS

- « Documentation Home
- « Release Notes
- « Embedded Coder

Category

Embedded Coder

- Code Generation from MATLAB Code

Documentation Examples Functions Blocks Apps Videos Answers

Embedded Coder Release Notes

Bug Reports | Bug Fixes

Found 320 notes | Release Range: R2021a to R2024a

▼ R2024a
New Features, Bug Fixes, Compatibility Considerations

▼ ARM Cortex-M Processors: CMSIS CRL Support for Additional Math Functions

Starting in R2024a, you can use the Embedded Coder Support Package for ARM Cortex-M Processors to generate code for the following math functions:

- Scale/Gain
- Bias/Offset
- Dot Product
- Saturate/Limit
- Unary Minus

▼ ARM Cortex-M Processors: Multichannel support for DSP Blocks and System object Code Replacement

The ARM Cortex-M CMSIS CRL now supports multichannel in these DSP blocks and System objects:

- Biquad Filter
- FFT
- IFFT
- Discrete FIR Filter
- Decimator
- Interpolator
- dsp.SOSFilter
- dsp.FFT
- dsp.IFFT
- dsp.FIRFilter
- dsp.FIRDecimator
- dsp.FIRInterpolator

▼ ARM Cortex-M Processors: Code Generation for Sound Classification on ARM Cortex-M Targets with CMSIS-NN

The Code Generation for Sound Classification on ARM Cortex-M Targets using CMSIS-NN example shows how to classify white noise, brown noise and pink noise by generating a PIL MEX function, which allows you to execute

Z każdą nową wersją środowiska MATLAB, zwiększa się ilość wspieranych funkcji interfejsu CMSIS w ramach CRL.

Rozwiązanie problemu

Krok 1: analiza „MATLAB conditions”

Krok 2: instalacja komponentów


Krok 3: analiza bibliotek producenta

Krok 4: przygotowanie skryptu demo

Krok 5: obsługa procedury Wizarda

Krok 6: weryfikacja via CRV

Krok 7: pakowanie zip

 my_signal_processing.m

```
function out = my_signal_processing(x)
% my code goes here
out = fft(x);
% my code goes here
end
```

```
calling_script.m x +
1 clear; clc; close all;
2
3 x = rand(1,1e3);
4 → out = my_signal_processing(x);
5
6 disp(out)
```

Skrypt demo umożliwia automatyczną sugestię typów i rozmiarów parametrów wejściowych oraz sprawdzenie wykonywalności funkcji.

Rozwiązanie problemu

Krok 1: analiza „MATLAB conditions”

Krok 2: instalacja komponentów

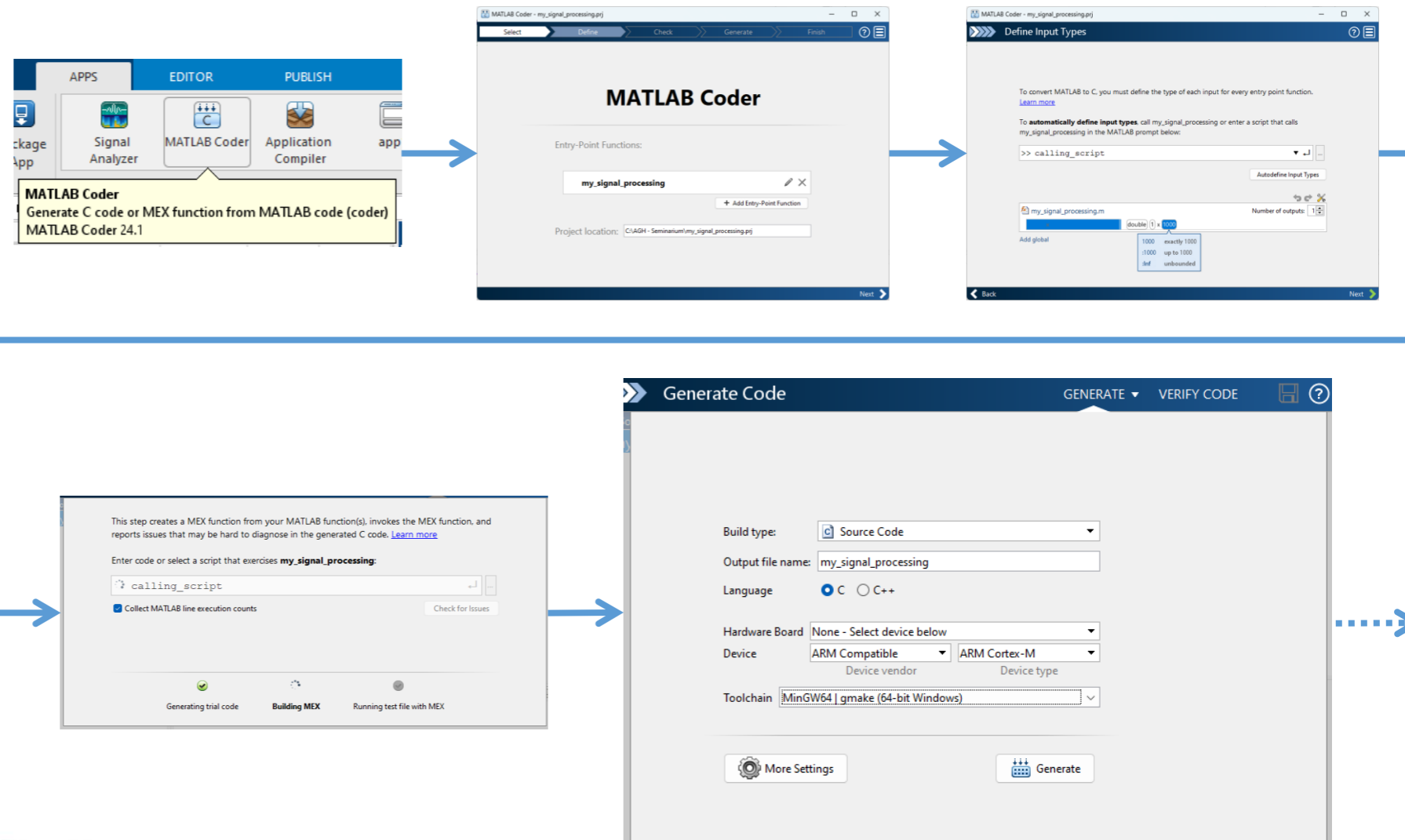
Krok 3: analiza bibliotek producenta

Krok 4: przygotowanie skryptu demo

Krok 5: obsługa procedury Wizarda

Krok 6: weryfikacja via CRV

Krok 7: pakowanie zip



Rozwiązanie problemu

Krok 1: analiza „MATLAB conditions”

Krok 2: instalacja komponentów

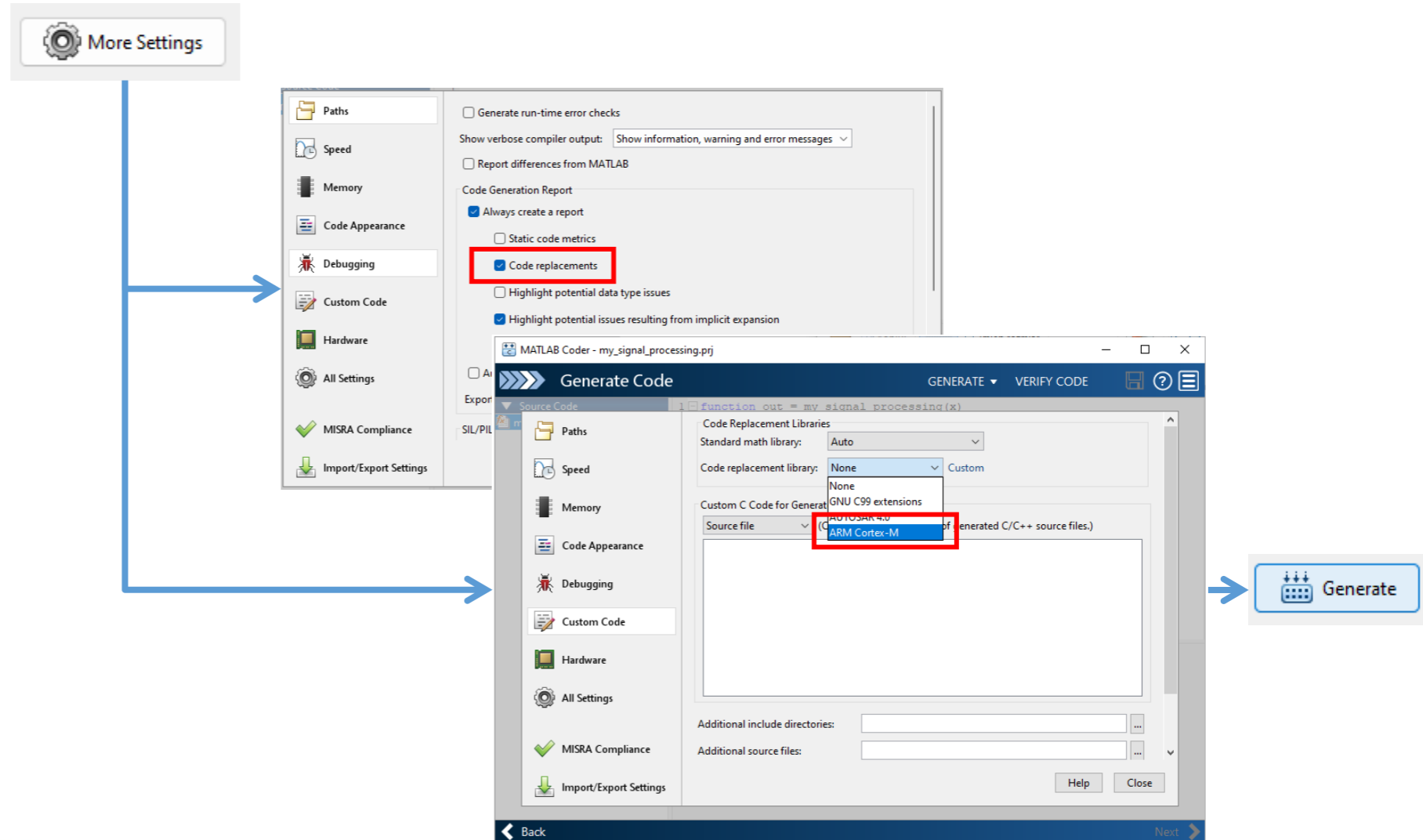
Krok 3: analiza bibliotek producenta

Krok 4: przygotowanie skryptu demo

Krok 5: obsługa procedury Wizarda

Krok 6: weryfikacja via CRV

Krok 7: pakowanie zip



The image shows a sequence of steps in the MATLAB Coder interface:

- A "More Settings" button is clicked, leading to a settings dialog where "Code replacements" is checked.
- The "Generate Code" wizard is shown, with "Code Replacement Libraries" set to "None" and "Source file" set to "ARM Cortex-M".
- A "Generate" button is highlighted, indicating the final step in the process.

Rozwiązanie problemu

Krok 1: analiza „MATLAB conditions”

Krok 2: instalacja komponentów

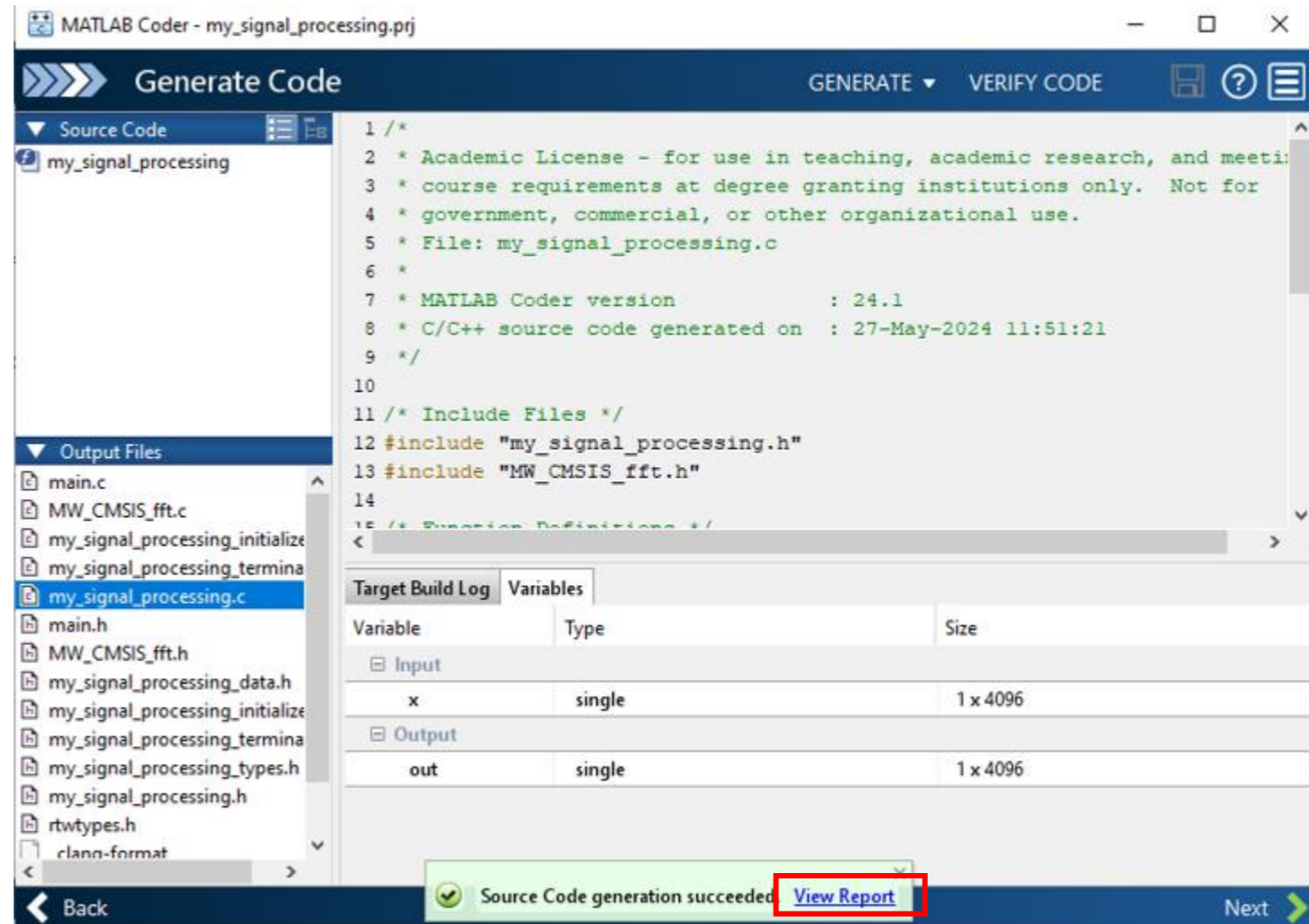
Krok 3: analiza bibliotek producenta

Krok 4: przygotowanie skryptu demo

Krok 5: obsługa procedury Wizarda

Krok 6: weryfikacja via CRV

Krok 7: pakowanie zip



The screenshot shows the MATLAB Coder interface for a project named 'my_signal_processing'. The 'Generate Code' window is active, displaying the source code for 'my_signal_processing.c'. The code includes a license notice, MATLAB version information (24.1), and include files for 'my_signal_processing.h' and 'MW_CMSIS_fft.h'. Below the code editor, the 'Variables' section shows the following table:

Variable	Type	Size
Input		
x	single	1 x 4096
Output		
out	single	1 x 4096

At the bottom of the window, a green status bar indicates 'Source Code generation succeeded' with a 'View Report' button highlighted in red. The 'Output Files' list on the left includes files like 'main.c', 'MW_CMSIS_fft.c', and 'my_signal_processing.c'.

Rozwiązanie problemu

Krok 1: analiza „MATLAB conditions”

Krok 2: instalacja komponentów

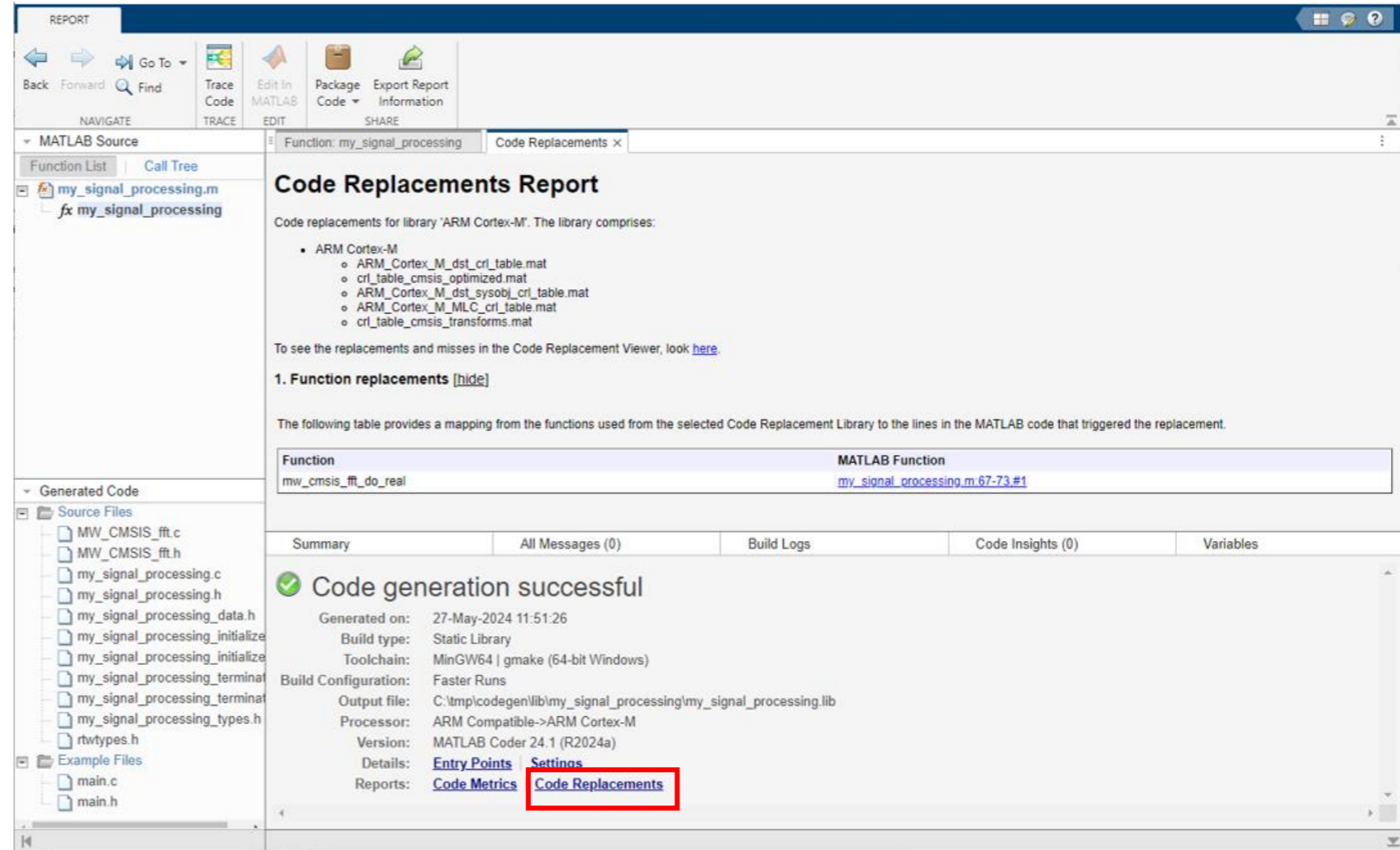
Krok 3: analiza bibliotek producenta

Krok 4: przygotowanie skryptu demo

Krok 5: obsługa procedury Wizarda

Krok 6: weryfikacja via CRV

Krok 7: pakowanie zip



Code Replacements Report

Code replacements for library 'ARM Cortex-M'. The library comprises:

- ARM Cortex-M
 - ARM_Cortex_M_dst_cri_table.mat
 - cri_table_cmsis_optimized.mat
 - ARM_Cortex_M_dst_sysobj_cri_table.mat
 - ARM_Cortex_M_MLC_cri_table.mat
 - cri_table_cmsis_transforms.mat

To see the replacements and misses in the Code Replacement Viewer, look [here](#).

1. Function replacements [hide]

The following table provides a mapping from the functions used from the selected Code Replacement Library to the lines in the MATLAB code that triggered the replacement.

Function	MATLAB Function
mw_cmsis_fft_do_real	my_signal_processing.m:67-73.#1

Summary | All Messages (0) | Build Logs | Code Insights (0) | Variables

Code generation successful

Generated on: 27-May-2024 11:51:26
 Build type: Static Library
 Toolchain: MinGW64 | gmake (64-bit Windows)
 Build Configuration: Faster Runs
 Output file: C:\tmp\codegen\lib\my_signal_processing\my_signal_processing.lib
 Processor: ARM Compatible->ARM Cortex-M
 Version: MATLAB Coder 24.1 (R2024a)
 Details: [Entry Points](#) | [Settings](#)
 Reports: [Code Metrics](#) | [Code Replacements](#)

Rozwiązanie problemu

Krok 1: analiza „MATLAB conditions”

Krok 2: instalacja komponentów

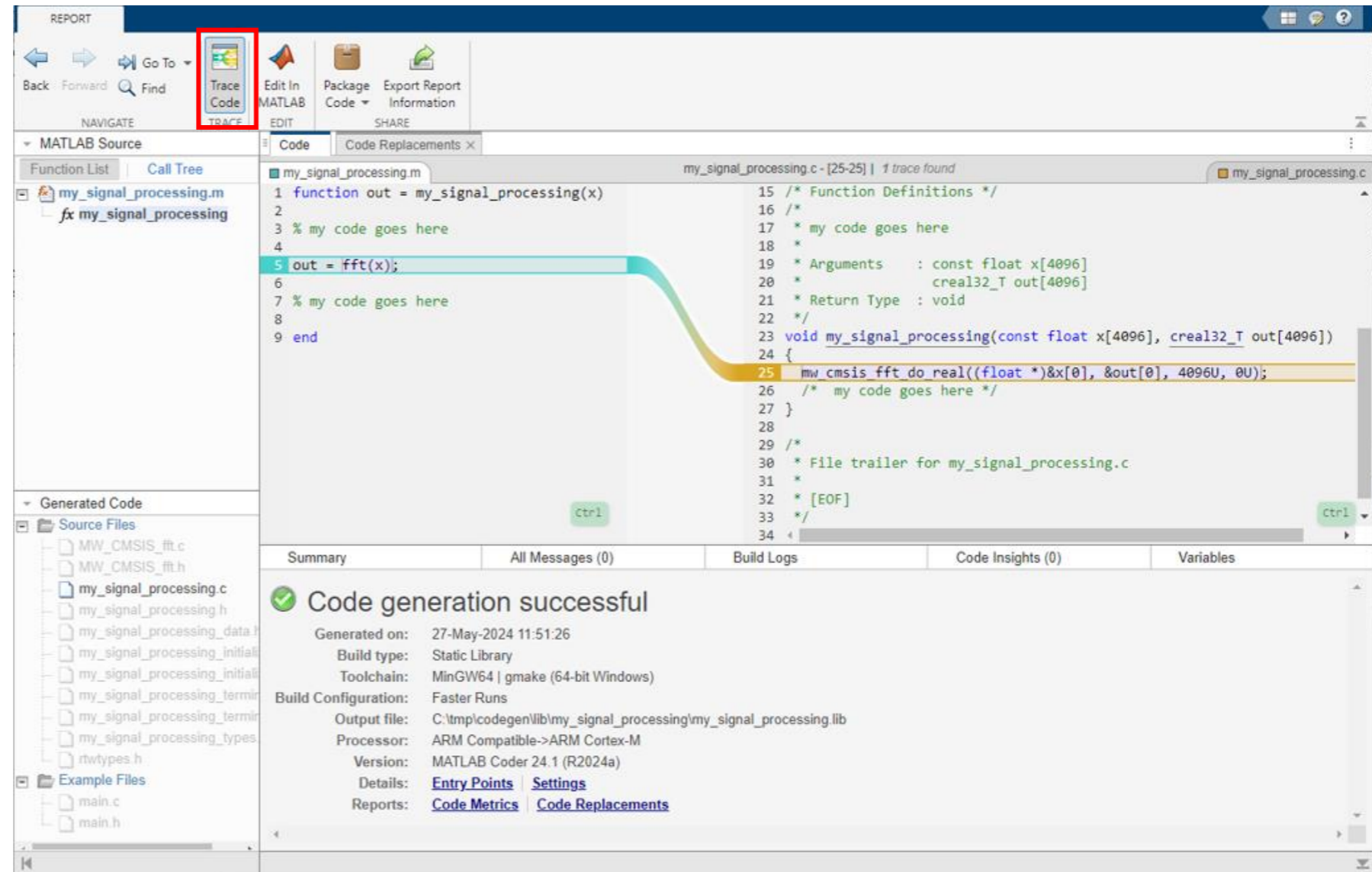
Krok 3: analiza bibliotek producenta

Krok 4: przygotowanie skryptu demo

Krok 5: obsługa procedury Wizarda

Krok 6: weryfikacja via CRV

Krok 7: pakowanie zip



The screenshot displays the MATLAB environment with the following components:

- REPORT** tab selected at the top.
- Trace Code** button highlighted with a red box in the top toolbar.
- MATLAB Source** pane on the left showing the function `my_signal_processing.m`.
- Code** window showing the MATLAB function definition and its C implementation. The C code includes a call to `mw_cmsis_fft_do_real`.
- Generated Code** pane on the left showing the source files for the generated code.
- Summary** pane at the bottom showing the status: **Code generation successful**.

Krok 6: weryfikacja via CRV: Porównanie NOK vs OK...

```
12 #include "my_signal_processing.h"
13 #include "FFTImplementationCallback.h"
14 #include "my_signal_processing_data.h"
```

```
12 #include "my_signal_processing.h"
13 #include "MW_CMSIS_fft.h"
```

MW_CMSIS_fft.c
MW_CMSIS_fft.h

Code Replacements Report

Code replacements for library 'ARM Cortex-M'. The library comprises:

- ARM Cortex-M
 - ARM_Cortex_M_dst_cr1_table.mat
 - cr1_table_cmsis_optimized.mat
 - ARM_Cortex_M_dst_sysobj_cr1_table.mat
 - ARM_Cortex_M_MLC_cr1_table.mat
 - cr1_table_cmsis_transforms.mat

To see the replacements and misses in the Code Replacement Viewer, look [here](#).

1. Function replacements [\[hide\]](#)

The following table provides a mapping from the functions used from the selected Code Replacement Library to the lines in the MATLAB code that triggered the replacement.

Function	MATLAB Function
memset	Replacements triggered in internal file: FFTImplementationCallback.m

Summary All Messages (0) Build Logs Code Insights (0) Variables

Code generation successful

Generated on: 27-May-2024 11:34:51
 Build type: Static Library
 Toolchain: MinGW64 | gmake (64-bit Windows)
 Build Configuration: Faster Runs
 Output file: C:\tmp\codegen\lib\my_signal_processing\my_signal_processing.lib
 Processor: ARM Compatible->ARM Cortex-M
 Version: MATLAB Coder 24.1 (R2024a)
 Details: [Entry Points](#) | [Settings](#)
 Reports: [Code Metrics](#) | [Code Replacements](#)

x = rand(1,1e3);

Code Replacements Report

Code replacements for library 'ARM Cortex-M'. The library comprises:

- ARM Cortex-M
 - ARM_Cortex_M_dst_cr1_table.mat
 - cr1_table_cmsis_optimized.mat
 - ARM_Cortex_M_dst_sysobj_cr1_table.mat
 - ARM_Cortex_M_MLC_cr1_table.mat
 - cr1_table_cmsis_transforms.mat

To see the replacements and misses in the Code Replacement Viewer, look [here](#).

1. Function replacements [\[hide\]](#)

The following table provides a mapping from the functions used from the selected Code Replacement Library to the lines in the MATLAB code that triggered the replacement.

Function	MATLAB Function
mw_cmsis_fft_do_real	my_signal_processing.m:67-73.#1

Summary All Messages (0) Build Logs Code Insights (0) Variables

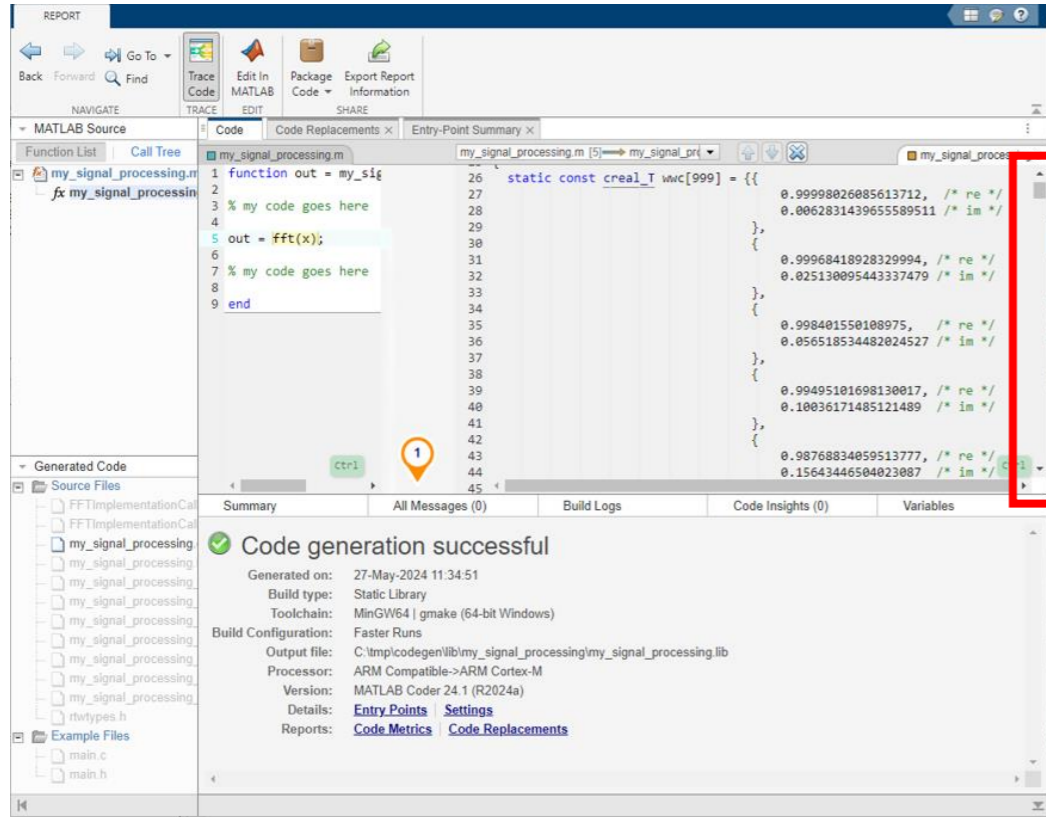
Code generation successful

Generated on: 27-May-2024 11:51:26
 Build type: Static Library
 Toolchain: MinGW64 | gmake (64-bit Windows)
 Build Configuration: Faster Runs
 Output file: C:\tmp\codegen\lib\my_signal_processing\my_signal_processing.lib
 Processor: ARM Compatible->ARM Cortex-M
 Version: MATLAB Coder 24.1 (R2024a)
 Details: [Entry Points](#) | [Settings](#)
 Reports: [Code Metrics](#) | [Code Replacements](#)

x = single(rand(1,4096));

VS

Krok 6: weryfikacja via CRV: Porównanie NOK vs OK...

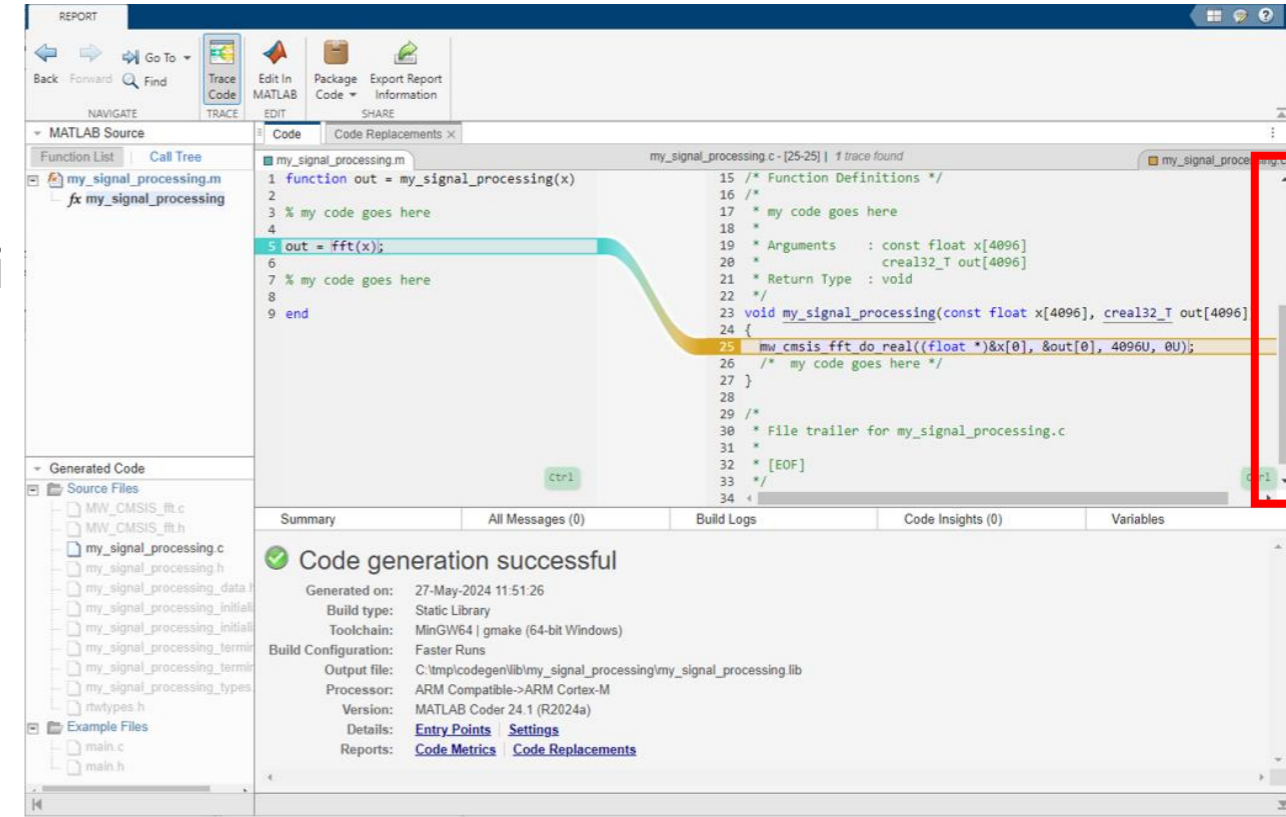


Code generation successful

Generated on: 27-May-2024 11:34:51
 Build type: Static Library
 Toolchain: MinGW64 | gmake (64-bit Windows)
 Build Configuration: Faster Runs
 Output file: C:\tmp\codegen\lib\my_signal_processing\my_signal_processing.lib
 Processor: ARM Compatible->ARM Cortex-M
 Version: MATLAB Coder 24.1 (R2024a)
 Details: [Entry Points](#) | [Settings](#)
 Reports: [Code Metrics](#) | [Code Replacements](#)

`x = rand(1,1e3);`

5000+ linii
 VS
 4 linijki



Code generation successful

Generated on: 27-May-2024 11:51:26
 Build type: Static Library
 Toolchain: MinGW64 | gmake (64-bit Windows)
 Build Configuration: Faster Runs
 Output file: C:\tmp\codegen\lib\my_signal_processing\my_signal_processing.lib
 Processor: ARM Compatible->ARM Cortex-M
 Version: MATLAB Coder 24.1 (R2024a)
 Details: [Entry Points](#) | [Settings](#)
 Reports: [Code Metrics](#) | [Code Replacements](#)

`x = single(rand(1,4096));`

Rozwiązanie problemu

Krok 1: analiza „MATLAB conditions”

Krok 2: instalacja komponentów

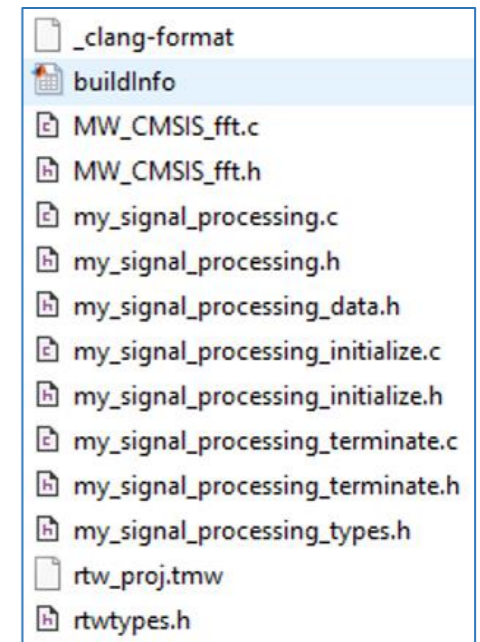
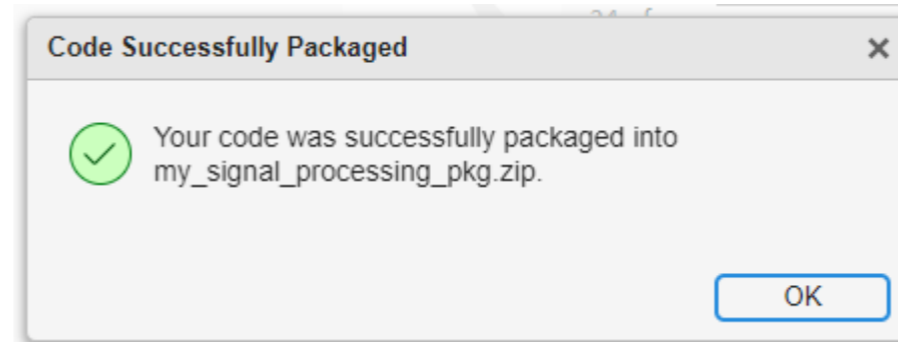
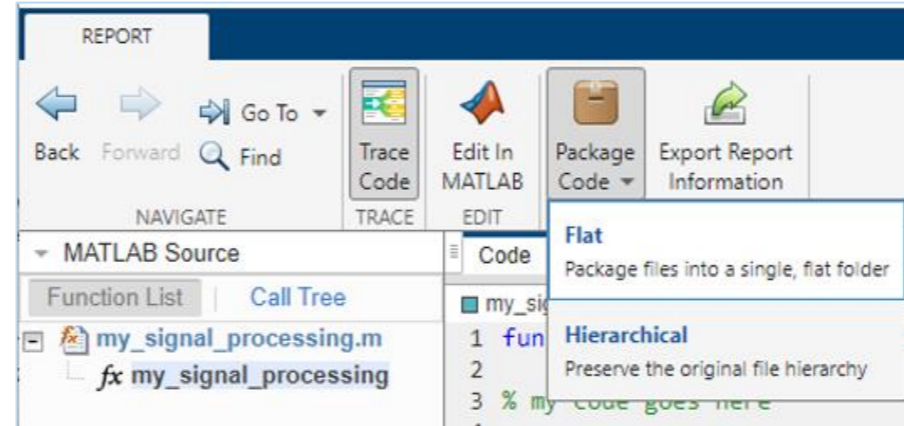
Krok 3: analiza bibliotek producenta

Krok 4: przygotowanie skryptu demo

Krok 5: obsługa procedury Wizarda

Krok 6: weryfikacja via Wizard Viewer

Krok 7: pakowanie zip



Praktyczne wskazówki

@Krok 1: funkcje z *DSP System Objects* (filtry) są opisane osobno od funkcji z rodziny "FFT"

@Krok 2: po instalacji wtyczek, MATLAB musi być zrestartowany

@Krok 3: zaleca się uruchomienie „*crviewer*” w *Command Window*

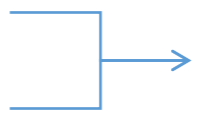
@Krok 4: zaleca się centralizację kodu MATLAB oraz statyczną konwersję precyzji

@Krok 5: konkretne projekty wymagają dodatkowej minimalizacji alokacji pamięci

@Krok 6: ręczna analiza „*Code Replacements*” jest zalecana do weryfikacji

@Krok 7: użycie interfejsu *zip* zapewnia, że dodane zostaną odpowiednie biblioteki MATLAB

@deployment filtrów: referencja [4] pokazuje rozwiązanie półautomatyczne; wzorując się na prezentacji można zautomatyzować proces, uważając, aby cutoff był wyraźnie różny od wartości granicznych zakresu widmowego

dsp.BiquadFilter
dsp.SOSFilter  *mw_arm_biquad_cascade_df2T_f32*

Dziękuję za uwagę!

Bibliografia:

- [1] Adam Jablonski, *Condition Monitoring Algorithms in MATLAB*, Springer, 2021
- [2] https://www.mathworks.com/help/ecoder/matlab-functions-for-cmsis-library.html?s_tid=CRUX_lftnav
- [3] <https://www.arm.com/technologies/cmsis>
- [4] <https://gaidi.ca/weblog/implementing-a-biquad-cascade-iir-filter-on-a-cortex-m4/>

