Renesas RA Family

Motor Failure Detection Example by TensorFlow for RA6T1

Introduction

Improving operational reliability is an important milestone for increasing stability and quality. Maintenance is one of the crucial disciplines that harbors great potential for improvement and cost savings and increases the operational reliability at the same time. New technologies like the Embedded AI (e-AI) leads to a significant improvement within the maintenance discipline. Specifically, within the motor driven systems, e-AI does not only make maintenance more efficient but also enables understanding of the reasons which could lead to downtime and enables also preventing them in the first place.

In this example, the maintenance discipline has been realized and implemented by involving a simple use case that is able to detect an abnormal state within the system behavior as a potential failure. Implemented e-AI algorithm (based on TensorFlow) discovers a failure due to an abnormal load condition of the motor. The abnormal load condition is detected by monitoring the current and rotation speed by a so-called inference model representing a trained neural network. Using only a single MCU (RA6T1), the motor control and fault detection algorithm are run simultaneously.

Target Device

RA6T1

Supported Kit

YROTATE-IT-RA6T1

Supported FSP version

FSP v2.3.0 or later

Reference Documents

1. YROTATE-IT-RA6T1 User Manual (UM-YROTATE-IT-RA6T1)
2. e-AI Translator V1.6.0 User Manual (REN_r20ut4135ej0600-e-ai__20201106)

Contents

1. Overview

Figure 1 shows the high-level system block diagram of the RA6T1 Motor Failure Detection Example with TensorFlow. This is an e-AI based motor system containing the learned DNN and brushless DC motor control MCU software.

A single MCU is simultaneously controlling the motor and monitoring its operation. This is accomplished by a so-called e-AI inference model representing the implementation of a trained neural network analyzing the motor current. The judgment result is displayed on the PC software.

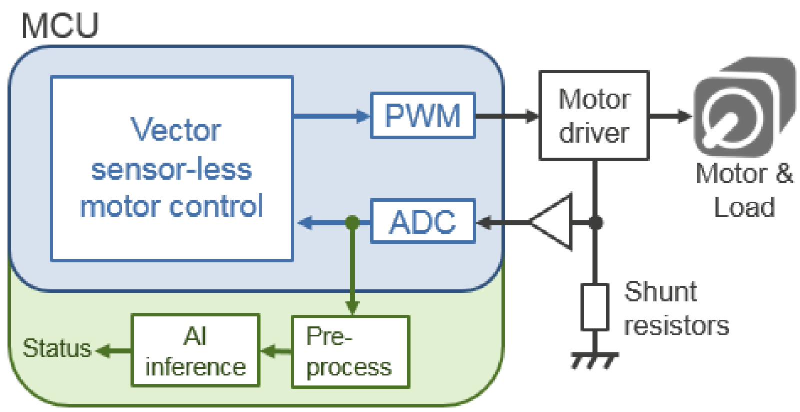

Figure 1. Application System Block Diagram

The system's brushless DC motor control employs the sensor less vector control method to monitor the 3-shunt current control with the A/D converter. In this system, focusing on the fact that the waveform of the 3-shunt current changes depending on the state of the motor, this 3-shunt current is used as the input of trained DNN.

1.1 Pre-Processing

The pre-processing in this example involves the following operations:

- Collect AD converter values of the three-phase current
- FFT processing of data frames (frequency spectrum generation)
- Feature point extraction from frequency spectrum (learned DNN input data generation)

1.2 AI Inference by TensorFlow

TensorFlow is an end-to-end open-source platform for machine learning (ML). It has a comprehensive, flexible ecosystem of tools, libraries, and community resources to easily build and deploy ML powered applications. Further detail can be found at https://www.tensorflow.org/about.

1.2.1 Neural Network

Neural Networks are a set of algorithms modeled after the human brain, to recognize pattern. The Neural Network is made up of neurons. These neurons are connected to each other and loaded with a weight. Each neuron has an activating function, which defines the output of the neuron. Training of a Neural Network means, learning the values of parameters: weights and bias.

1.2.2 AI Neural Network Model and its Characteristic

A lot of AI Neural Network models are available for different use cases:

- Convolutional Neural Network CNN (Image Recognition)
- Recurrent Neural Network RNN (Voice command, Voice recognition, Translation…)
- Fully Connected Neural Network (Non-Linear Systems)

1.2.3 AI Neural Network Model

In this example, simple AI by TensorFlow is made to detect normality and anomality by following layers:

1. Input layer: FFT-processed U-phase shunt current data is input to the input layer
2. Hidden layer: The hidden layer uses the fully connected layer
3. Output layer: The output layer outputs the probability of normality and anomality.

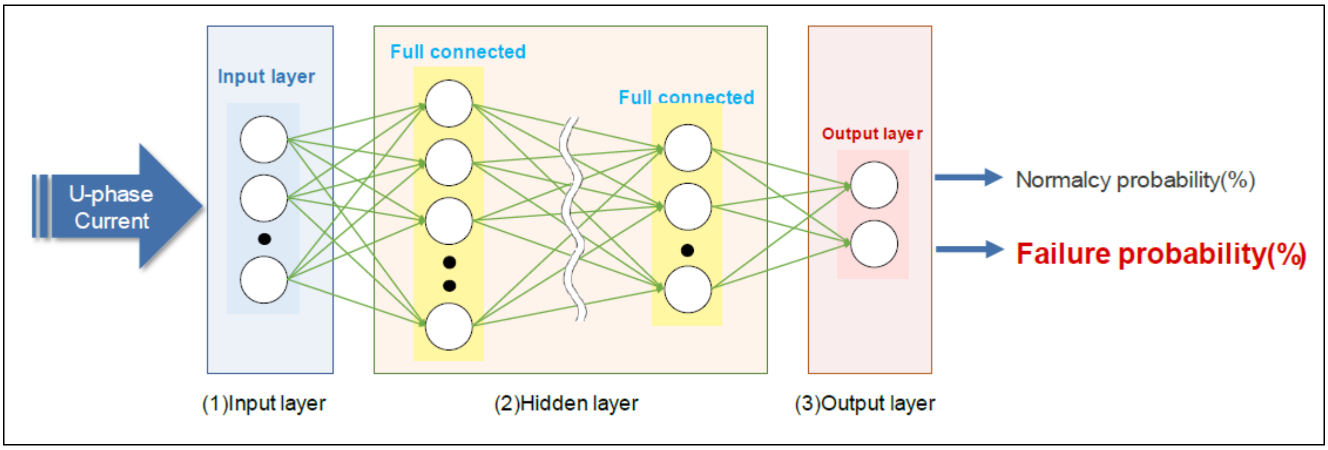

Figure 2. AI Model Configuration

2. Hardware Configuration

This section discusses hardware configuration of the RA6T1 Motor Failure Detection Example demo. This is based on the Renesas RA6T1 YROTATE-IT Motor Control Demo Kit shown as follows:

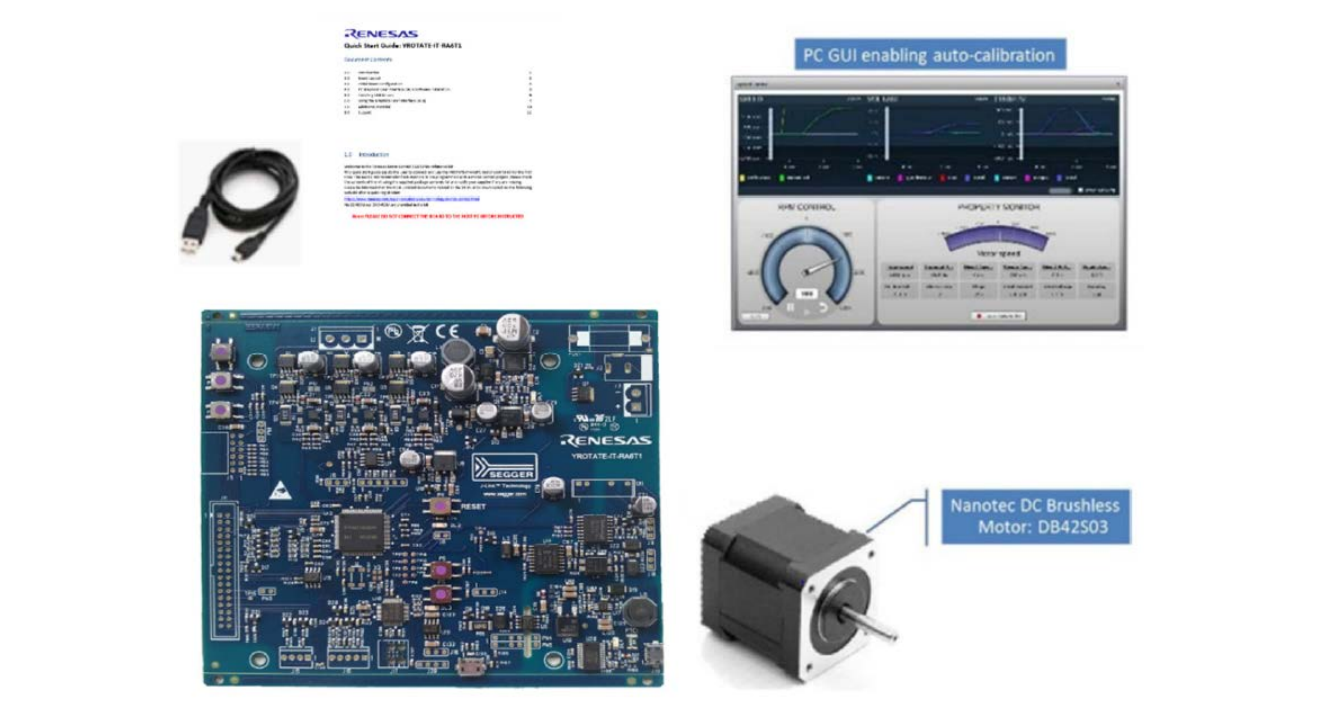

Figure 3. RA6T1 YROTATE-IT Motor Control Demo Kit

The demo kit includes:

- 3-phase brushless motor control with sensor-less/sensored Field Oriented Control algorithm
- Royalty-free embedded software source code
- Intuitive and isolated PC control Graphical User Interface
- On-board in-circuit debugger MCU; basically, no external JTAG debugger tool is needed
- Compatible 1.5 KW external power stage
- Auto-tuning and calibration
- Motor parameters identification
- Video and tutorial demonstration

For a detailed description of the software and hardware of the RA6T1 Demo Kit please refer to the *YROTATE-IT-RA6T1 User Manual* [1], which can be obtained by contacting to a local Renesas sales or distributor.

2.1 Software and Tools

Table 1 lists Software and Tools required. For Failure Detection Demo Example, Project contains Motor application and AI application with TensorFlow. Data Collection Tool and Training Tool will be used for AI model development.

Table 1. Software and Tools

Item	Description
Operating environment	OS: Windows® 10
Integrated Development Environment (IDE)	e^2 studio Version: 2021-01
Tool chain	GCC ARM Embedded 9.2.1.20191025
FSP	Version 2.3.0
Tools in demo package	Data Collection Tool Training Tool Demo Project: YROTATE_RA6T1_eAI_V1.0

3. Failure Detection Demonstration

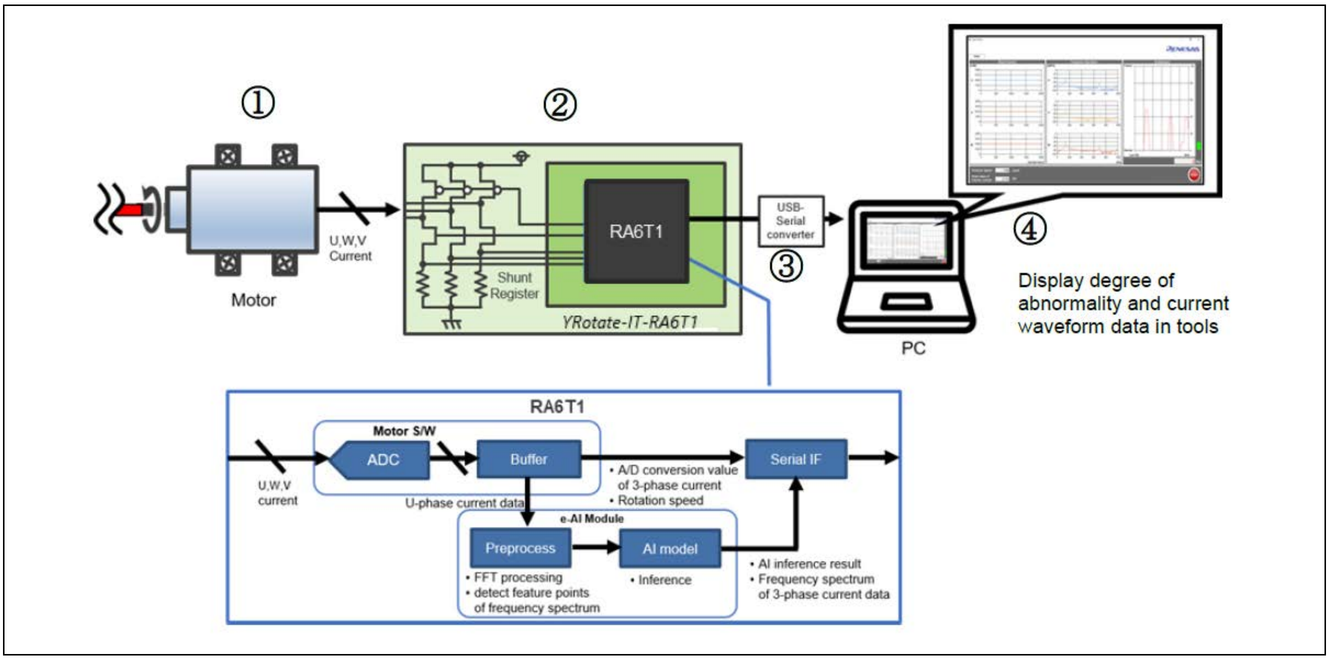

Figure 4. System Operation Flow

Following is an overview of operations:

1. Execute sensor-less vector control on motor.
   When power is applied to the kit, it starts the motor driver operations. See Reference Documents [1] for details on board operations.
2. Execute pre-processing for motor drive current data, determine anomaly using e-AI inference
   A. A/D conversion value accumulation
   B. Data pre-processing
   C. AI inference
3. Serial communication with PC
   The data is transferred to the PC using a USB cable.
4. Display degree of anomaly and current waveform data in tools.
   The received data is displayed in numerical values and graph form in the DataCollectionTool (GUI tool) on the PC.

In this example,

- Normal state is defined as when drive the motor in unloaded condition
- Anomaly state is defined as when the extra load is placed on the shafts by hand.

3.1 Preprocessing Specifications

The Motor Failure Detection Example described in this document (referred to as “target system” below) preprocesses motor drive current data for use as AI input data. The following outlines the preprocessing used by the target system.

A. **Framing** : Frames the A/D conversion value of motor drive current
   An internal timer generates the 2 kHz sampling frequency and acquires the A/D conversion value of the motor 3-shunt current. U, V and W phase among of 3-shunt current are input to the 12-bit A/D converter. One frame (512 samples) of A/ D conversion values are accumulated for the FFT. From the next frame on, A/D conversion values are accumulated by overlapping 64 samples of the previous frame.

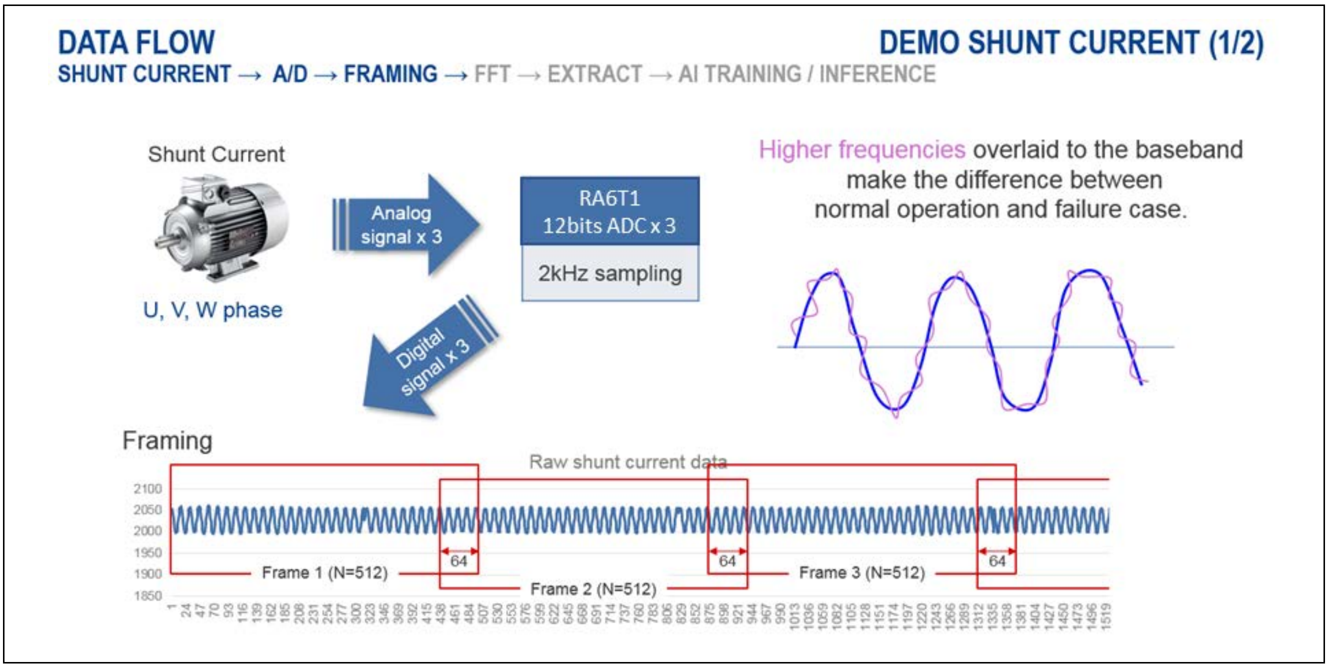

Figure 5. Data flow : Shunt Current Capture and Framing

B. **FFT**: FFT is performed on the A/D conversion value of the motor drive current framed to detect the feature value.
   The MCU performs the FFT operation using CMSIS DSP. The frequency spectrum resulting from the FFT operation is converted into dBFS. This sample software defines 0 dB = 4095 LSB Full Scale. Next, the peak value of the frequency spectrum (excluding the DC component) and the previous and successive 10 samples (A/D conversion values) are selected to extract the frequency spectrum feature points.
C. **Data extraction**: Extract data in the vicinity where the feature is detected.
   The extracted feature points are input to the trained DNN, and the probability of the two classes (normal and anomaly) are output by inference. In this example, the probability of anomality is taken as the degree of anomaly.

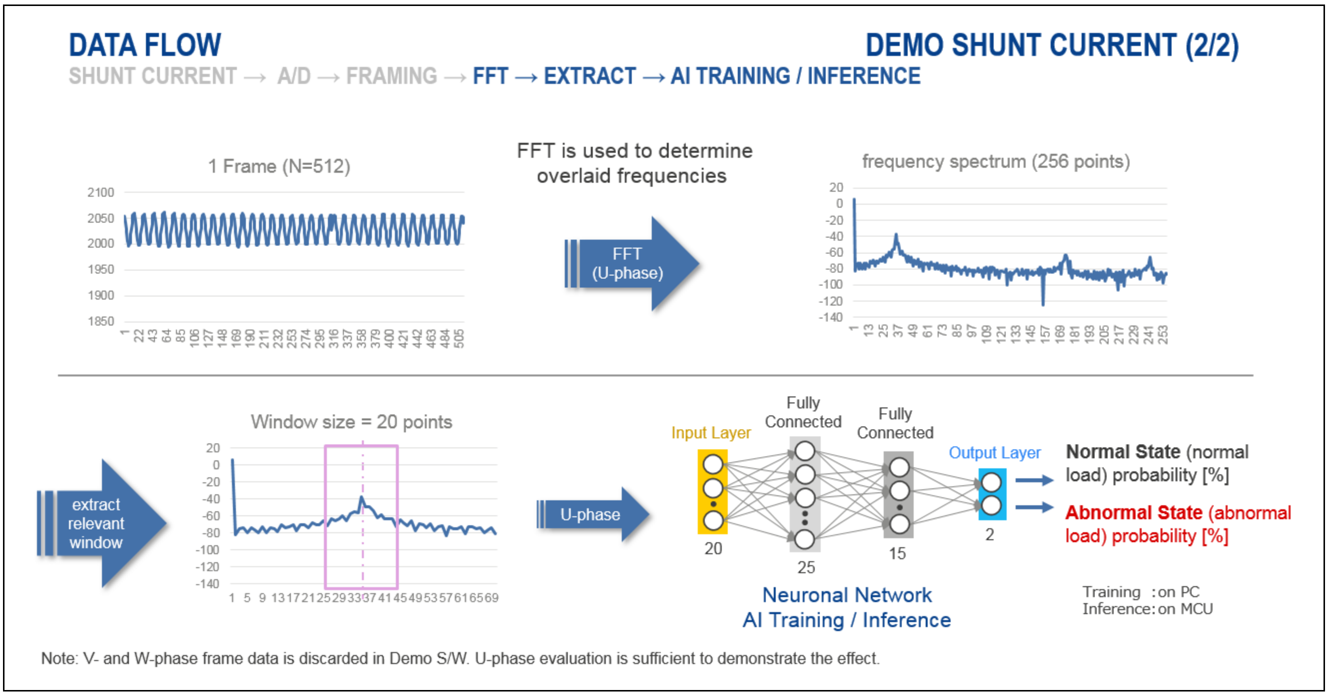

Figure 6. FFT, Data Extraction and AI Inference

3.2 AI Model Development Flow

The e-AI Implementation Flow comprises data acquisition, AI model creation and integration of the resulting Inference Model by Renesas' IDE e[2] studio enhanced by the Renesas plug-in **e-AI Translator**.

Figure 7 shows the flowchart of AI model development. This section describes the development sequence based on this flowchart.

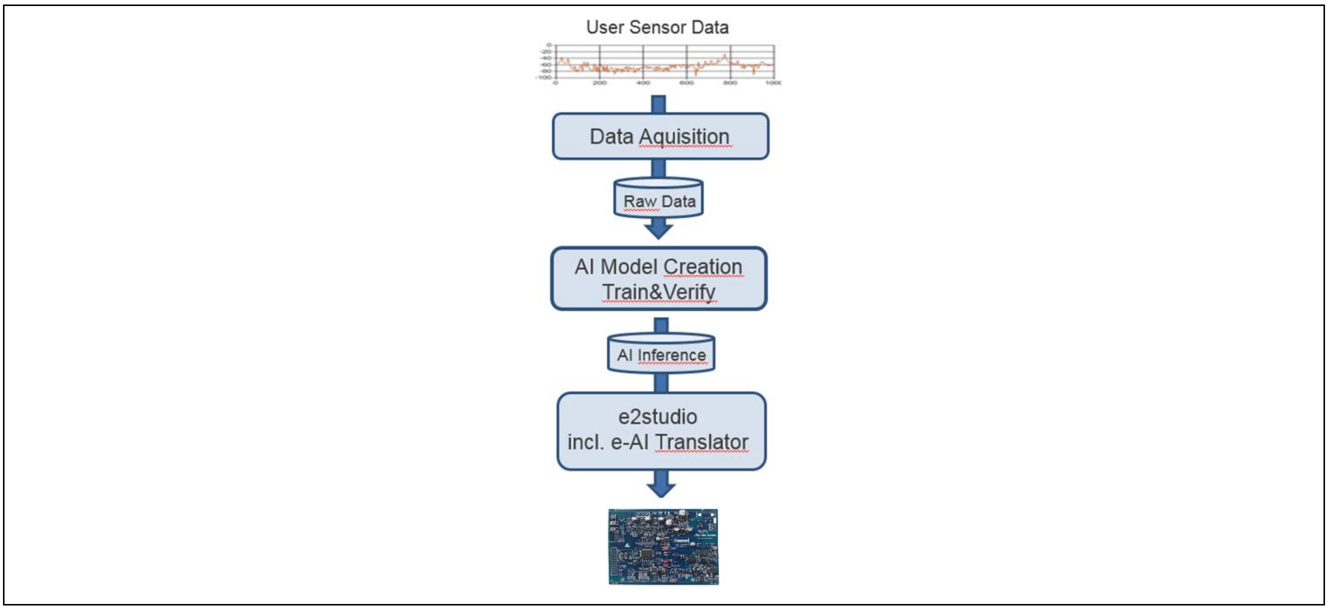

Figure 7. e-AI Implementation Flow

3.2.1 Input Data Collection

This section describes the sequence for collecting data using the Data Collection Tool.

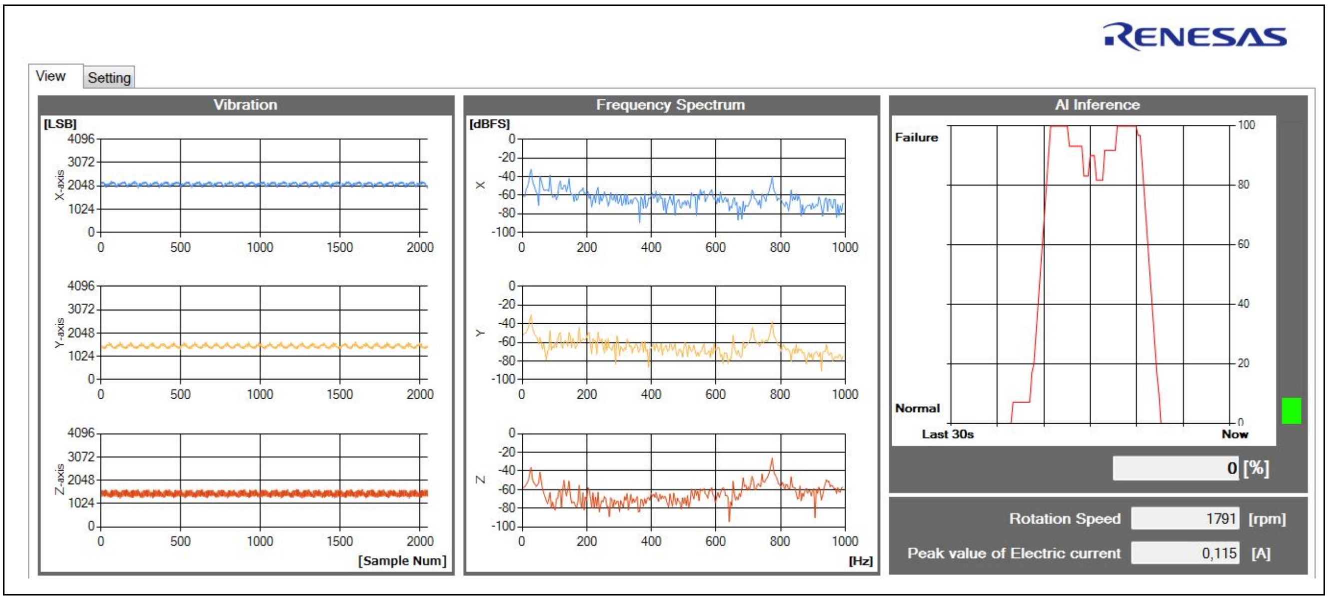

Figure 8. DataCollectionTool GUI

The Renesas DataCollectionTool can be used to acquire input raw data. It is a Windows® stand-alone tool which requires a USB connection to the board/MCU to enable access to the desired input data. The input data can be saved as `.csv` file. The saved `.csv` file is then used as input for TensorFlow to train a Neural Network. Two types of data are collected: data for training and data for testing. Testing data is used by the Training Tool when testing the AI model.

In addition, DataCollectionTool can be used to verify the trained inference model in real application thus, to allow a quick check of the inference model as implemented in MCU code.

3.2.1.1 Data Collection Sequence

Power on the YRotate-IT kit using external power supply. Connect the PC and hardware using an USB cable (J19 connector) and execute the `DataCollectionTool_for_RX.exe` file. For instructions on connecting the hardware, see Reference Document [1]. If you open the `.exe` file before connecting the PC with a USB serial conversion cable, the error shown in Figure 9 will appear on the screen.

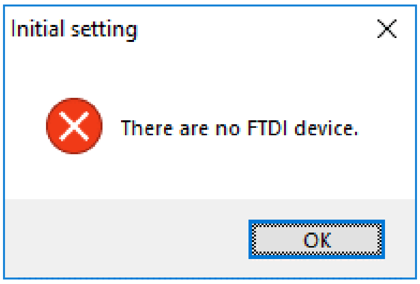

Figure 9. Error Dialog

In the **Settings** tab, change communication baud rate to **256000** and set the path to where you want to store the collected data for Normal / Abnormal `csv` files.

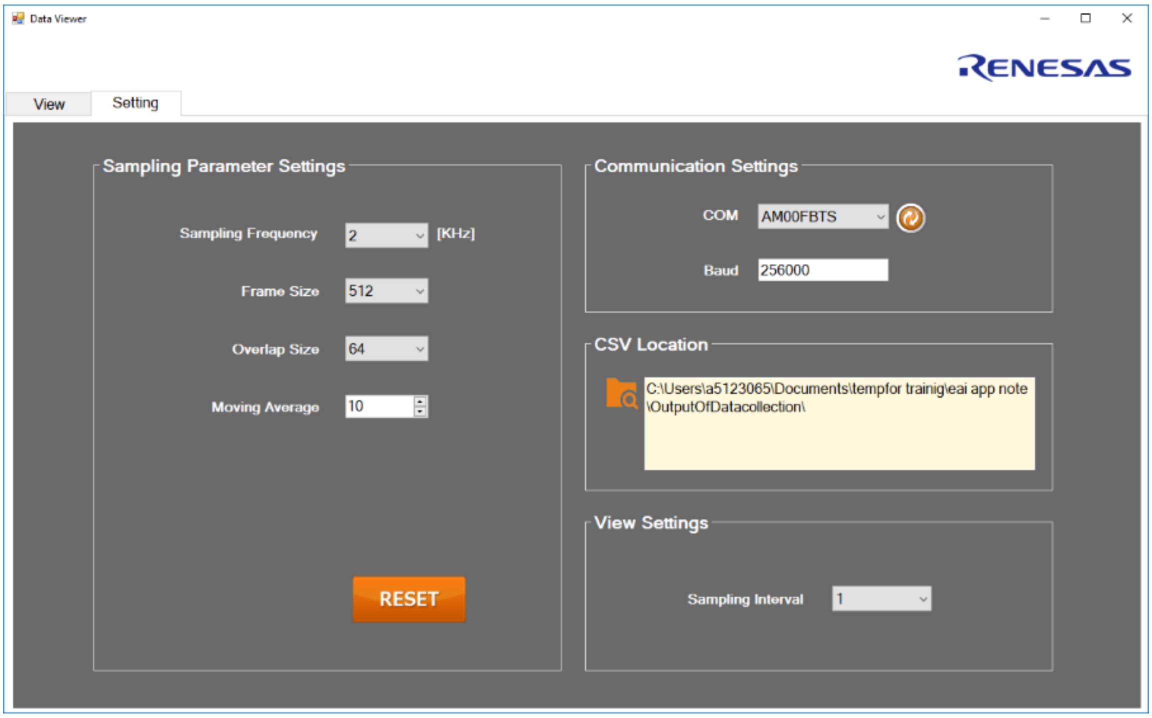

Figure 10. Screen Shot During Data Collection

In the **View** tab, from the dropdown list in the lower right corner of the window, select **Save to CSV (combined)**. The Training Tool supports the csv format output in this mode.

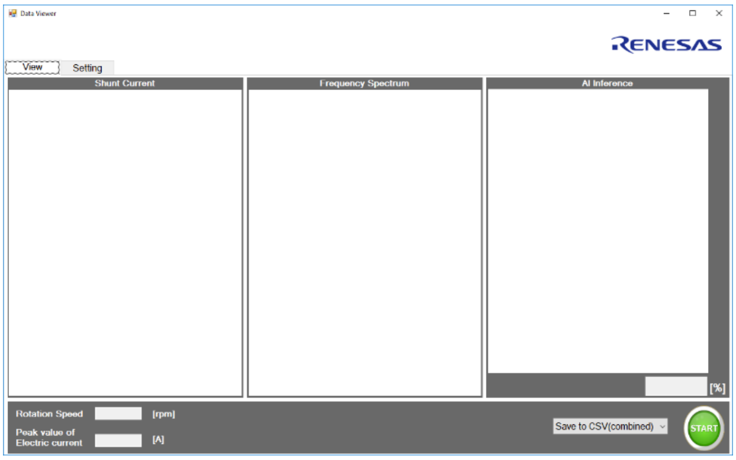

Figure 11. Settings for csv Files

3.2.1.2 Recording the Normal Data without Load (normal condition) with the DataCollectionTool

Check that data output is enough. The longer the recorded time, the output is more accurate, but also the calculation duration is bigger. To get a usable data set of approximately 200-500, record the normal state for at least approximately 20 sec.

- Press **P1** 2 times (2x) to record the Normal state at 1500 rpm
- Click **Start** (DataCollectionTool), after for 20 sec click **Stop** (DataCollectionTool)
- Press **P1** 1x to record the Normal state at 1750 rpm
- Click **Start** (DataCollectionTool), after 20 sec click **Stop** (DataCollectionTool)
- Press **P1** 1x to record the Normal state at 2000 rpm
- Click **Start** (DataCollectionTool), after 20 sec click **Stop** (DataCollectionTool)
- Press **P3** 4x to stop the motor

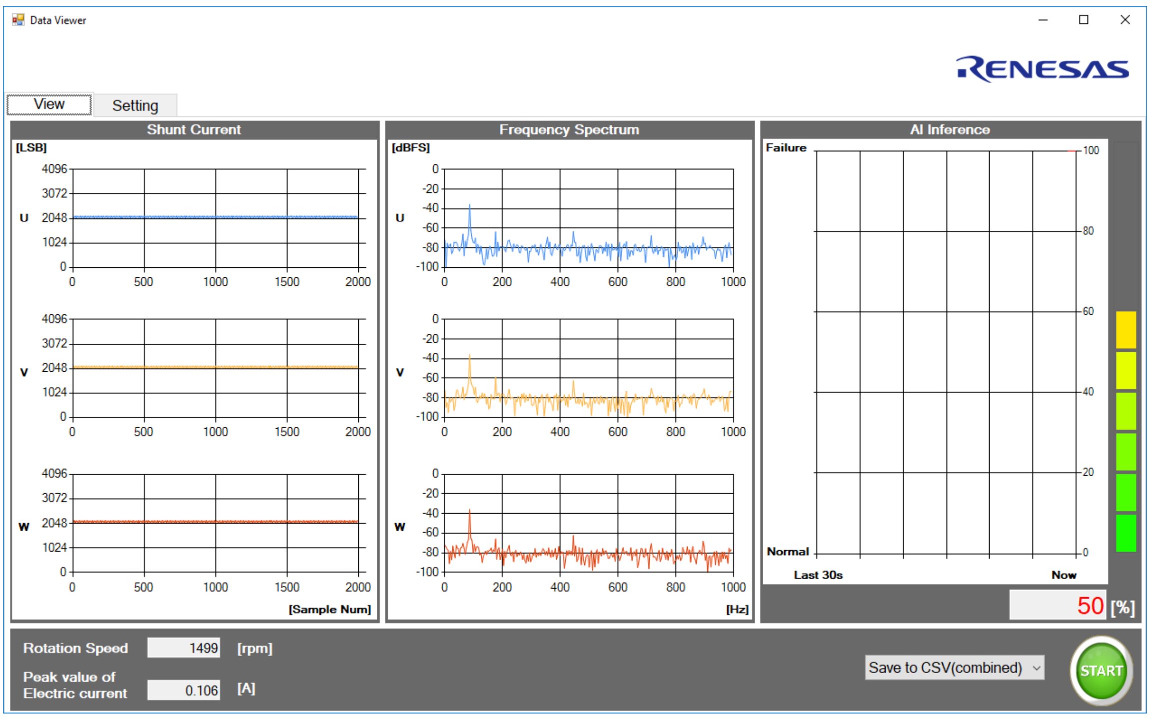

Figure 12. Screen Shot During Data Collection Without Load

The recorded data will be found then in the directory that you have defined in the DataCollectionTool **Setting** view. Rename the files from `SensorData*` to `Normal_No<1-3>`.

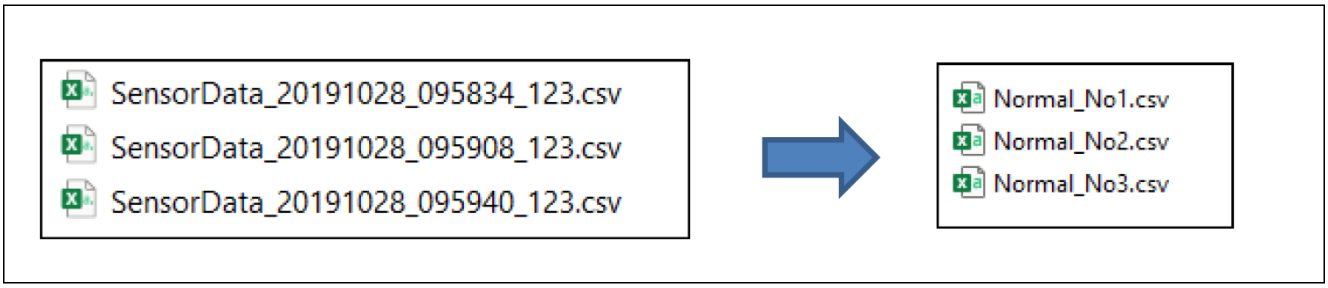

Figure 13. File Renaming for Normal Case

3.2.1.3 Recording the Data with Friction

Simulate a friction of the motor (simply by applying load on the motor shaft by hand) and monitoring the data with DataCollectionTool as follows:

- Press **P1** 2x to record the Abnormal state at 1500 rpm
- Simulate a friction with your fingers on the motor axis
- Click **Start** (DataCollectionTool), after for 20 sec click **Stop** (DataCollectionTool)
- Press **P1** 1x to record the Abnormal state at 1750 rpm
- Simulate a friction with your fingers on the motor axis
- Click **Start** (DataCollectionTool), after 20 sec click **Stop** (DataCollectionTool)
- Press **P1** 1x to record the Abnormal state at 2000 rpm
- Simulate a friction with your fingers on the motor axis
- Click **Start** (DataCollectionTool), after 20 sec click **Stop** (DataCollectionTool)
- Press **P3** 4x to stop the motor

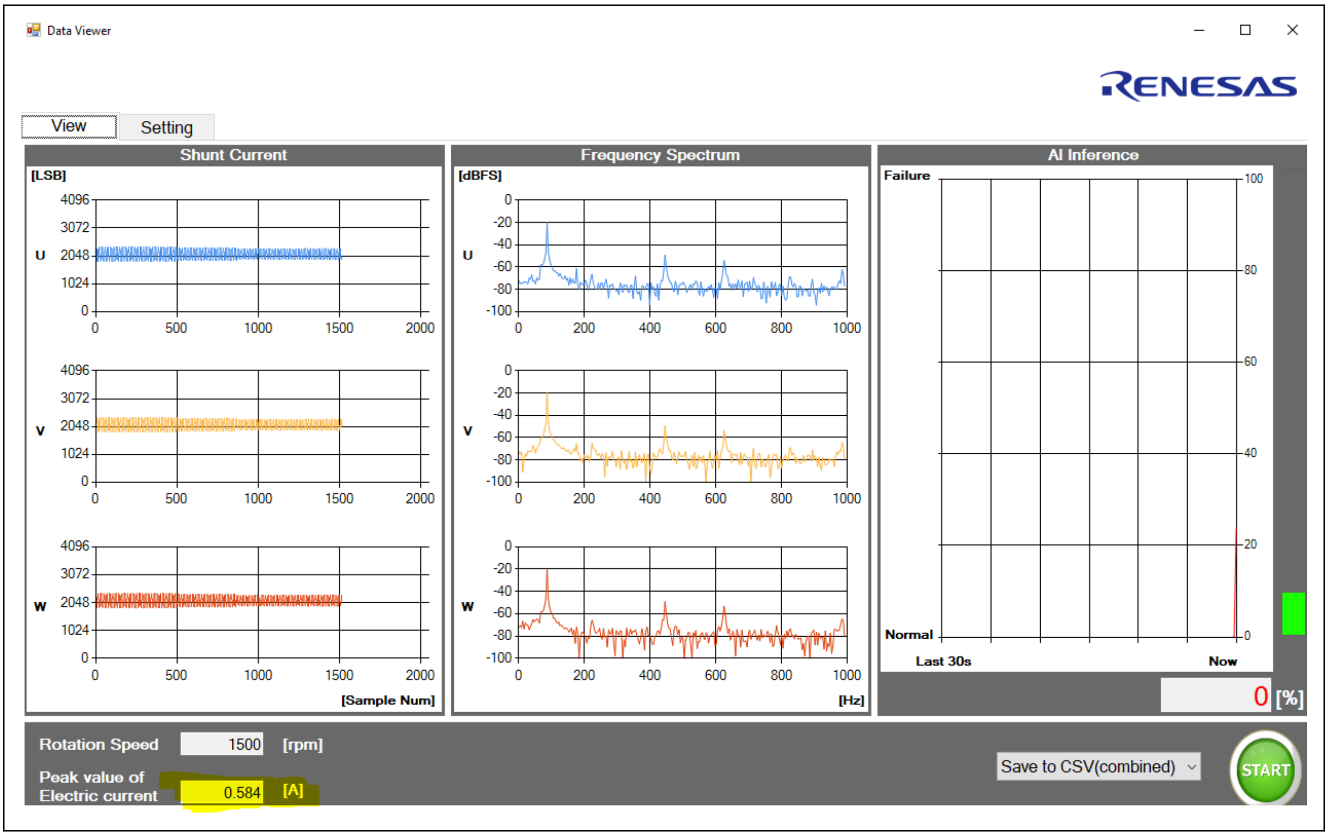

Figure 14. Screen Shot During Data Collection with Load

DataCollectionTool will generate 1 `csv` file for each speed, needed to train and optimize the NN. Rename the files from `SensorData*` to `Abnormal_No<1-3>`.

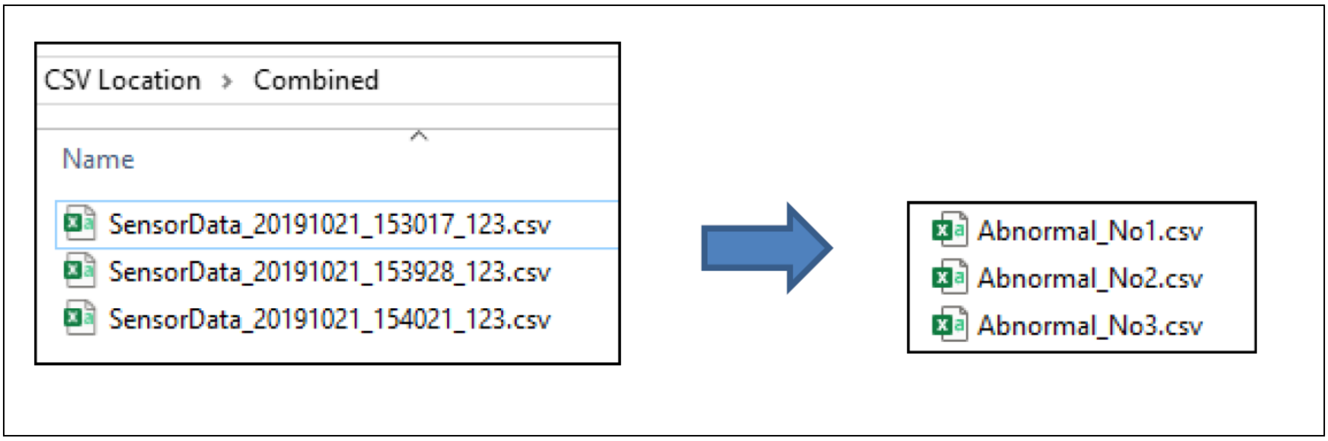

Figure 15. File Renaming for Abnormal Case

3.2.2 Training the Neural Network

This section describes the sequences for training and testing the AI model using the Training Tool (which is provided with the sample project package). To re-train and test the AI model, use the training data and testing data previously collected by the Data Collection Tool.

3.2.2.1 AI Model Training Sequence

- Prepare three folders ahead of time and name them as follows: `Training data`, `Testing data`, `AI_Model`. Store the collected training data and testing data in their respective folders. The `AI_Model` folder is for storing the output AI model.
- Go to the Training Tool directory and launch `python e-AI_Training_Tool.py` as shown as follows.

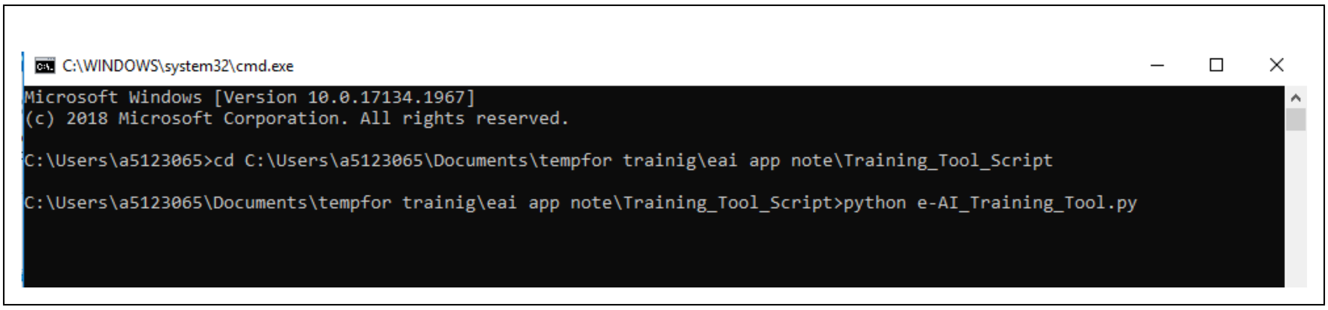

Figure 16. Launch the Training Tool from `cmd` prompt

- Make sure the program is set to **Training** mode.
  — Specify the folder that stores the training data as “Training Data Set.”
  — Specify the folder created for the output AI model as the “Output AI Model.”

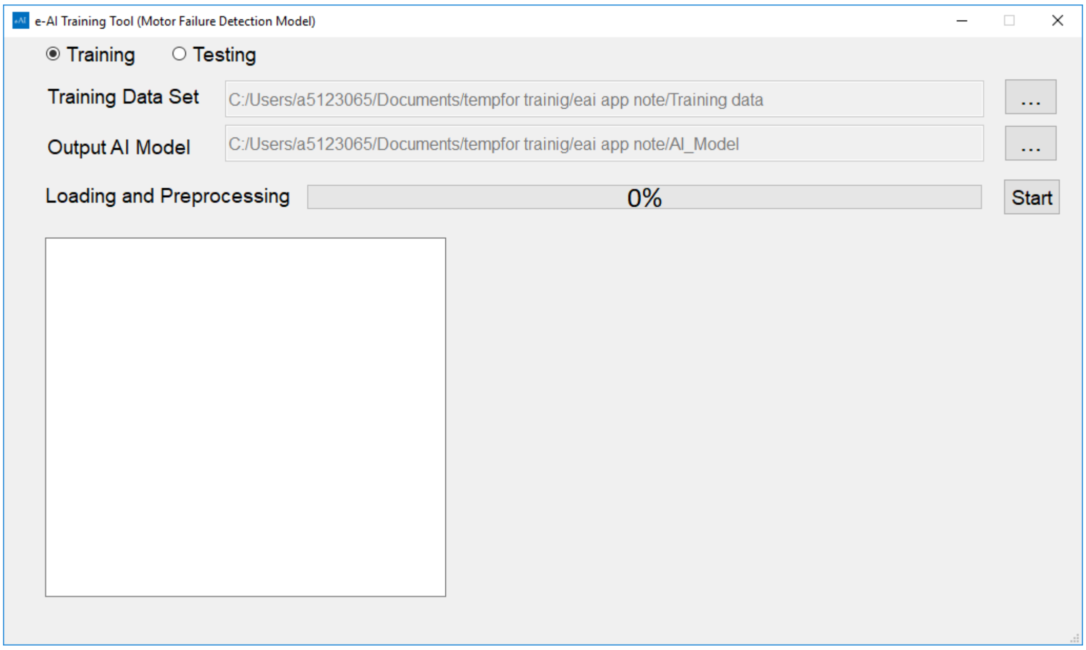

Figure 17. e-AI Training tool window

- Click **Start** to initiate the training. Preprocessing of the training data will start. When the progress bar shows “100%”, preprocessing is complete and “dataframe.csv” is created in the folder where the collected data is stored. When preprocessing finishes, the sequence proceeds to the training process.
- When training finishes, “Training completed” is displayed as shown on the left in Figure 19. If a problem is detected during preprocessing, operations stop and “Training failed” is displayed.

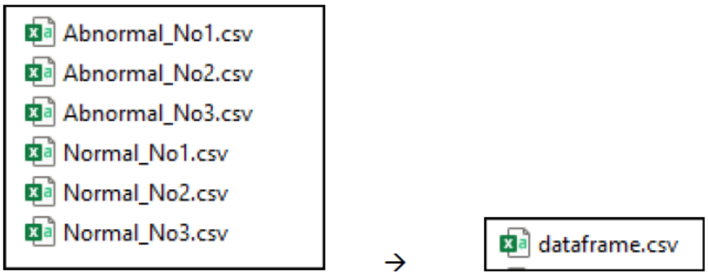

Figure 18. File Generated after Preprocessing

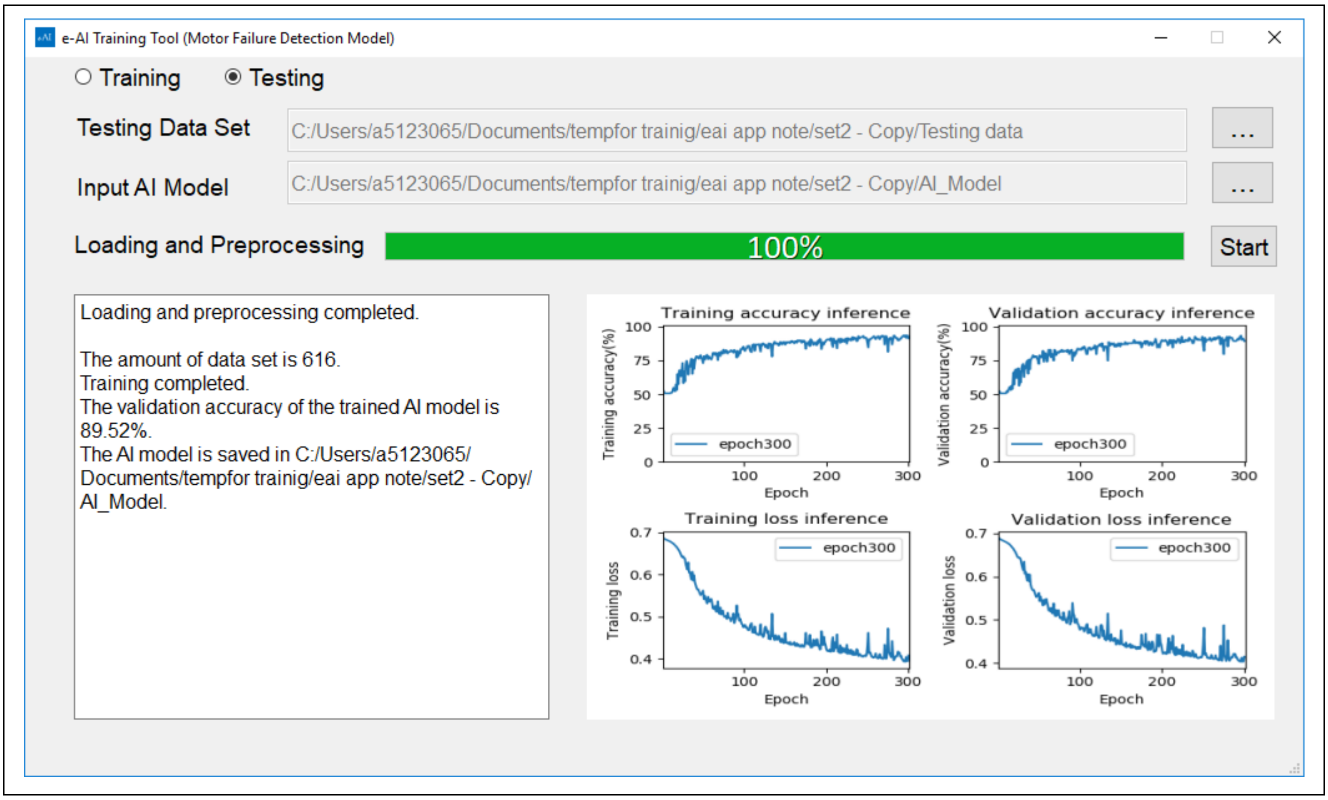

Figure 19. End of Training by the Training Tool

Confirm that the files shown in Figure 20 have been created in the `AI_Model` folder.

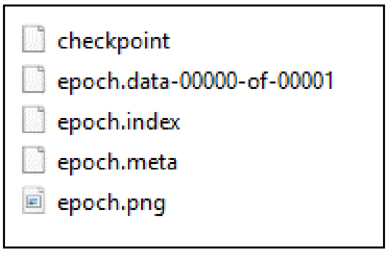

Figure 20. AI Model Output Files

3.2.3 Translation of a Completed, Trained Neural Network into C-code

Since the completed, trained Neural Network is available as a binary data base, it must first be translated into C-code. To support the conversion, the Renesas tool ‘e-AI Translator’ is available. It is provided as an e^2 studio plug-in.

For installation and integration of the e-AI Translator tool please refer to *e-AI Translator V1.6.0 User Manual* [2]. Further information about the tool can be accessed on the Renesas website: e-AI Development Environment & Downloads | Renesas

Input for the Translator is a Neural Network trained by TensorFlow or Caffe.
Output is an inference model in C-Code including a data array containing the weights and bias values calculated during the neural network training process. The C-code output files can easily be implemented into the normal user code by normal e^2 studio functionality.

After the tool has been installed and all the dependencies are installed correctly, the Translator is accessible via the e^2 studio toolbar.

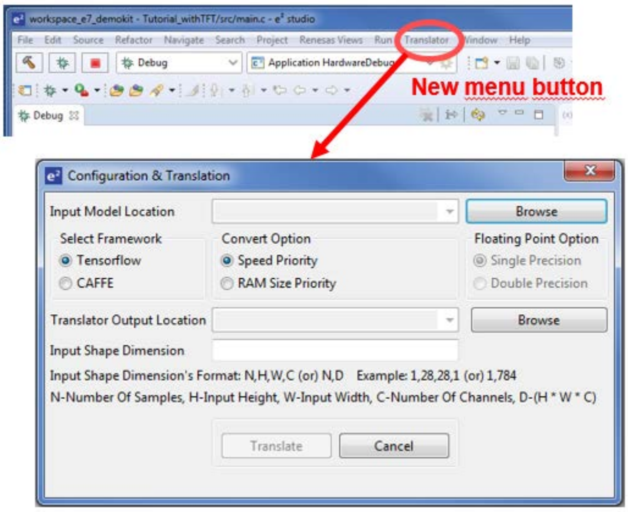

Figure 21. Menu of Renesas' e-AI Translator

3.2.3.1 Translate Python to C-Code

Enter the Input model location. This corresponds to the folder location in which the Neural network model was outputted in the last step. Select a folder for Translator Output location in which the translated C files will be generated. Select the other options as shown below:

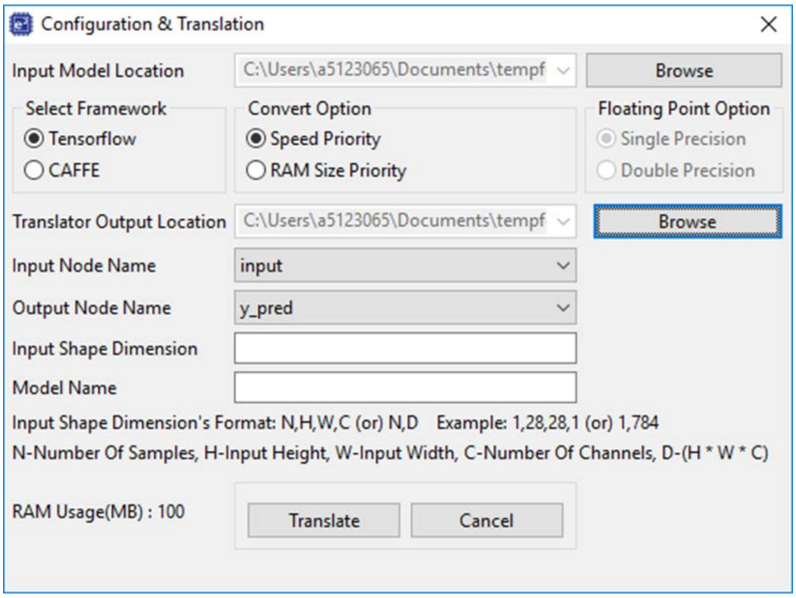

Figure 22. Tool Setup Configuration

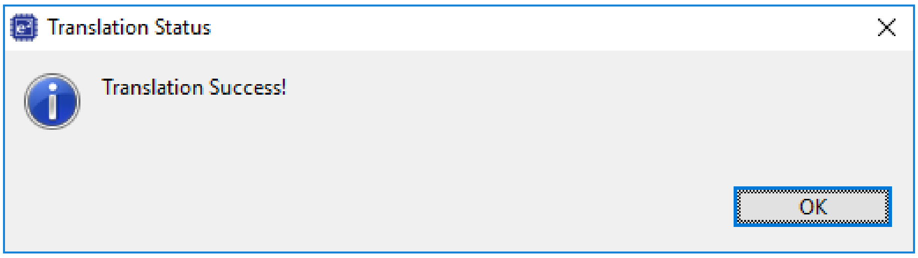

Figure 23. Translation Successful Message

The output of the translation process comprises various files as shown in Table 2.

Table 2. Translated NN C-Code

File name	Description
dnn_compute.c	Inference execution function of the converted neural network
network.c	Neural network function library
layer_shapes.h	Variable/size definitions used in C-Code
weights.h	Weight- and bias values of the trained neural network
network_description.txt	Info: overall structure of the NN, type and size of the single layer
checker_log_output.txt	Info: estimated ROM/RAM size

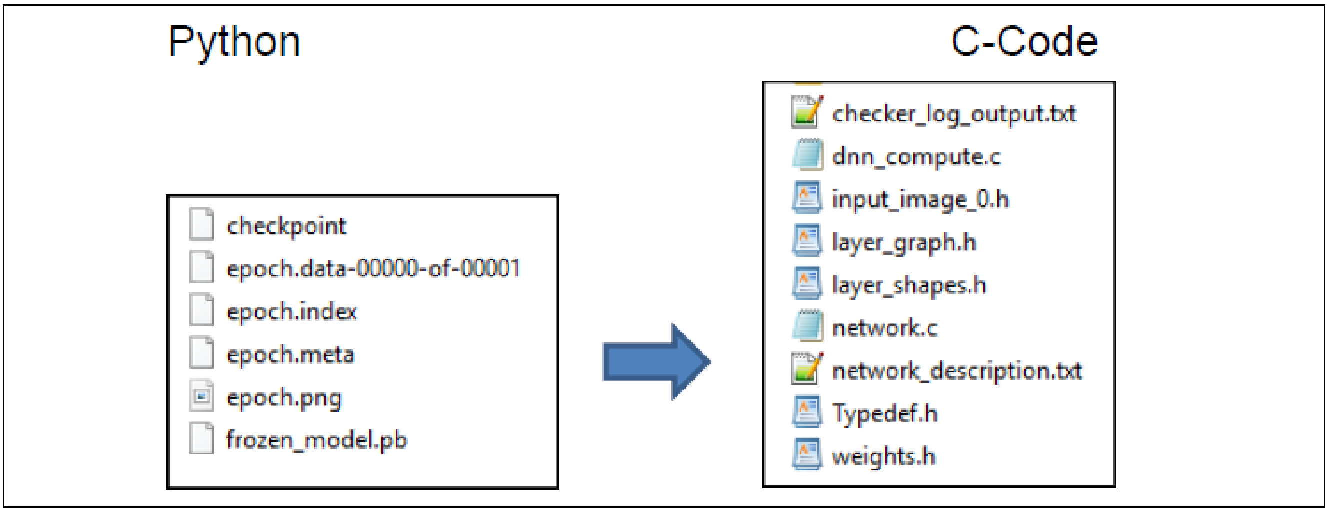

Figure 24. Translation TensorFlow AI Neural Network Model to C-Code AI Neural Network Model

As an example of the resulting C-code, the content of main function `dnn_compute.c` is given by the following.

```
TPrecision* dnn_compute(TPrecision* input_0)
{

  innerproduct(input_0,add_0_weights,add_0_biases,dnn_buffer1,layer_shapes.add_0_shape);
  relu(dnn_buffer1,dnn_buffer1,layer_shapes.Relu_0_shape);

  innerproduct(dnn_buffer1,add_1_0_weights,add_1_0_biases,dnn_buffer2,layer_shapes.add_1_0_shape);
  relu(dnn_buffer2,dnn_buffer1,layer_shapes.Relu_1_0_shape);

  innerproduct(dnn_buffer1,add_2_0_weights,add_2_0_biases,dnn_buffer2,layer_shapes.add_2_0_shape);
  relu(dnn_buffer2,dnn_buffer1,layer_shapes.y_pred_temp_0_shape);
  softmax(dnn_buffer1,dnn_buffer2,layer_shapes.y_pred_0_shape);
  return(dnn_buffer2);
}
```

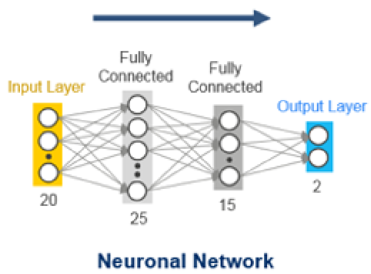

Figure 25. dnn_compute.c: Sequential Processing through the Single Network Stages

In addition to the actual C-Code and the weights array, two text files are generated, which contain detailed information about the overall structure of the neural network as well as estimated ROM/RAM sizes.

checker_log_output.txt - Notepad

Layer Information		Size Information		Speed Information
Layer Output No.	Layer Name	ROM(Byte)	RAM(Byte)	MAC Operations(times)
1	Input	-	64	-
2	Full Connect	1,360	80	320
3	ReLU	-	80	-
4	Full Connect	1,260	60	300
5	ReLU	-	60	-
6	Full Connect	128	8	30
7	ReLU	-	8	-
8	Softmax	-	8	-
9	Output	-	8	-
	TOTAL	2,748	376 212	650

MAC : Multiply and ACcumulation (Number of product-sum operations)
Total RAM size (Upper) : Total size of All layers.
Total RAM size (Lower) : Actual size (Input + dnn_buffer1 + dnn_buffer2 + Output).

Figure 26. checker_log_output.txt: RAM/ROM Sizes and the Number of MAC Operations

network_description.txt - Notepad

Layer Output Index	Function	Layer Input Indexes	Input Size (C x W x H x D)	Parameter	Output size (C x W x H x D)
1	Input	0	16 x 1 x 1 x 1	-	-
2	Full Connect	1	16 (16 x 1 x 1 x 1)	Output node : 20	20
3	ReLU	2	20	-	20
4	Full Connect	3	20	Output node : 15	15
5	ReLU	4	15	-	15
6	Full Connect	5	15	Output node : 2	2
7	ReLU	6	2	-	2
8	Softmax	7	2	-	2
9	Output	8	2 x 1 x 1 x 1	-	-

Figure 27. Network Description

3.2.4 Implement AI Neural Network Model to RA6T1

In order to program the AI Neural Network Model C-Code to the embedded system, copy the C-Code to the appropriate directory, build the project and execute the debugging process to download the C-Code to the RAM.

3.2.4.1 Copy Translated C-Code to e² studio

Since the AI Neural Network Model is converted to C-Code, you can copy the C-Code into the following directory.

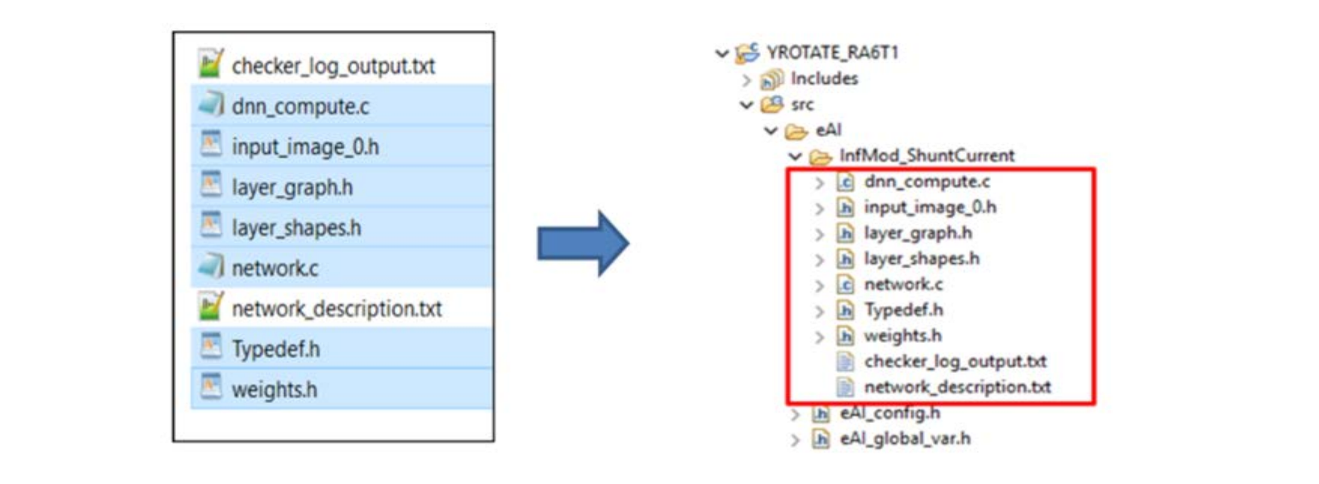

Figure 28. C code Location for AI Interference Model in Project Folder

3.2.4.2 Build the Design in e² studio

After copying the new neural network C model files to the project directory, compile, build and download the new firmware to the CPU. Now we can test the results of the inference model in next step.

3.2.5 Display the Normal and Abnormal Behavior

Check the behavior of the predictive Maintenance with the DataCollectionTool as follows:

- Open `DataCollectionTool.exe` and select View only mode.
- Push **P1** x 2 → 1500 rpm
- Select **Start** button in DataCollectionTool
- The result of the AI inference model will be shown. You can simulate loaded condition (apply load by hand) and verify the results.

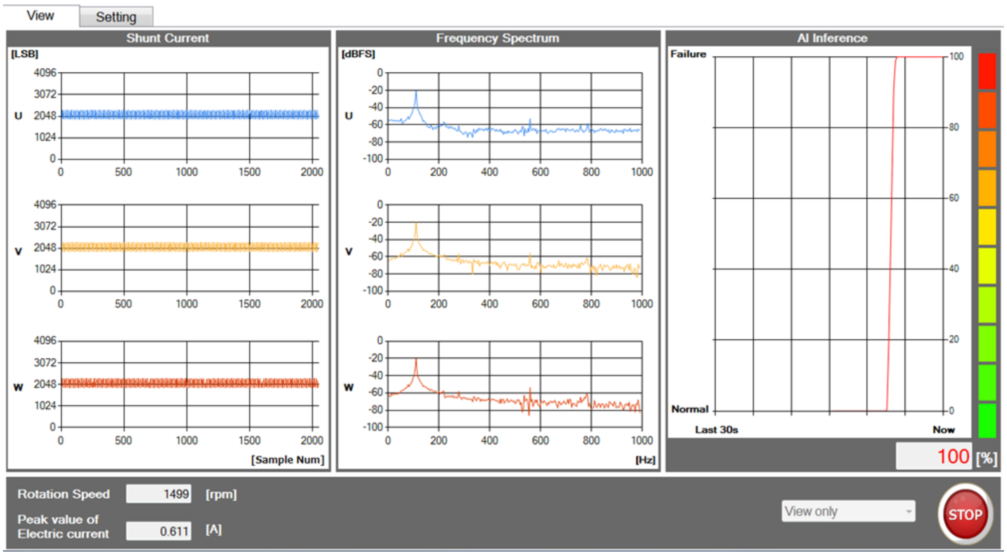

Figure 29. Verifying result of Ai inference model with DataCollectionTool

4. e-AI Software Integration

The basic concept of the RA6T1 e-AI software integration is depicted in Figure 30.

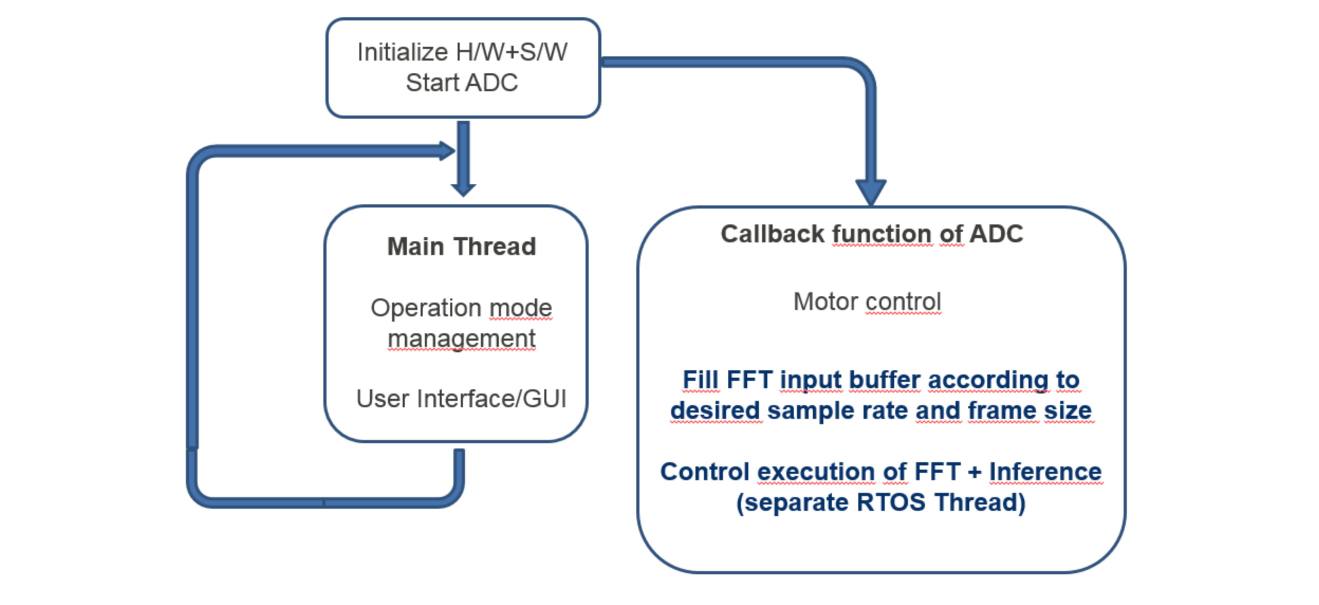

Figure 30. Basic Concept of the RA6T1 e-AI Software Integration

Note: The upper- and left side software parts represent normal motor control procedures. For a detailed description please refer to the *YROTATE-IT-RA6T1 User Manual* [1]. Parts relevant for e-AI are marked bold blue.

The call back function of the shunt current ADC is executed according to the user defined PWM control frequency. It is used to fill up the FFT input buffer and once filled up, to start the execution of the FFT. The FFT itself is executed by a separate RTOS Thread in order not to exceed the available CPU time of the call back function.

In case of the demo, the PWM control frequency is 16 kHz while the sample frequency of the shunt current used for e-AI is 2 kHz. Consequently, only every eighth sample value is taken over and written into the input buffer for FFT.

The start of the FFT in the separate RTOS Thread is controlled by the flag ‘start_fft_flag’. It is set to 1 in case the FFT input buffer is filled up and the preceding FFT is completed and thus a new FFT can be executed. It is set to 0 when the current FFT is completed. This ensures that an FFT is not started while the previous one is still running.

4.1 Filling the FFT Input Ring Buffer and FFT/Inference Model Execution Control

As part of the normal ADC call back function, the ADC input ring buffer is continuously filled. Once filled up, the execution of the FFT and inference model is enabled and will start once the preceding one is completed.

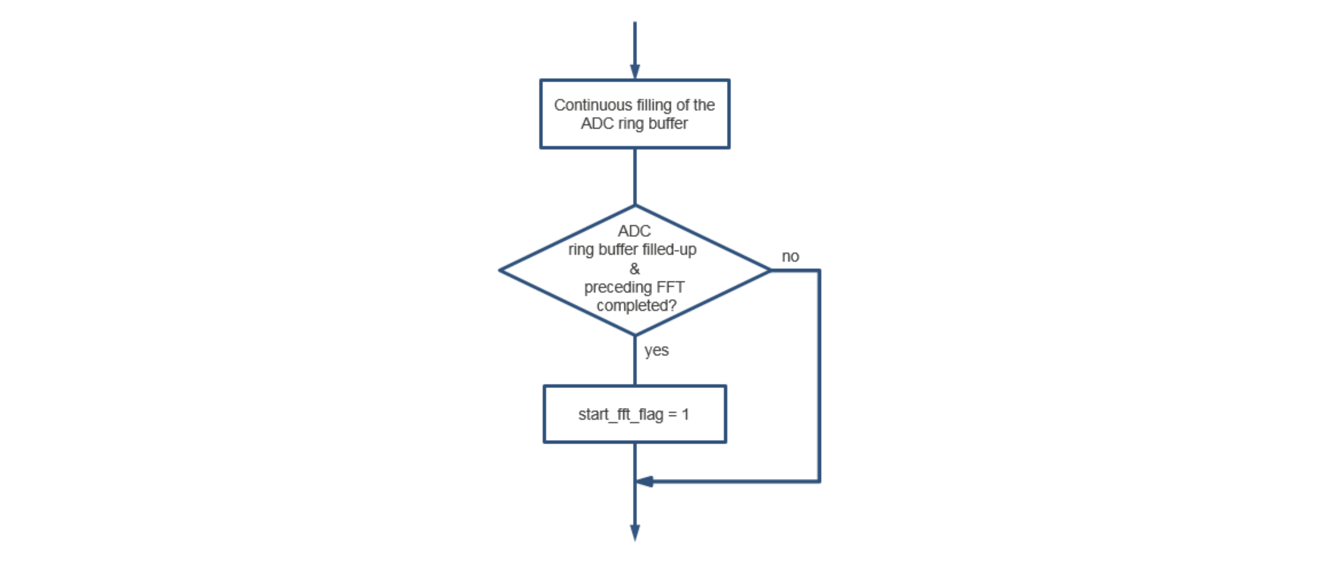

Figure 31. ADC Ring Buffer Flow

4.2 RTOS Thread FFT and Inference Model Execution

To separate the execution of the FFT and the execution of the Inference Model a separate RTOS Thread is defined. It has no additional stacks and interacts via global flags and variables with the ADC call back function filling up the ADC input ring buffer. The execution of the FFT function and the inference model is only started after completion of the previous one.

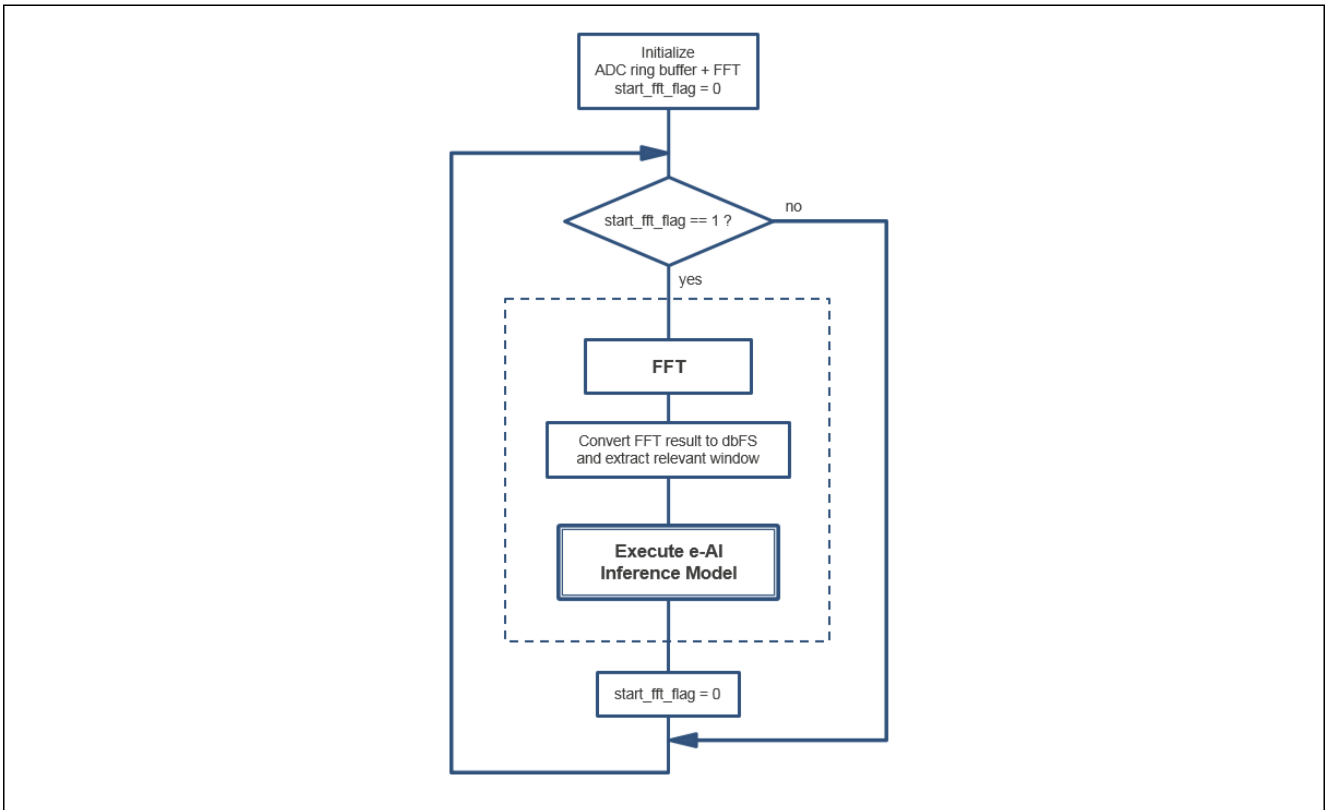

Figure 32. FFT and Inference Model Flow

4.3 Fast Fourier Transformation

The Fast Fourier Transformation is to convert the sensor data from the time to the frequency domain. This is needed to determine the amplitudes of the various frequencies in the original waveform. To perform a Fast Fourier Transformation (FFT) the pre-compiled ARM CMSIS DSP functions are used. The following CMSIS functions are required:

```
arm_rfft_fast_init_f32(..);- initialization function for the FFT
arm_rfft_fast_f32(…);    - FFT processing function
arm_cmplx_mag_f32(…);-complex magnitude function
```

4.4 Inference Model and e-AI Result

The Inference Model as generated by the Renesas e² studio plug-in 'e-AI Translator' is located within the RA6T1 demo kit software as shown in Figure 33.

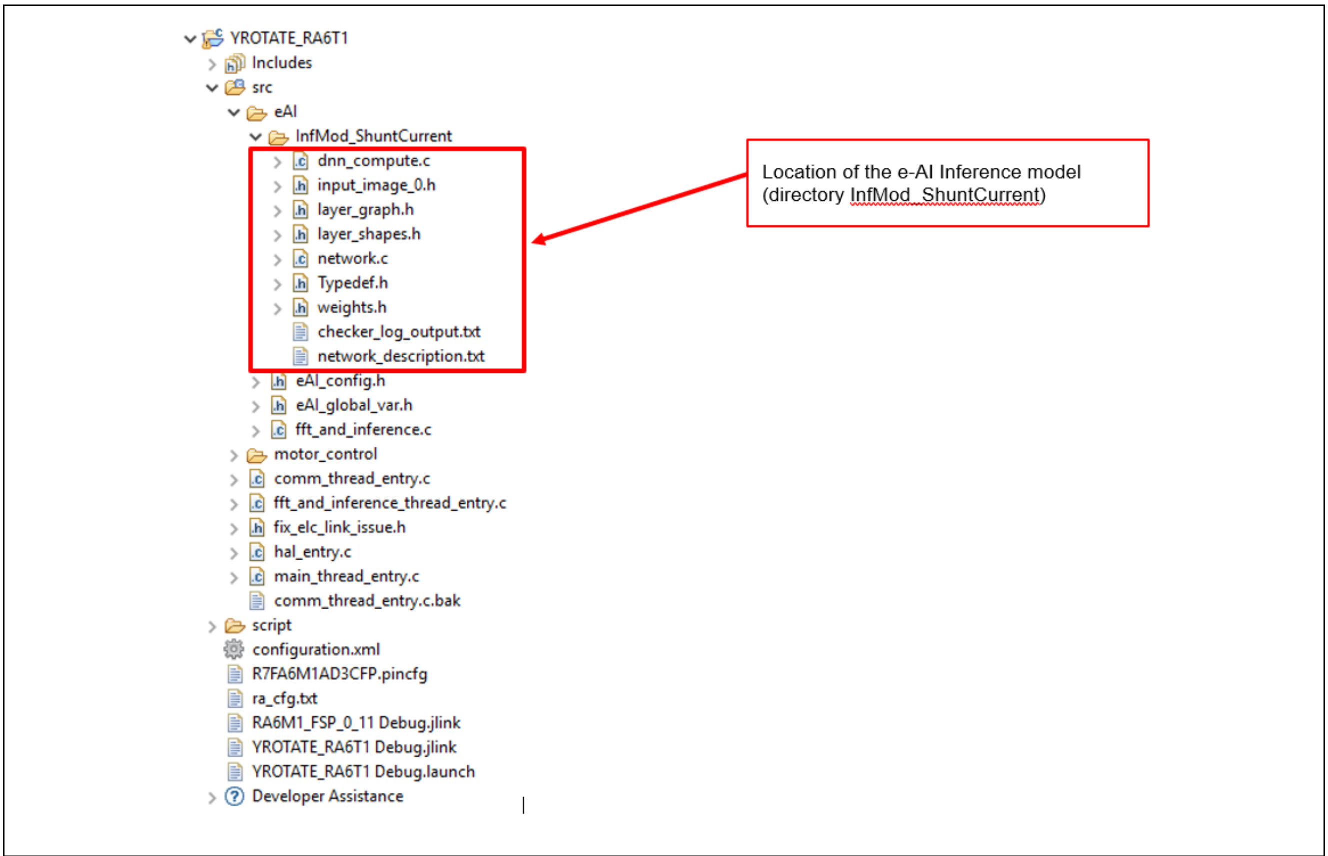

Figure 33. e² studio RA6T1 e-AI YROTATE-IT Project Explorer

The result of the inference model is continuously updated and can be transferred to the 'AI Inference' window of the DataCollectionTool.

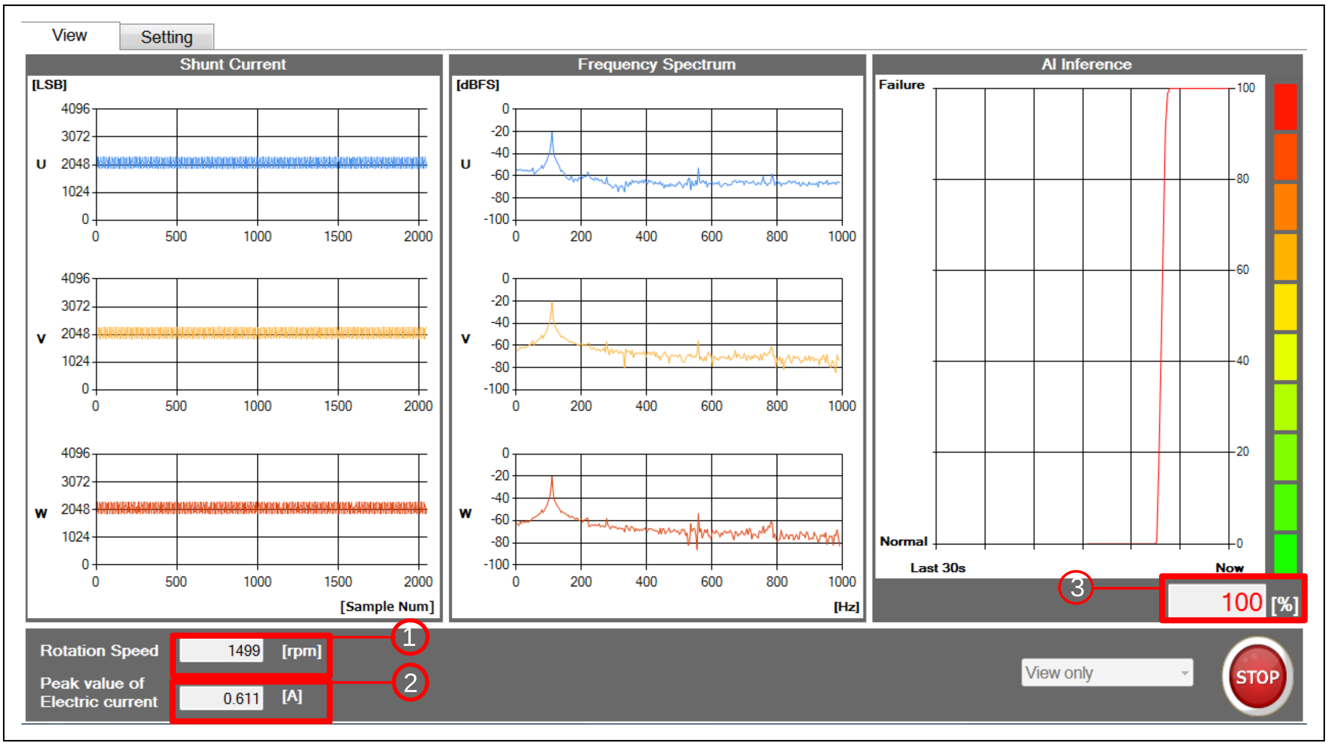

Figure 34. Renesas DataCollectionTool GUI

Legend:

1: Rotation speed [rpm]*
2: Motor current [A]
3: Result of e-AI Inference

*Check chapter 5 for more details.

The corresponding variables that reflect the above mentioned data are listed in Table 3. RA6T1 Board Data Transferred to the DataCollectionTool

Table 3. RA6T1 Board Data Transferred to the DataCollectionTool

Variable	Description
gv_SamplingMonitor.m_SamplingData.m_Ad<n>Value	Shunt Current
gv_SamplingMonitor.m_Sampling256.m_FrequencySpectrum[0][n]	FFT Spectrum
gv_SamplingMonitor. m_RotationSpeed	Rotation Speed [rpm]
gv_SamplingMonitor.m_PeekCurrent	Motor current [A]
gv_SamplingMonitor.m_AiResult	Result of the e-AI inference model

4.5 USB Communication

The RA6T1 YROTATE-IT demo software supports two USB protocols: one for communication with the YROTATE-IT GUI and one to communicate with the DataCollectionTool. To enable the desired communication a #define statement as part of the header file ‘`eAI_global_var.h`’ must be set accordingly:

```
#define USB_COM“YROTATE-IT”   // to set the USB protocol to YROTATE-IT
```

or:

```
#define USB_COM“DCT”         // to set the USB protocol to DataCollectionTool
```

The default setting is “DCT”.

In addition to the USB protocol, the baud rate is different as well: 9600 baud in case of YROTATE-IT and 256,000 baud in case of DataCollectionTool. The baud rate needs to be adapted by e² studio, tab **Project** > **Open FSP Configuration** > **Stack** > **Comm Thread** > **g_uart0 UART Driver on r_sci_uart** > **Properties**, prior to a built. Figure 35 shows the relevant entry. In this example, it is set to 256000 baud, applicable for the DataCollectionTool.

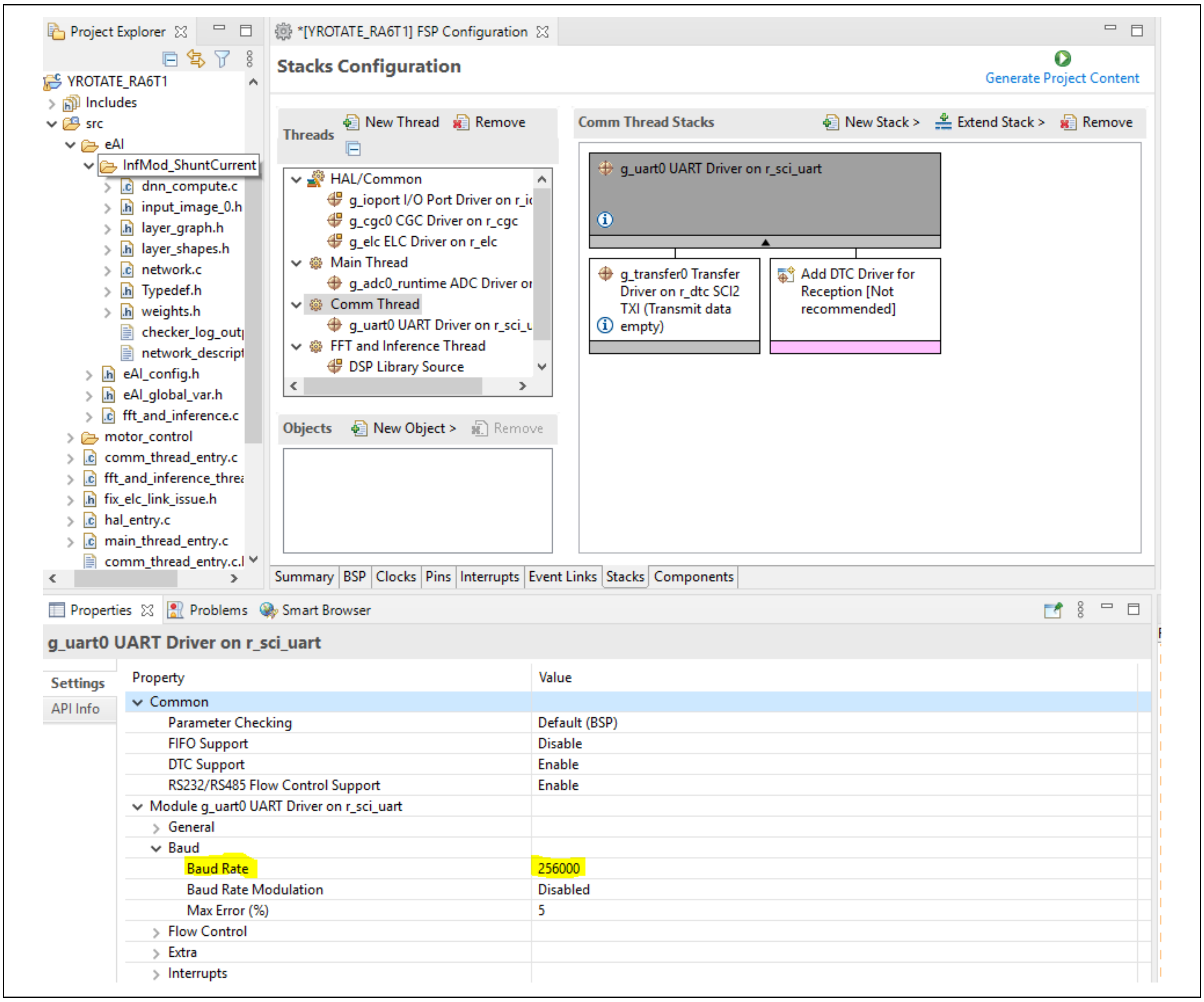

Figure 35. e² studio Threads Configuration Baud Rate Setting

4.6 Additional- or modified C-code Source Files

Table 4. RA6T1 e-AI YROTATE-IT Kit, Additional/modified C-code Source Files

Directory/File name	Description
eAI/InfMod_ShuntCurrent	Directory; contains various C-code source files as automatically generated by Renesas' 'e-AI Translator'. For a more detailed description refer to *e-AI Translator V1.6.0 User's Manual* [2].
eAI/eAI_global_var.h	Variable definitions used for FFT, e-AI Inference and USB protocol
eAI/fft_and_inference.c	Contains FFT pre-processing, FFT- and inference model invocation
fft_and_inference_thread_entry.c	Corresponding RTOS Thread entry function
motor_control/motorcontrol.c	Original RA6T1 YROTATE-IT Kit motor control source code - modified
motor_control/userif.c	Original RA6T1 YROTATE-IT Kit USB data/protocol preparation - modified
comm_thread_entry.c	Original RA6T1 YROTATE-IT Kit USB interface - modified

The location of the additional e-AI related source code files within the project directory is shown by Figure 36.

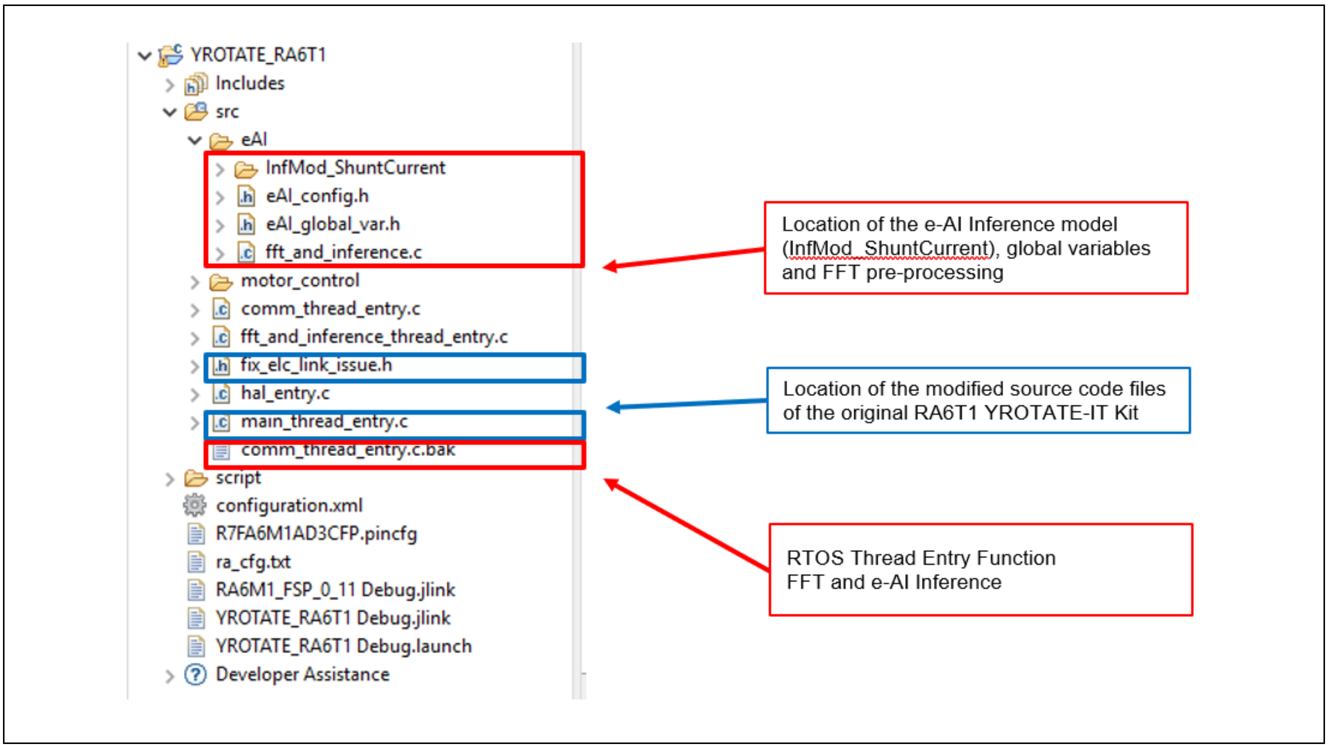

Figure 36. e² studio RA6T1 e-AI YROTATE-IT Project Explorer

4.7 PCB Button Functions Description

The YROTATE-IT RA6T1 board is equipped with the two buttons P1 and P3. When the USB communication is set to "DCT", the two buttons can be used to control the motor rotation speed:

P1: to increase the motor rotation by 250 rpm each push
P3: to decrease the motor rotation by 250 rpm each push

Please note that in the demo software, the adjustable range is limited from -2000 rpm to +2000 rpm with a minimum speed of +/-1250 rpm.

By the two buttons P1/P3, the motor rotation speed can be set independently from an actual USB connection to DataCollectionTool. However, when the USB communication is set to "YROTATE-IT", the function of the P1 and P3 buttons is disabled.

5. Support Tools

5.1 Data Collection Tool

5.1.1 Overview

The Data Collection Tool is software that collects and displays 3 shunt current data and AI inference values from the MCU. The software comes in an EXE format executable file and does not require installation.

5.1.2 Functional Explanations

This software tool has a **View** tab for displaying all information and a **Setting** tab for setting up operations.

5.1.3 View Tab

Figure 37 shows the display layout used in the **View** tab. The numbers in the Figure 37 correspond to the numbered function descriptions below.

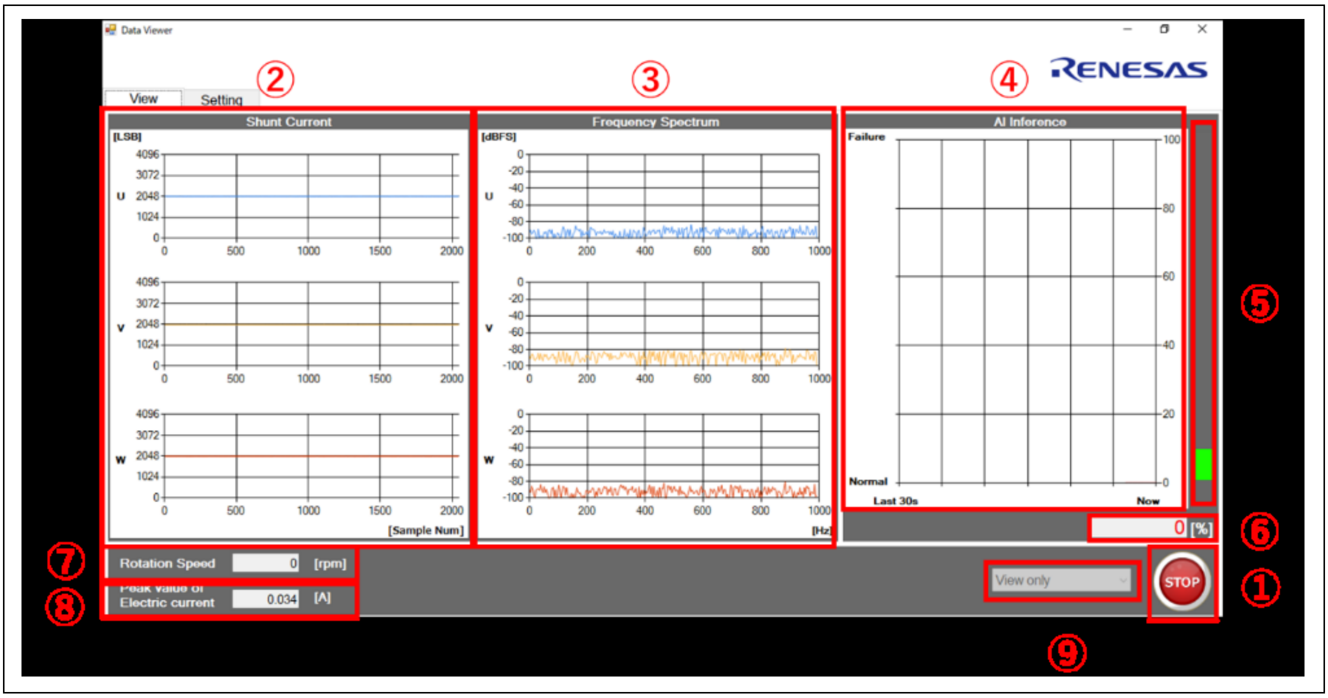

Figure 37. Data Collection Tool Features

1. **Data acquisition START/STOP button**

   The START button is displayed when the GUI software is started up. Each function is described below.

   A. START button is pushed:

   Data Send Request Commands are sent from the PC to RA6T1, and data is sent from RA6T1 to the PC. Received data is displayed in real time.

   B. STOP button is pushed:

   Data Send Stop Command is sent from the PC to RA6T1 and data acquisition ends.

2. **3-shunt current A/D sampling results (magnitude waveform)**

   3-shunt current sampling data is plotted on a graph as U, V and W.

3. **3-shunt current FFT result (frequency characteristics)**

   The 3-shunt current waveform data in (2) above are transformed into the frequency spectrum via FFT, converted to dBFS and plotted on a graph.

4. **Moving average waveform of AI inference result**

   The moving average of the abnormality probability output by AI inference is generated and plotted in a waveform graph.

5. **AI inference result indicator bar**
   Displays the abnormality probability output by AI inference in a stacked bar graph in 10% increments.

6. **AI inference result in percentages**
   Displays the abnormality probability output by AI inference in percentages.

7. **Numerical value of rotation speed**
   Displays the motor rotation speed in numerical value.

8. **Numerical value of peak current value**
   Displays the numerical value of the 3 shunt current's peak current value, which, in this example, is the U phase current's peak value.

9. **Log function selection**
   User selects whether to output log (CSV file) from dropdown list. The CSV file is stored in the "CSV Location" folder immediately under the C drive in the initial settings.

   A. **View only** : Only monitors various data.
   B. **Save to CSV (divided)** : Monitors various data and outputs logs. This setting outputs the sampling waveform and frequency spectrum (dBFS) displayed in (2) and (3) to the file independently for each phase. Data is recorded after a line feed for every FFT frame.
   C. **Save to CSV (combined)** : Monitors various data and outputs logs. This setting outputs the sampling waveforms and frequency spectrums (dBFS) displayed in (2) and (3) together in a single file. Data is added until the acquisition record is completed.

5.1.4 Setting Tab

Figure 38 shows the display layout used in the **Setting** tab. The numbers in the figure correspond to the numbered function descriptions below.

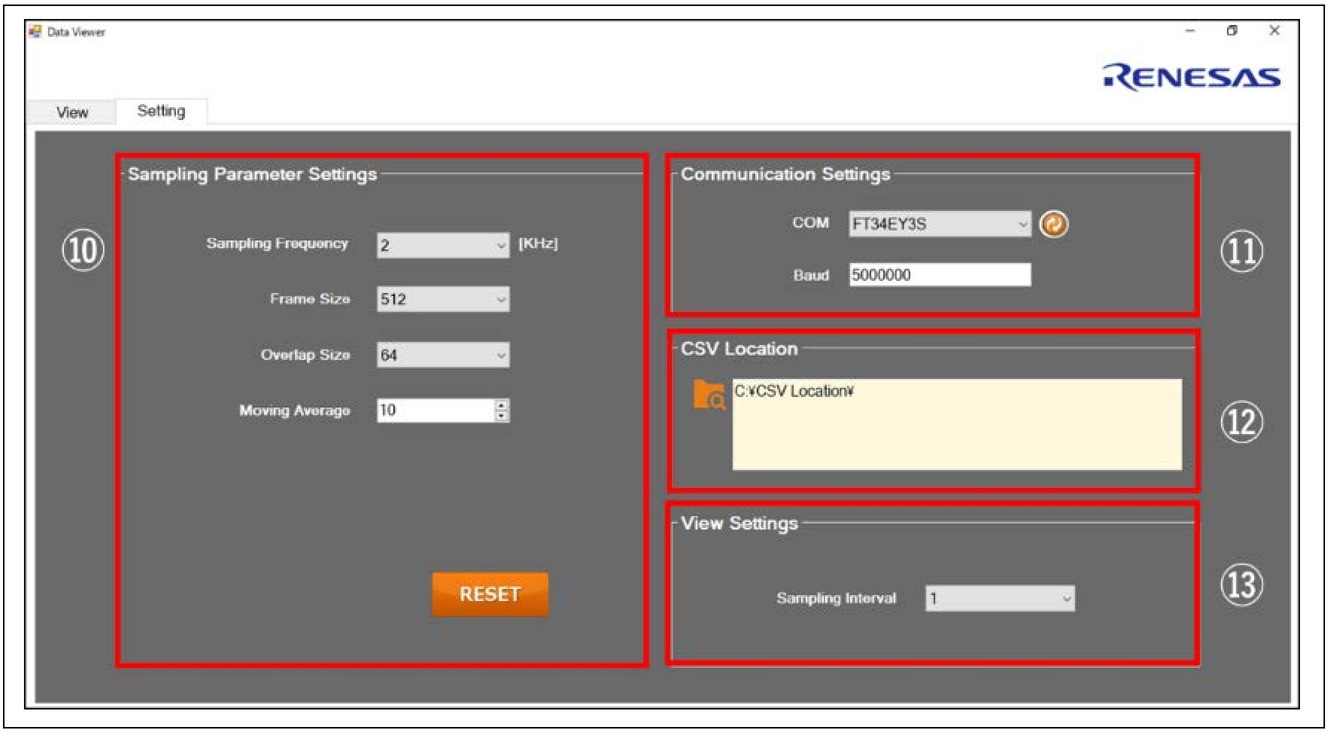

Figure 38. Setting Tab Layout

10. **Sampling parameter setting**
    The trained DNN in this example is optimized to the default setting except for the moving average.
    A. Sampling Frequency : Specifies the sampling frequency (1/2/4/8 kHz, default: 2 kHz).
    B. Frame Size: Specifies the FFT frame size (128/256/512/1024, default: 512).
    C. Overlap Size: Specifies the FFT frame overlap size (16/32/64/128, default: 64).
    D. Moving Average: Specifies the moving average of the graph for the AI inference result (specified range: 1 to 100 times, default: 10).

11. **Communication setting**
    A. COM : Displays the name of the FTDI device connected to the PC.
    B. Baud : Specifies the Baud rate for communications between the MCU and PC (range: 9600 to 5000000, default: 5000000)

12. **CSV storage location setting**
    Specifies the CSV file output location when the **View** tab is set to output logs.

13. **View settings**
    Specifies the update interval of the **view** tab (1/2/4/8/16/32/64, default: 1)

5.2 Training Tool

The Training Tool is the software that trains and tests the AI model. The software comes in an EXE format executable file and does not require installation. It is bundled with trained DNN and can be retrained. The following is an overview of the Training Tool operations.

— AI model training
— AI model testing
— Testing preprocessing

5.2.1 Function Descriptions

This software has two modes: Training mode, which trains the AI model, and Testing mode, which tests the trained AI model.

5.2.2 System Requirements

This training tool needs following software installed in PC.

- Python 3.5.3
- Keras 2.3.1
- NumPy 1.16.3
- Pandas 0.25.3
- TensorFlow 2.1.0

5.2.3 Training Mode

Figure 39 shows the screen layout when in training mode. The numbers in the figure correspond to the numbered function descriptions below.

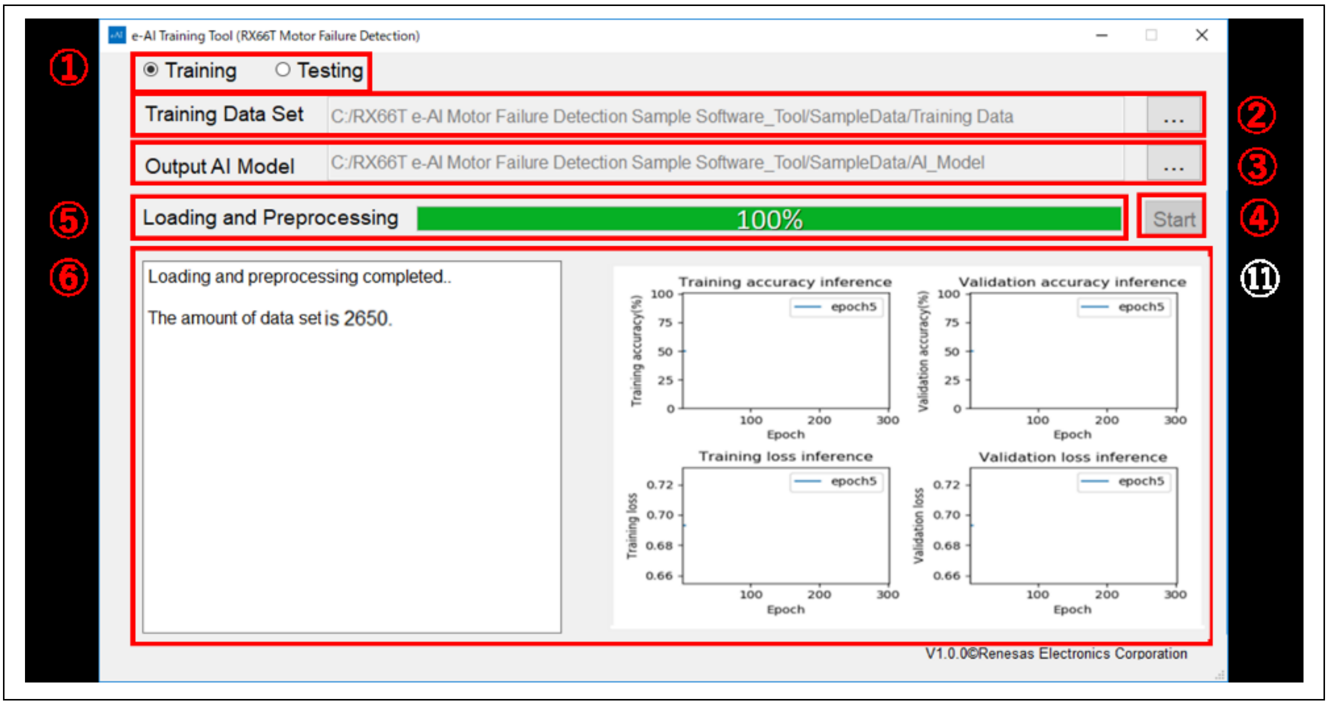

Figure 39. Screen Layout in Training Mode

1. **Mode selection**: Select between Training and Testing mode.
2. **Training Data Set folder setting :** Select the folder that stores the training data.
3. **Output AI Model folder setting**: Select the folder for the Output AI Model.
4. **Start button**: Starts to pre-process the training data and then trains AI model.
5. **Loading and Preprocessing**: Displays the status of the training data preprocessing.
6. **Training Status display:** Displays the status of the training. The accuracy and loss of the AI model are plotted in the graph.

5.2.4 Testing Mode

Figure 40 shows the screen layout when in Testing mode. The numbers in the figure correspond to the numbered function descriptions below.

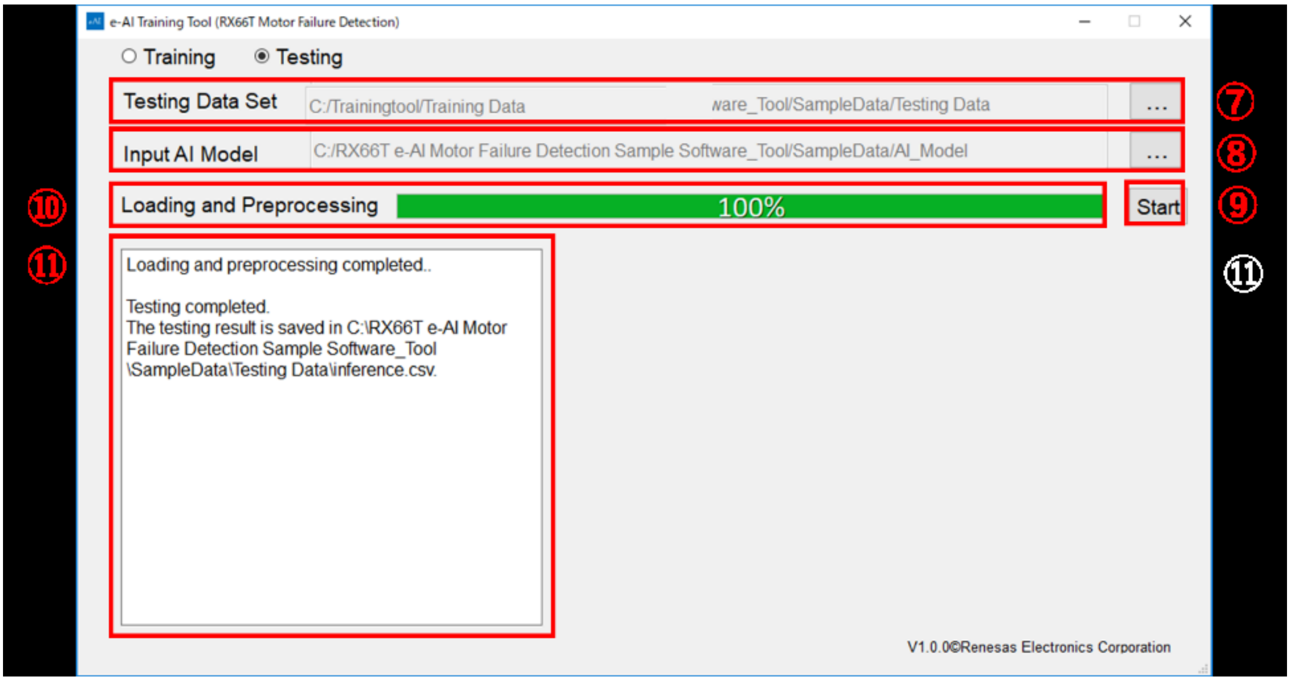

Figure 40. Screen Layout in Training Mode

7. **Testing Data Set folder setting**: Select the folder that stores the testing data.
8. **Set Input AI Model folder setting**: Select the folder that stores the AI model to be tested.
9. **Start button**: Starts to pre-process the testing data and then tests the trained AI model.
10. **Loading and Preprocessing**: Displays the status of the testing data preprocessing.
11. **Testing Status display**: Displays the Testing status.

Website and Support

Visit the following vanity URLs to learn about key elements of the RA family, download components and related documentation, and get support.

RA Product Information	www.renesas.com/ra
RA Product Support Forum	www.renesas.com/ra/forum
RA Flexible Software Package	www.renesas.com/FSP
Renesas Support	www.renesas.com/support

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Mar.5.21	-	Initial release

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on

   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state

   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins

   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals

   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin

   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit: www.renesas.com/contact/.