# RX24U e-AI Motor Failure Detection Sample Software

Application Note

## Introduction

This application note provides a usage example of e-AI (embedded Artificial Intelligence) described through sample software with an additional function that detects motor abnormality in a motor control system using RX24U.

The sample software comes with learned Deep Neural Network (DNN).

## Target Device

RX24U (R5F524UEADFB).

## Contents

## 1. Introduction

Beginning with Endpoint Intelligence, Renesas aims to contribute to the realization of an eco-friendly, smart society that supports safer and healthy living in areas where this cannot be solved simply by using big data in the cloud. With its flexible and scalable embedded artificial intelligence (e-AI) concept, Renesas offers a future-proof, real-time, low power AI processing solution that is unique in the industry and addresses the specific needs for artificial intelligence in embedded devices at the endpoint. Anyone can use AI (Artificial Intelligence) relatively easily by using Caffe developed by UC Berkeley or TensorFlow developed by Google. Although AI's specialty field varies according to the algorithm used, DNN (Deep Neural Network), a multi layered network, is used for embedded AI. Through an algorithm that learns input information that is labelled normal / abnormal DNN has dramatically improved the estimation of a failure condition. DNN has a large difference in the amount of computation required for learning and inference execution, and it is a major feature that it can be executed with less computing power in the inference phase. Focusing on the asymmetry of this computing power and for its main use for inference execution in embedded devices, we named this AI "e-AI" (embedded-AI). The e-AI development environment solves these problems and makes it possible to implement learned DNN results onto an MCU/MPU in conformance with an e² studio C/C ++ project.
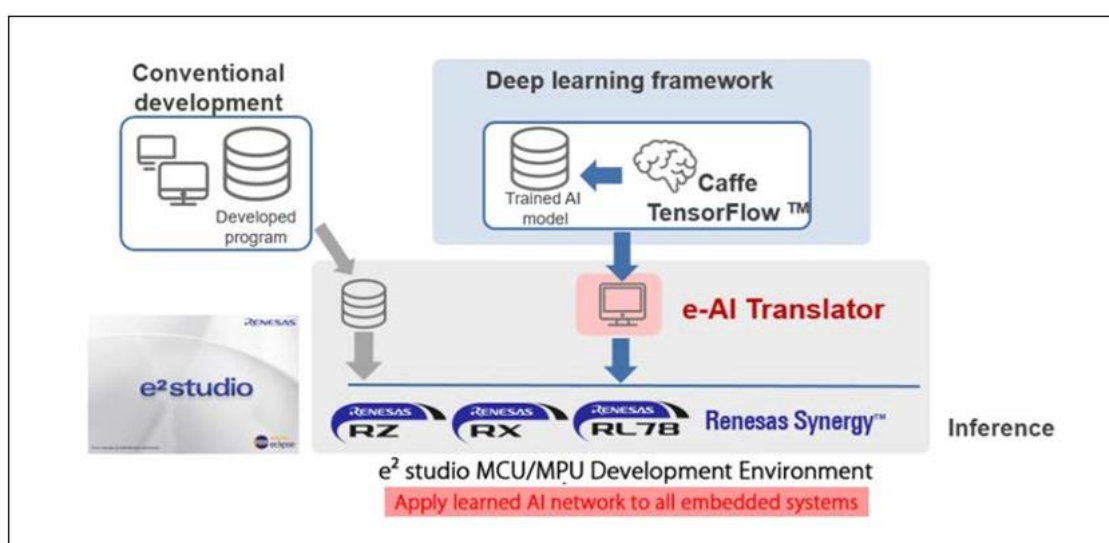


**Figure 1**

## 2. Overview

Figure 2 shows the system block diagram of the e-AI Motor Abnormality Detection Sample Software. This example is an e-AI-based motor status (abnormal value) display system in which learned DNN is added to the brushless DC motor control MCU software. The AI inference results are displayed via computer software.



**Figure 2. System Block Diagram**

The AI inference process of this example performs the following operations.

1. Collect A/D conversion values of the U-phase current and generate an FFT frame.

    2. Pre-processing before learned DNN input data

        a. FFT processing of data frames (frequency spectrum generation)
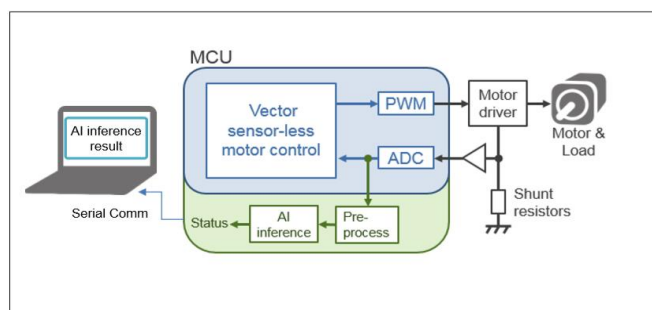
        b. Feature point extraction from frequency spectrum (learned DNN input data generation)

    3. AI inference

The system's brushless DC motor control employs the sensor less vector control method to monitor the U phase shunt current control with the A/D converter. In this system, focusing on the fact that the waveform of the U phase shunt current changes depending on the state of the motor, this U-phase shunt current is used as the input of trained DNN. A/D conversion values are accumulated for a fixed time to obtain waveform data on the time axis.

In input data pre-processing, a frequency spectrum is generated via FFT making it easier for AI to detect feature points of the U-phase shunt current waveform. In the e-AI system with limited storage area, reduction of the DNN network layer is a benefit, allowing peak extraction and use these extracted peaks as input data feature point along with the RPM at which motor is running.



**Figure 3. Data Processing Flow (1/2)**
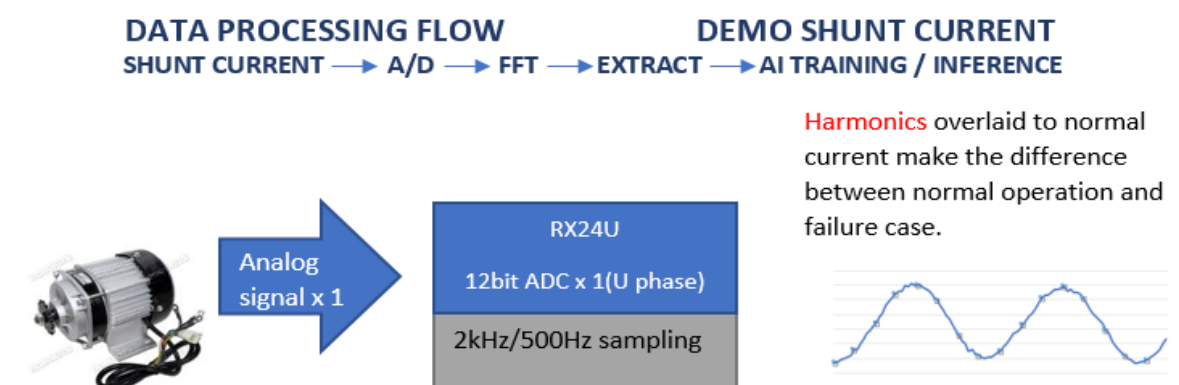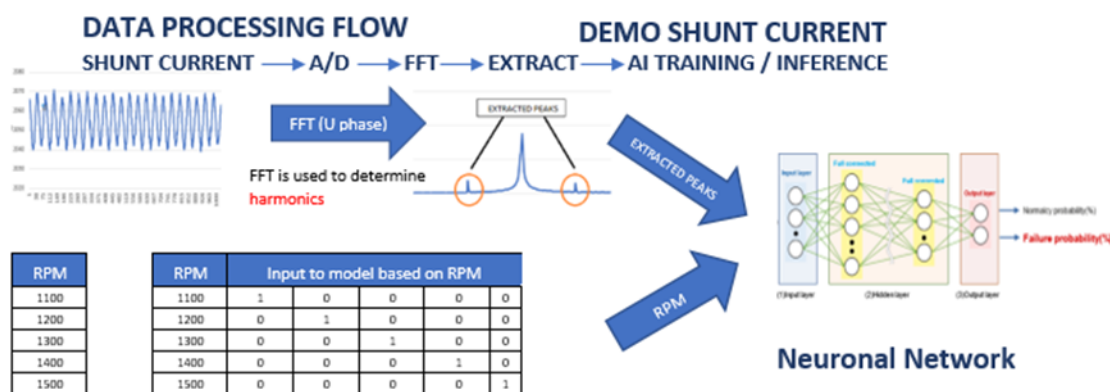


**Figure 4. Data Processing Flow (2/2)**

AI inference results are displayed in the PC software (DataCollectionTool). In addition to the AI inference results, this sample software also displays the U-phase shunt current A/D conversion value and the spectrum waveform. The system's waveform data log function allows the system to accumulate DNN training data while displaying waveforms.
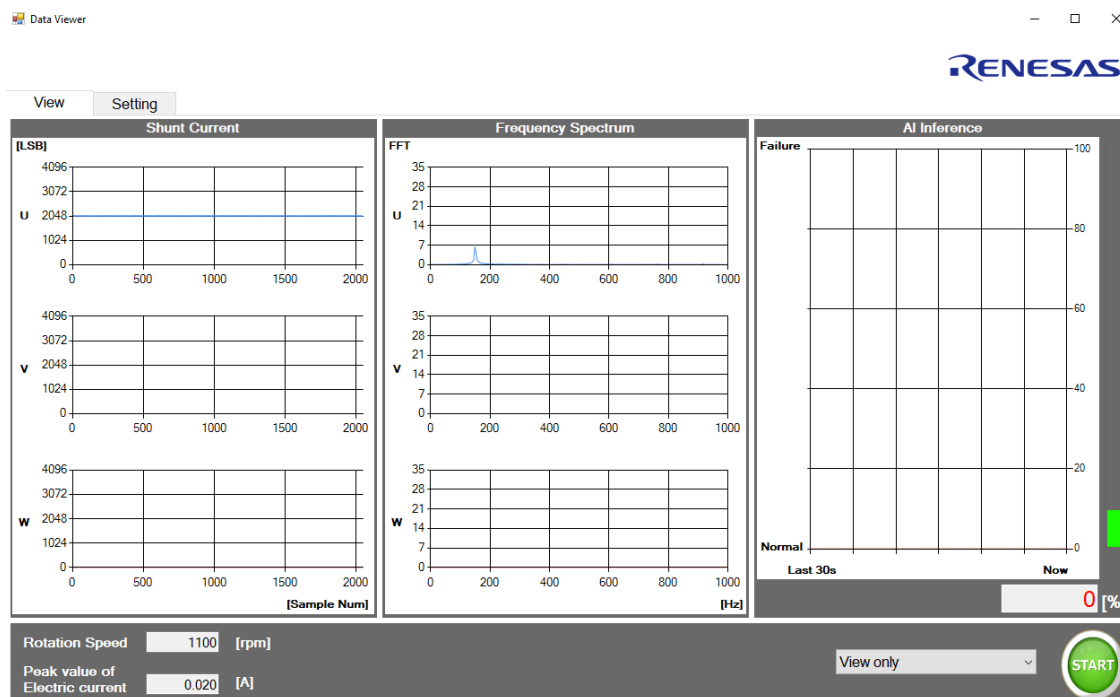
**Figure 5. DataCollectionTool – Appearance**

## 3.  Specification

### 3.1  Operating Conditions

| Item | Description |
|---|---|
| MCU | RX24U |
| Operating Frequency | Main clock oscillator frequency: 1 to 20 MHZ<br>CPU clock (ICLK): 80MHz<br>Peripheral module clock A (PCLKA): 80MHz<br>Peripheral module clock B (PCLKB): 40MHz<br>Peripheral module clock D (PCLKD): 40MHz<br>FlashIF clock (FCLK): 32MHz |
| Operating Voltage | 5.0V |
| Operating Mode | Single-chip mode |
| Endian | Instructions: Little Endian<br>Data: Selectable as little endian or big endian |
| Integrated Development Environment | Renesas Electronics e$^2$studio V7.2.0 |
| C Compiler | Renesas Electronics CC-RX: V3.01.00<br>● Compiler option: -isa=rxv2 |
| Emulator | Renesas Electronics  E1 emulator (2 wire UART interface) The emulator such as E2, E2 lite, etc. also can be used. |
| Middleware | RX24U Data Pre Processing Lib |
| e-AI Development Environment | Renesas Electronics  e-AI Translator V1.0.2 |
| Evaluation Board | Renesas Electronics<br>● 24V Motor Control Evaluation System for RX23T (RTK0EM0006S01212BJ)<br>● RX24U CPU card (RTK0EMX590C02000BJ) |

## 3.2 Operations Overview

### 3.2.1 Workbench Setup of demo

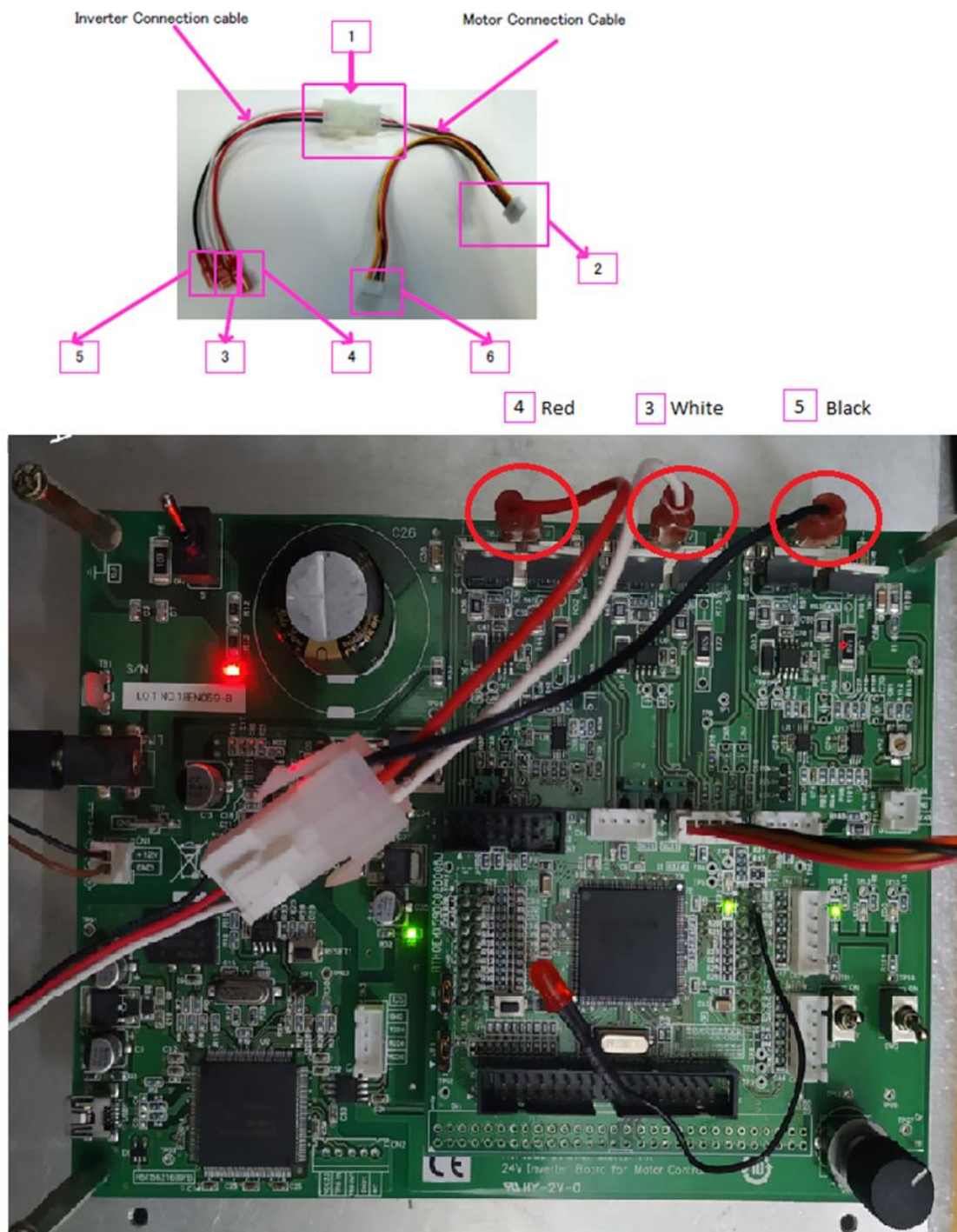- **Figure 6 shows the BLDC motor connection to 24V invertor board**



**Figure 6. Motor Connection to 24V inverter board**

The Inverter cable red probe should go to U, White probe should go to V and Black probe should go to W on inverter board.

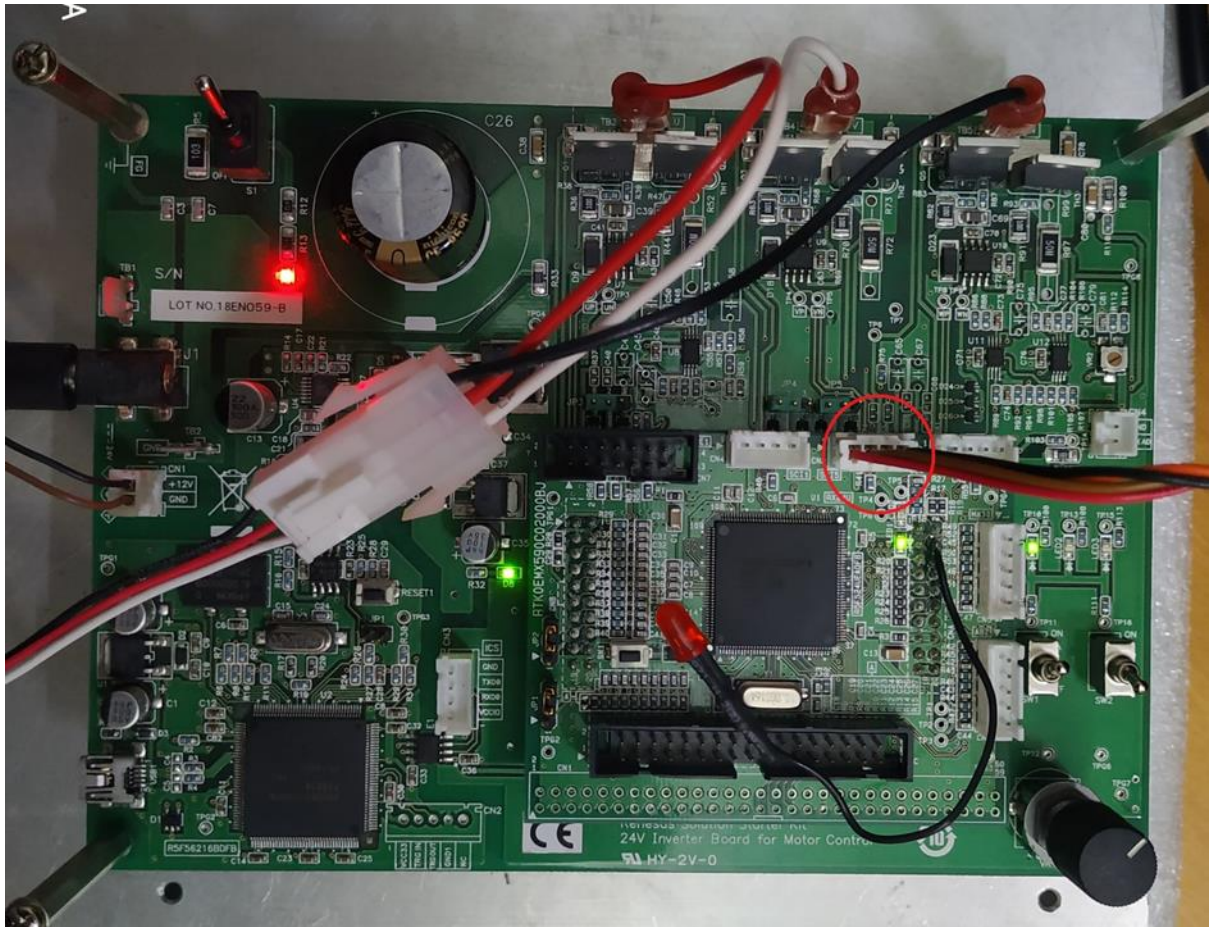- **Figure 7 below shows the SCI connection to RX24U CPU card**



**Figure 7. SCI connection of RX24U**

SC5 of RX24U is used for serial communication with the Data Collection Tool.

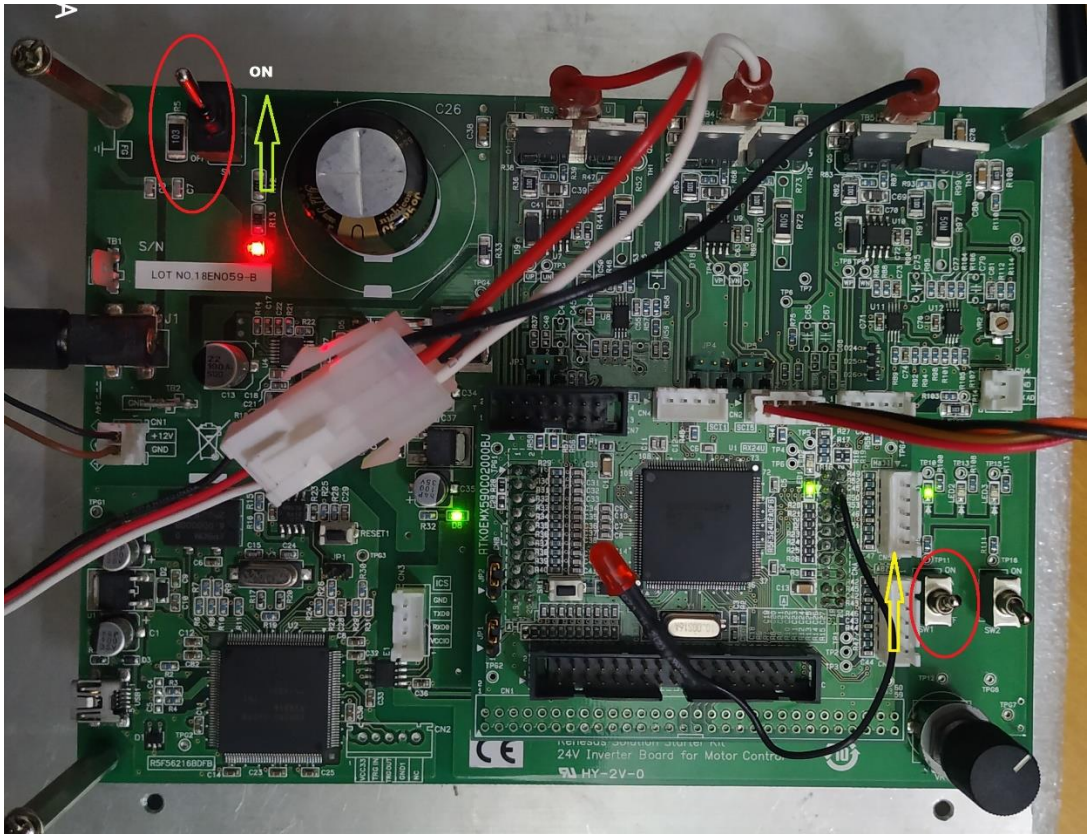- **Figure 8 below shows the switch conditions required to run the motor**



**Figure 8. Switch Conditions**

| State | S1 switch | SW1 switch | LED1 (inverter board) | LED1 (RX24U) | LED2 (inverter board) | LED2(RX24U) |
|-------|-----------|------------|-----------------------|--------------|-----------------------|-------------|
| Working | ON | ON | ON | ON | OFF | OFF |
| Error | ON | OFF | OFF | OFF | ON | ON |

Follow the above table switch conditions and rotate the Knob VR1 to run the motor.

In the Error state, please turn on and turn off the SW2 to remove error. Then Turn ON SW1 and rotate the Knob to run the motor.

### 3.2.2 Data collection Tool settings for demo

- **Figure 9 shows the USB-Serial Cable connect.**

USB-serial converter cable C232HM-EDHSL-0, manufactured by FTDI (Future Technology Devices International), is used for communication between RX24U and the PC.

cable C232HM-EDHSL-0 pin assigned

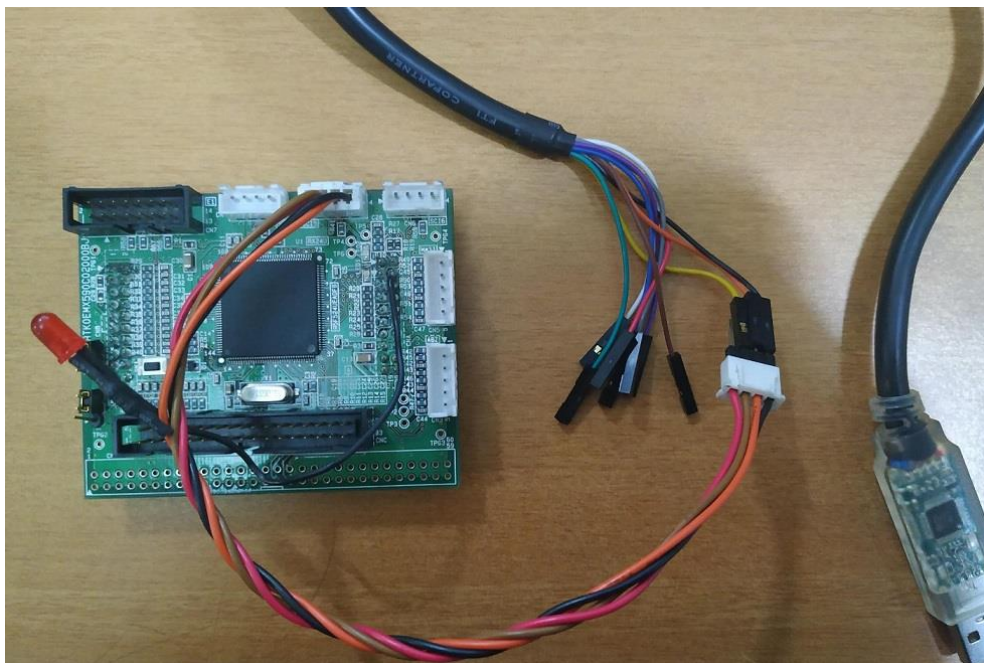| Function | Wire color |
|----------|------------|
| TXD | Orange |
| RXD | Yellow |
| GND | Black |

**Figure 9 USB-Serial cable**



**Figure 10 USB-Serial Connection cable to RX24U**

As SCI5 is used for serial communication in RX24U, please connect cable C232HM-EDHSL-0 as shown in figure 10.

- **Figure 11 shows the parameter setting and communication settings of Data collection Tool for RX24U.**
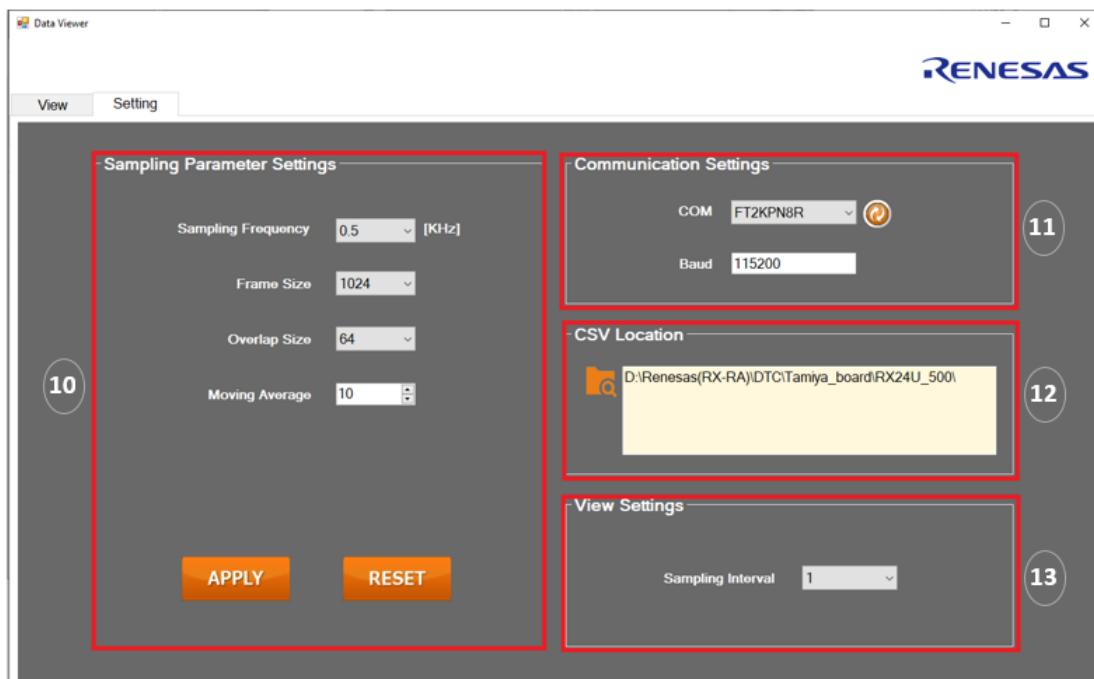


**Figure 11. Data Collection Tool Parameter settings**

Please select Sampling frequency as 0.5kHz, Frame Size as 1024 and Baud rate as 115200.
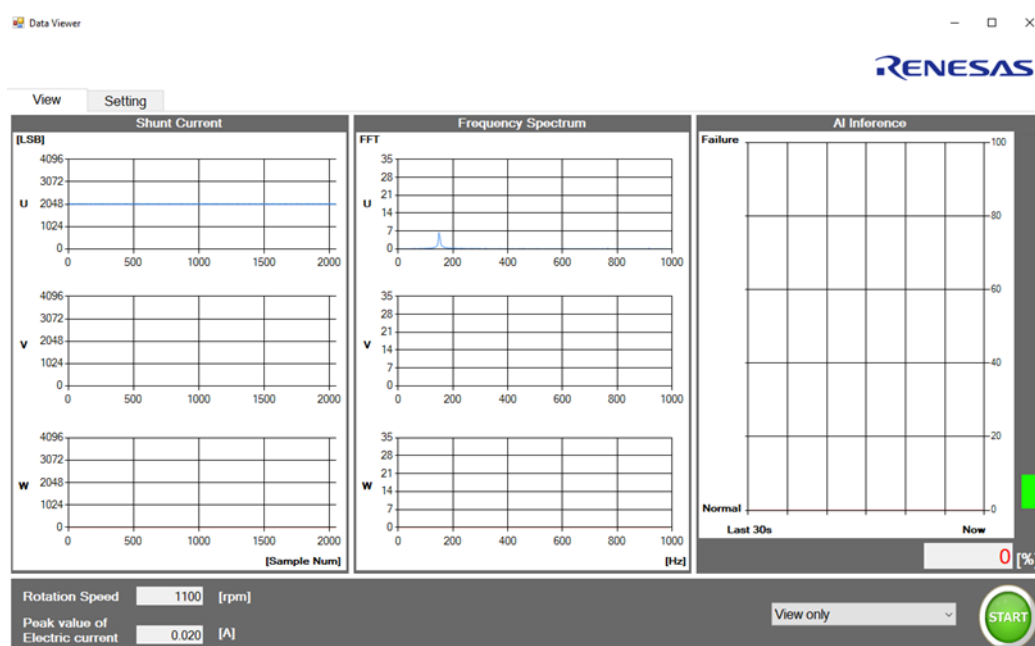
- **Figure 12 shows the results of RX24U demo.**



**Figure 12. Data Collection Tool display**
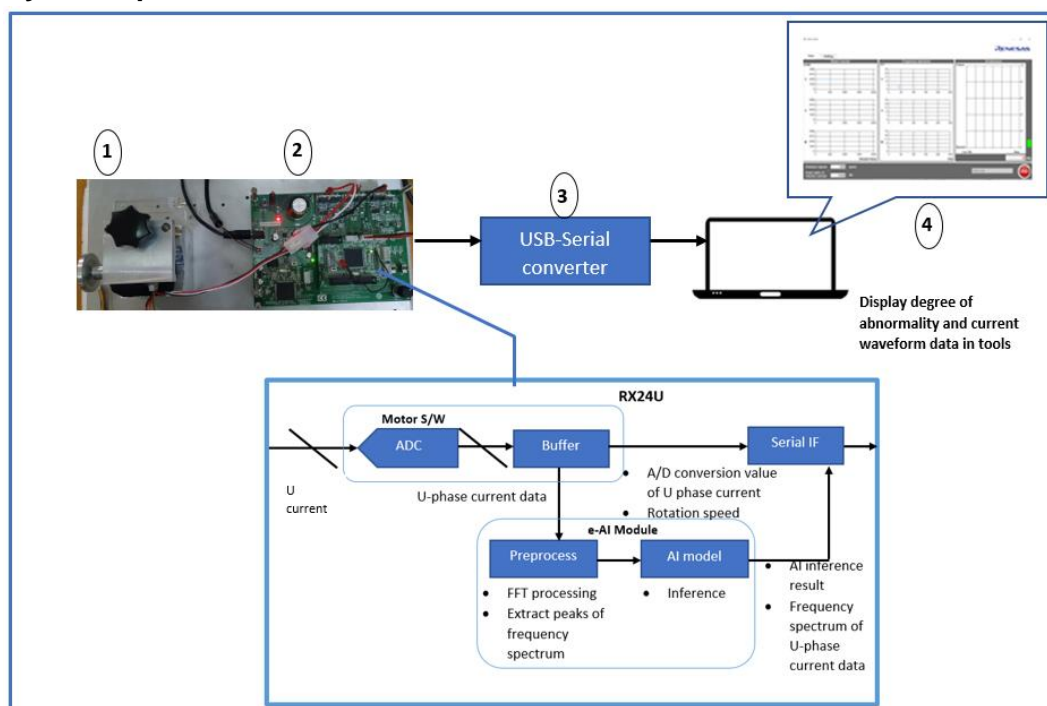
### 3.2.3  **System operation flow**



**Figure 13 System Operation Flow**

Figure 13 shows the system operation flow. The numbers in the Figure 13 correspond to the numbers in the function descriptions below.

1.  Execute sensorless vector control on motor

    When power is applied to the 24V inverter board, it is also applied to the RX24U CPU board (Hardware setup), which starts the motor driver operations.

2.  Execute pre-processing for motor drive current data, determine abnormality using e-AI inference

    a.  A/D conversion value accumulation

       The A/D conversion value of the motor U-phase shunt current are acquired based on sampling frequency. The U-phase shunt current is input to the 12-bit A/D converter (S12ADH). One frame (1024 samples) of A/ D conversion values are accumulated for the FFT.

    b.  Data pre-processing

       The MCU performs the FFT operation. First and Third harmonic of the fundamental frequency are extracted from the frequency spectrum (excluding the DC component).

    c.  AI inference

       The extracted First and Third harmonic from the frequency spectrum along with RPM in one hot encoding format are used as input to the trained DNN, and the probability of the two classes (normal and abnormal) are output by inference. In this example, the probability of abnormality is taken as the degree of abnormally.

3.  Serial communication with PC

    Using SCI5, data is transferred to the PC using a USB-serial converter cable.

4.  Display degree of abnormality and current waveform data in tools

    The received data is displayed in numerical values and graph form in the DataCollectionTool (GUI tool) run on the PC.
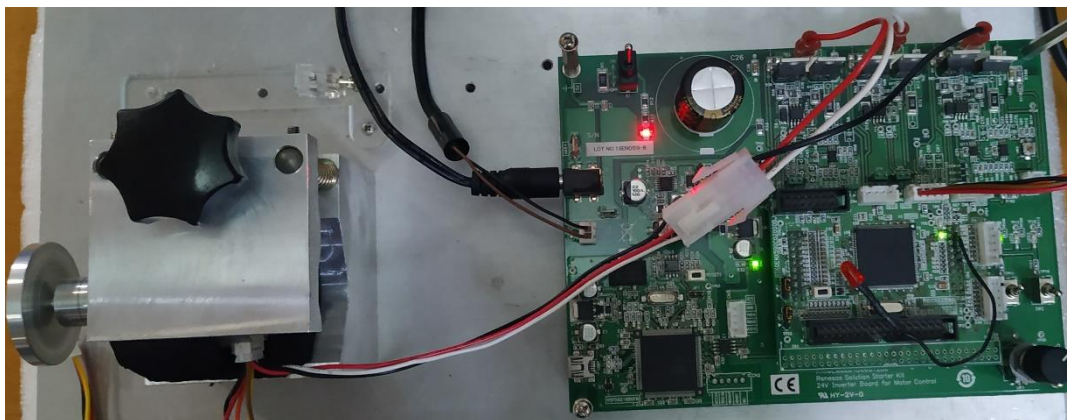
**Figure 14. RX24U enlarged demo setup**

Figure 15 shows images of the system in normal and abnormal states. Normal state is defined as when the drive motor and balanced load disc is balanced and abnormal state is defined when the load disc is imbalanced. In this example, normal and abnormal states are recreated using a simple motor bench, coupling the drive motor and disc load with an imbalance.
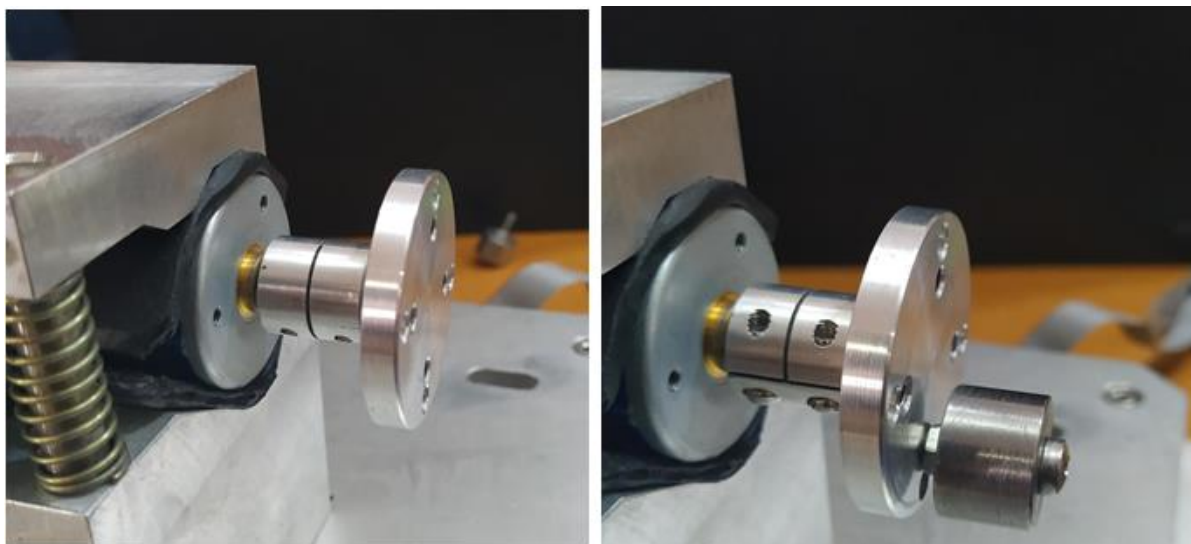


**Figure 15. Images of Normal (left) and Abnormal (right) States**

### 3.3  **State Transition**

Figure 16 provides an image of the state transition. The SW1, SW2, VR1, LED1 and LED2 discussed in this section indicate the devices mounted on the 24V inverter board of the 24V Motor Control Evaluation System.
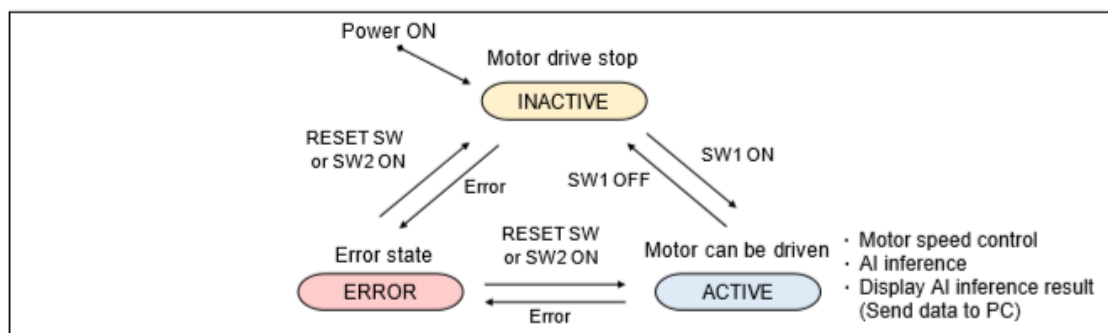


**Figure 16. State Transition**

(1)  INACTIVE

INACTIVE indicates the state immediately after power is supplied to the system. The motor is not driven in this state. When SW1 is turned ON, the system transitions to ACTIVE state. If an error is detected in the inactive state, the system shifts to the error state.

(2)  ACTIVE

In the ACTIVE state, LED1 goes on and the motor can be driven.

The following processes are carried out in the ACTIVE state.

- Motor rotation speed control

  VR1 controls the rotation speed of the motor.

- e-AI inference

  Infers the degree of motor abnormality.

- Data transfer to PC Sends the motor drive current data and/or inference result to the PC.

When an error occurs in the ACTIVE state, the system transitions to the ERROR state.

(3)  ERROR

When motor overcurrent or other error is detected, the LED2 lamp goes on and the system transitions to the ERROR state. To clear the error and return to the INACTIVE/ACTIVE state, push the RESET button or turn SW2 ON -> OFF.

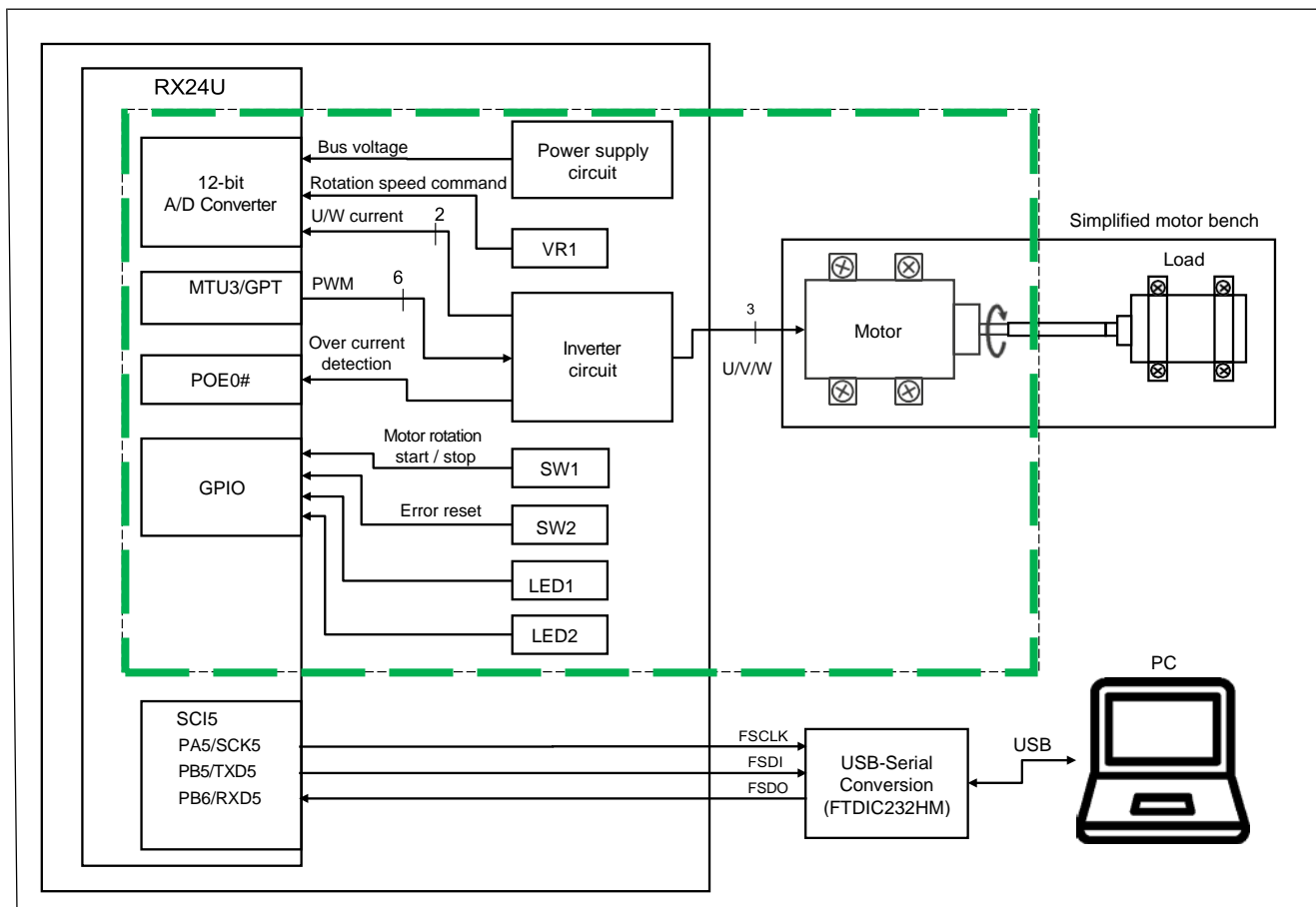### 3.4 **Hardware Diagram**



**Figure 17. Hardware Diagram**

## 4.   MCU Software Explanation

### 4.1   Software Configuration

Figure 18 shows the software configuration of the e-AI Motor Abnormality Detection Sample Software.
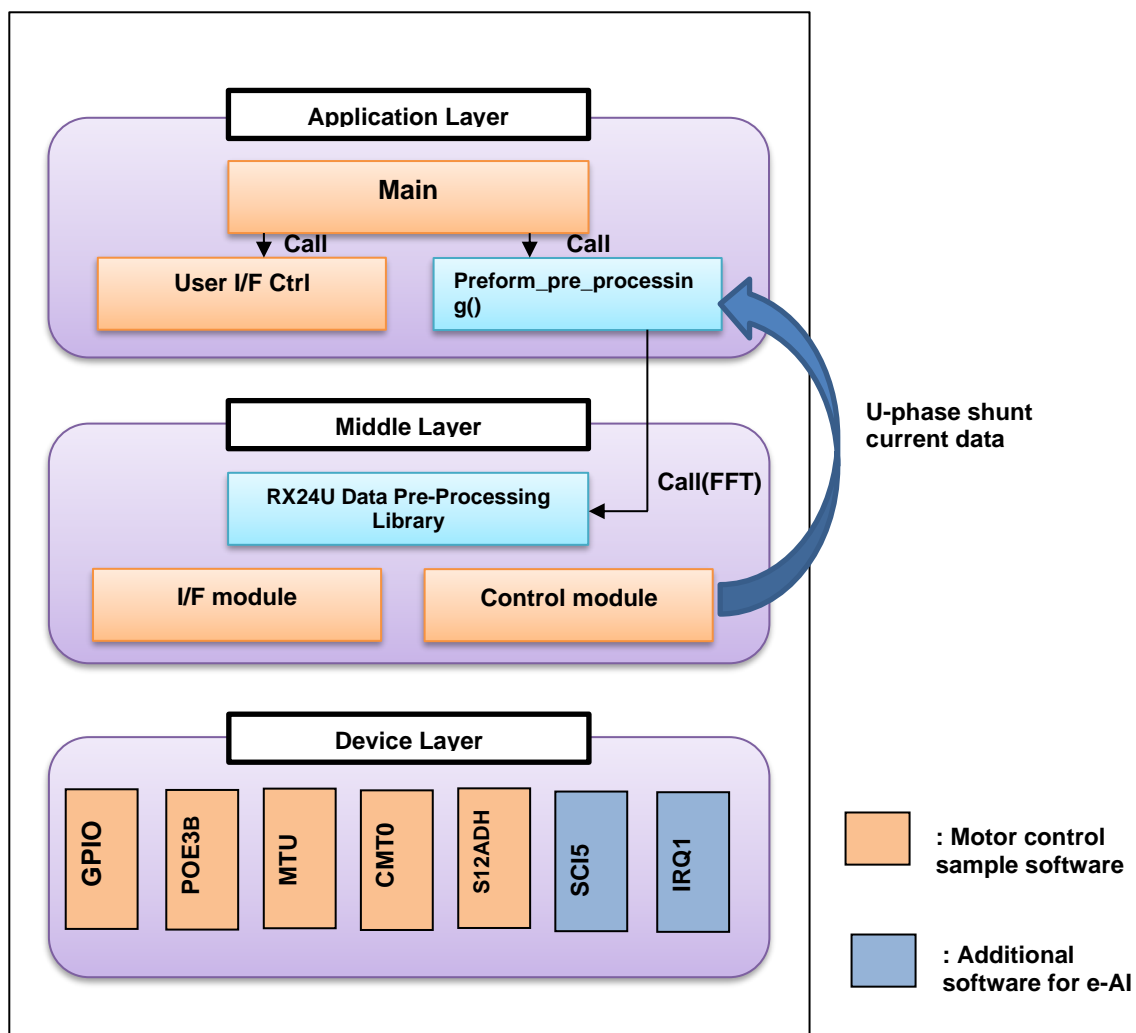


**Figure 18. Software Configuration Overview**

## 4.2 Directory Configuration

Figure 19 shows the directory configuration of the e-AI Motor Abnormality Detection Sample Software.
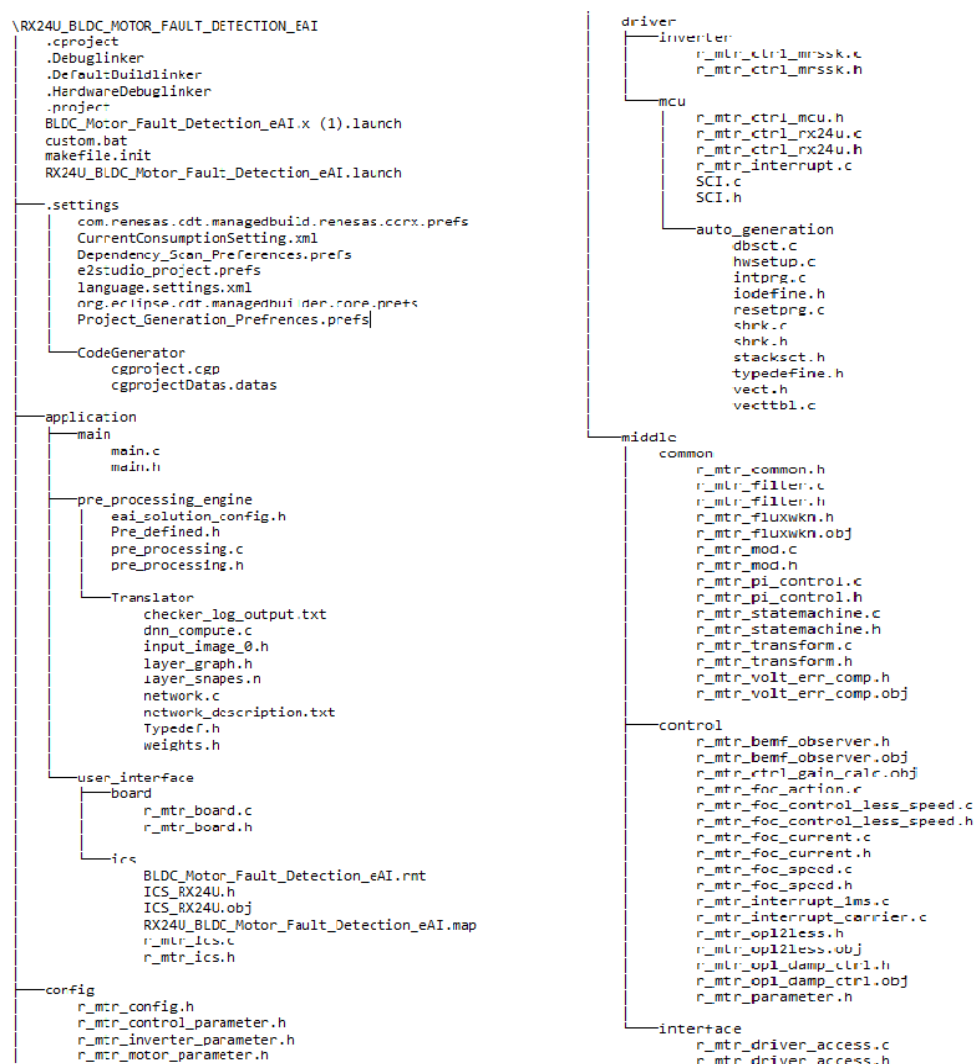


**Figure 19. Directory configuration**

## 4.3 Resources

### 4.3.1 Resource List

Table provides a list of resources used in the sample software.

| Item | Description |
|---|---|
| Clock generation circuit | Generates operating clock from external oscillator. |
| S12ADH | • U-phase shunt current measurement<br>• Inverter bus line voltage measurement<br>• Rotation speed command value input |
| CMT0 | 1ms interval timer |
| MTU3 | Complementary PWM output |

| | | |
|---|---|---|
| GPIO | • Switch (SW1, SW2) input<br>• LED (LED1, LED2) ON/OFF control | |
| SCI5 | Transfers data to send data register based on SCI5 transfer request | |

### 4.3.2 Interruptions

Table provides a list of interrupt processings.

| Interrupt request generation source (peripheral module) | Name (peripheral module name) | Interrupt priority level (IPR) | Description |
|---|---|---|---|
| Group Interrupt 1 （POE3） （SCI5） | GROUPBL1 （OEI1） （TEI5,ERI5） | 15 (highest) | • Motor interrupt processing (overcurrent detected) •<br>• Data send/receive to/from PC □<br>   - SCI5 send complete interrupt (TEI5) □<br>   -SCI5 receive error interrupt (ERI5) |
| PERIA （MTU3） | INTA209 (TCIV4) | 12 | Motor interrupt processing(1ms) |
| SCI5 | RXI5 | 12 | Data receive from PC |
| CMT0 | CMI0 | 11 | Motor interrupt processing (100 µs) |
| SCI5 | TXI5 | 8 | Clear transfer complete flag |

## 4.4 **Main Processing**

Figure 20 shows the flow chart for main processing. Main processing executes system initialization, then executes a loop of motor control process and AI inference process. The motor state and speed are controlled by user using the 24V inverter board devices (SW1, SW2, VR1). The motor control process monitors SW1, SW2, VR1 parameters and converts the input device information to system state and motor rotation speed command values. AI inference process performs data collection, pre-processing and prediction when system state is active.
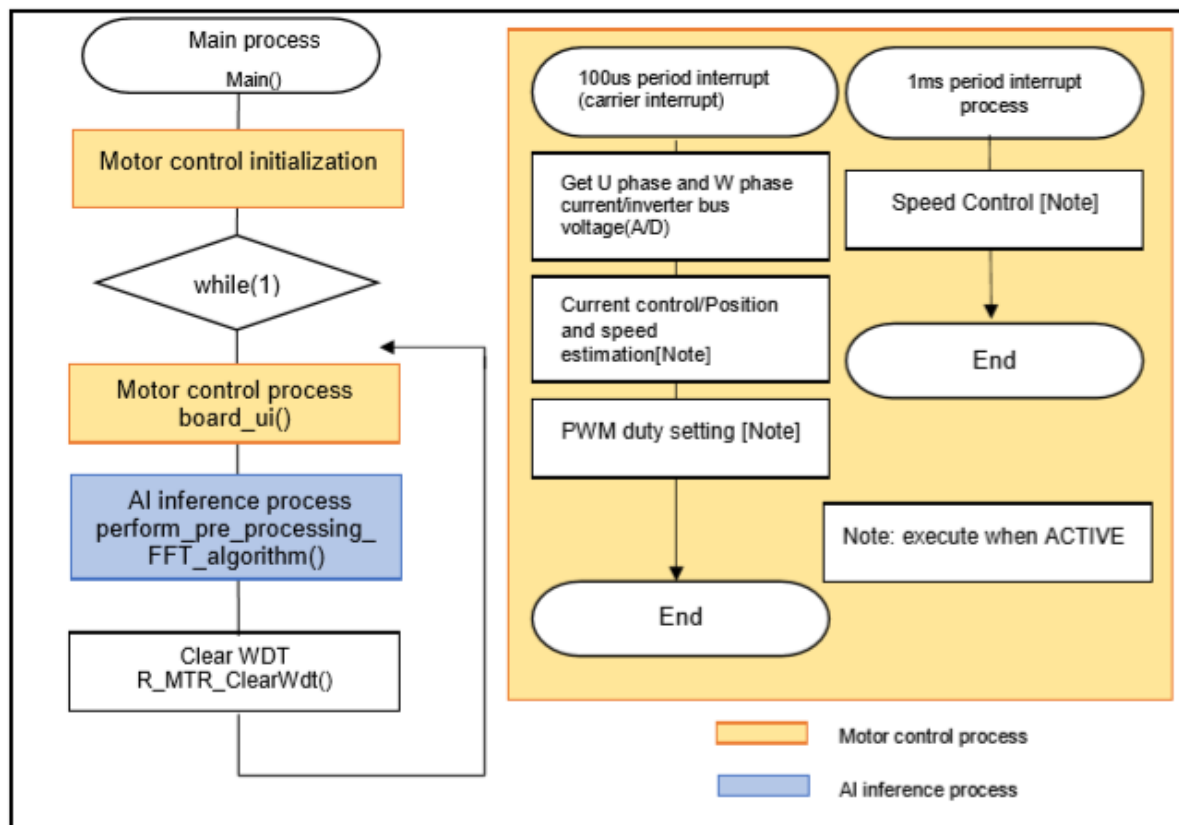


**Figure 20. Main Processing Flow Chart**

## 4.5 **Motor Control Processing**

The following definition has been changed in order to employ the 24V inverter board user interface.

Target file: : r_mtr_config.h

Before change:

```
#define  CONFIG_DEFAULT_UI  (ICS_UI)
```

After change:

```
#define  CONFIG_DEFAULT_UI  (BOARD_UI)
```

The user interface processing code (board_ui function) has been rewritten for this example in order to limit motor rotation speed.

**Note:-** Motor rotation speed is limited between RPM 1100 to RPM 1500

| Item | | Conversion Rate (command value: A/D conversion value) |
|---|---|---|
| Rotation speed command value | CW | 1100[rpm]~1500[rpm] |

Target file: main.c

Target function: board_ui()

Before change

```
/*============================*/
/*   Set speed reference    */
/*============================*/
u2_temp_vr1_signal = get_vr1();
s2_temp = (u2_temp_vr1_signal - ADJUST_OFFSET) * VR1_SCALING; /* Read speed
reference from VR1 */
```

After change: :

```
/*============================*/
/*    Set speed reference     */
/*============================*/
u2_temp_vr1_signal = get_vr1();
s2_temp = u2_temp_vr1_signal - ADJUST_OFFSET; /* Read speed

reference from VR1 */


/* Change VR1 A/D value to rotation speed command value of +1000 to
  +2000rpm*/
/* The area near the VR1 center is the dead zone.*/
if(-200 >= s2_temp)
{
s2_temp = (int16_t)(((0.542f * (float)s2_temp) - 891) *      (-1));
}
else if(200 <= s2_temp)
{
s2_temp = (int16_t)(0.542f * (float)s2_temp) + 891;
}
else
{
s2_temp = 0;
}
s2_temp = R_MTR_LimitAbs(s2_temp, g_u2_max_speed_rpm);
if(s2_temp >= 1100 )
{
 s2_temp = ((s2_temp+50)/100) * 100;
 R_MTR_SetSpeed(s2_temp); /* Set speed reference */
 g_u32_speed_rpm =s2_temp;
}
else
{
```

```
if(s2_temp>1010)
{
s2_temp = 1100;
R_MTR_SetSpeed(s2_temp); /* Set speed reference */
g_u32_speed_rpm =s2_temp;
}
else
{
s2_temp=0;
R_MTR_SetSpeed(s2_temp); /* Set speed reference */
g_u32_speed_rpm =s2_temp;
}
}
if(s2_temp > 1500 )
{
s2_temp = 1500;
R_MTR_SetSpeed(s2_temp); /* Set speed reference */
g_u32_speed_rpm =s2_temp;
 }
 else
 {

 }
s2_temp = R_MTR_LimitAbs(s2_temp,
g_u2_max_speed_rpm); R_MTR_SetSpeed(s2_temp);
/* Set speed reference */
```

## 4.6  FFT Processing

| Function Name | Description |
|---|---|
| FFT (int8_t nDir,int16_t nNPointFft, stXComplex *pstInOutData) | This routine updates the FFT computation to pstInOutData buffer.<br><br>pstInOutData - Pointer for input/output buffer to store the FFT computation. |

The following shows the FFT source code.

```
uint32_t i;

uint16_t SamplingHalf = gv_SamplingConditions.m_SamplingCount / 2;

//Computes FFT

  FFT(FFT_WITH_FORWARD_TRANSFORM, DATA_PREPROCESSING_DATA_POINT,
  (stXComplex *)&astComplex_FFT_Buffer);

//Magnitude

for(uint16_t uni = 0;uni <SamplingHalf ;uni++)

{

  pOutBuf[uni]=(uint32_t )(sqrt(pow(pInBuf[uni].fReal,2)+pow(pInBuf[uni].fIm
  ag,2)))*2.0;

 }
```

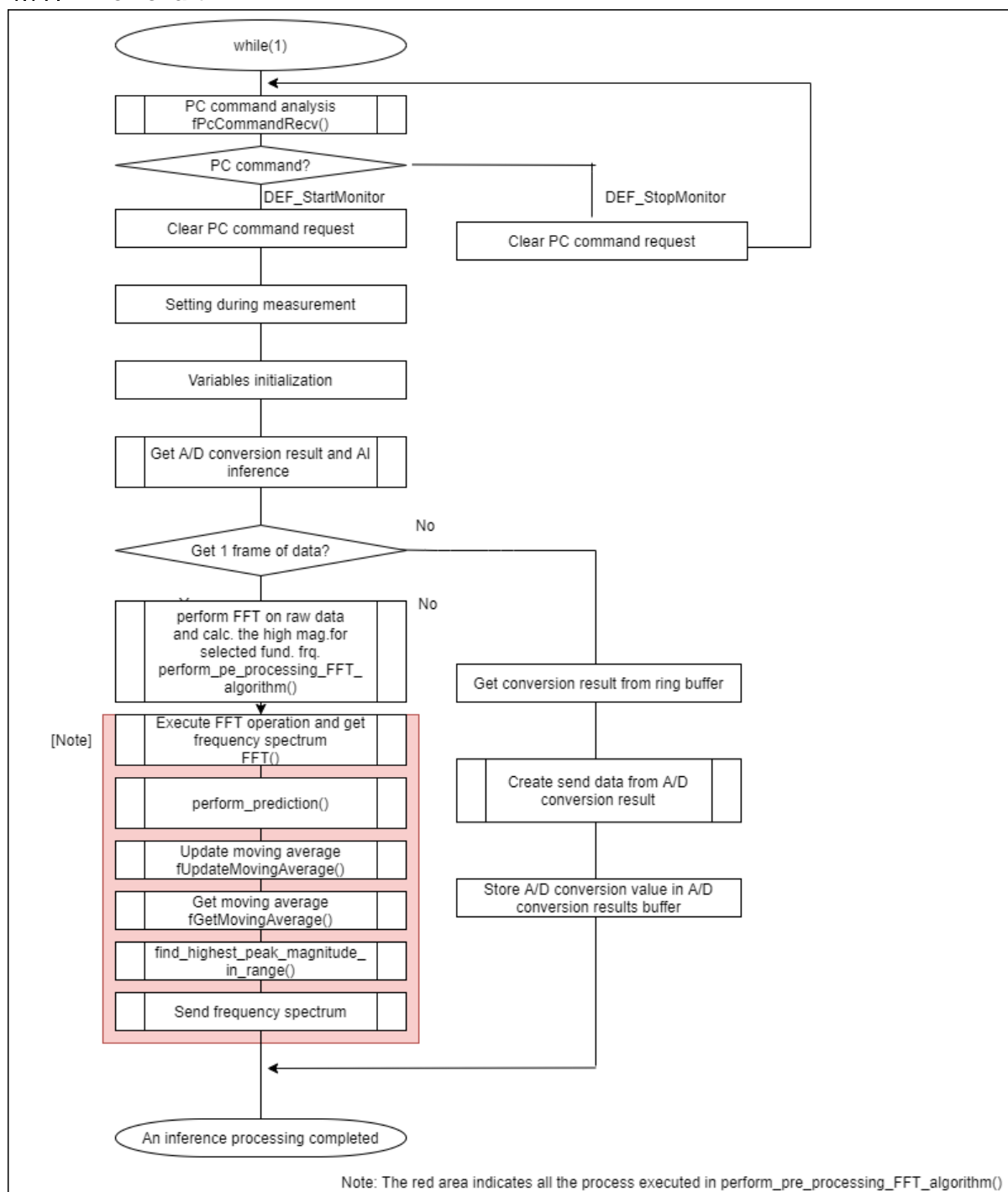## 4.7  **AI Inference Processing**

### 4.7.1  **Flowchart**



**Figure 21. AI Inference Processing Flowchart**
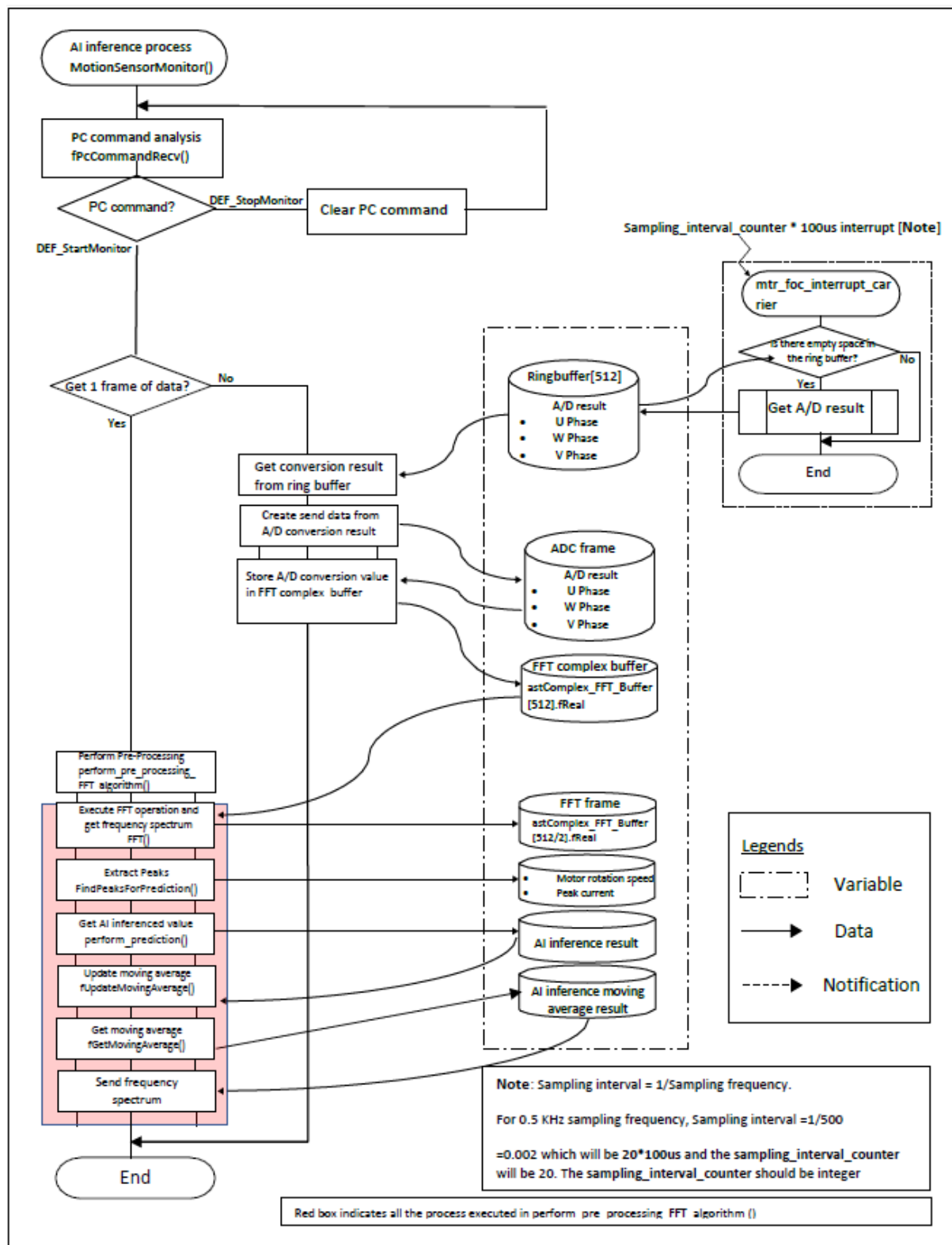
### 4.7.2　Data Flow



**Figure 22. AI inference processing data**

### 4.7.3  **AI Model**

Figure 23 shows the normal state and Figure 24 shows the abnormal state for this example. The degree of abnormality is inferred by e-AI from the difference in generated current waveform. In this software, disc imbalance is defined as abnormal.
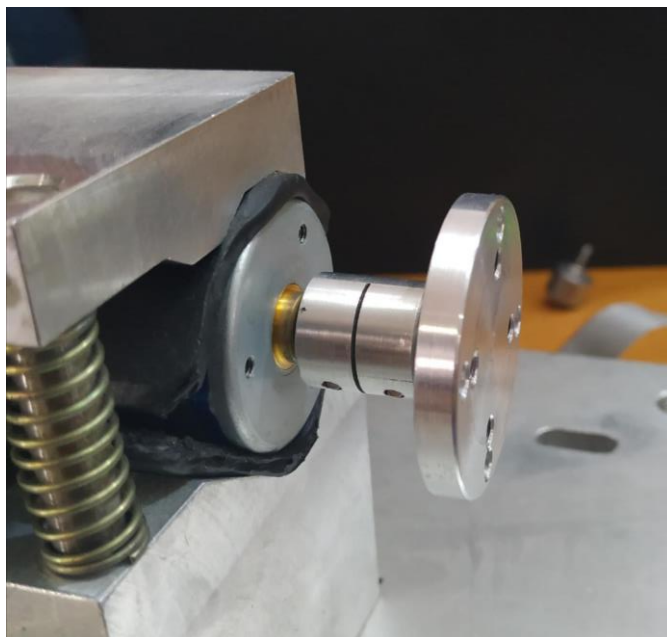


**Figure 23. Normal State**



**Figure 24. Abnormal State**

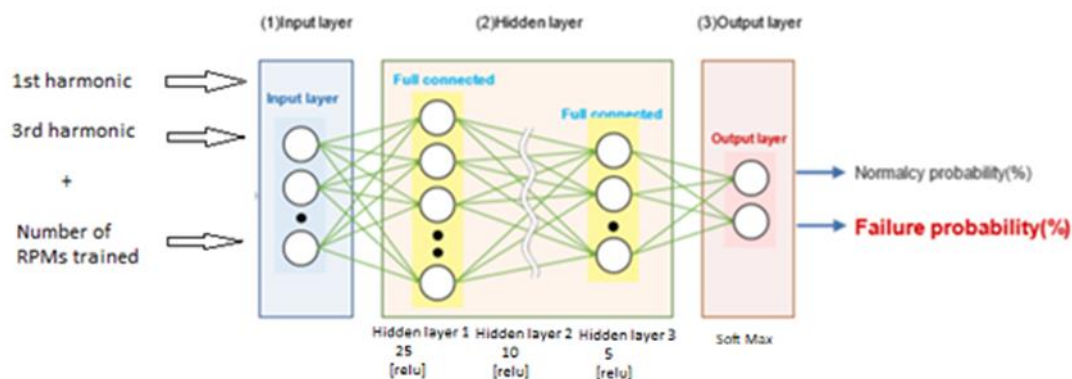Figure 25 shows the AI model configuration.



**Figure 25. AI Model Configuration**

1. Input layer

   The inputs to AI Model are 1st and harmonic from extraction of peak process and number of trained RPM. Iputs to model varies according to number of RPM trained.

2. Hidden layer

   Current AI Model contains 3 fully connected hidden layers with 25,10 and 5 neurons respectively in each hidden layer.

   "Relu" function is used as activation function in each hidden layer.

3. Output layer

   There are 2 outputs from the AI Model.

   The output layer outputs the probability of normality and abnormality.

   "SoftMax" function is used as activation function in output layer.

# 5. Training AI

## 5.1 Collecting data

- Set the work bench setup by referring to section 3.2.1
- Connect the Data Collection Tool by referring to section 3.2.2
- Click Start button to collect the data
- For detail instructions about data collection tool follow section 6.1

## 5.2 Train AI Model

- Run the training tool exe.
- Select the path "Training Mode" for detail instructions follow 6.2.2.2
- Choose the training data set path as data collection from Data Collection Tool.
- Select the required output Model path.
- Click Start.
- Once the Model creation is completed Model accuracy will be displayed.

## 5.3 Translate AI Model

- Open e2 studio.

- Click on 'TR'.



**Figure 26. e-AI translator**

- Browse input model location (Model need to translate).
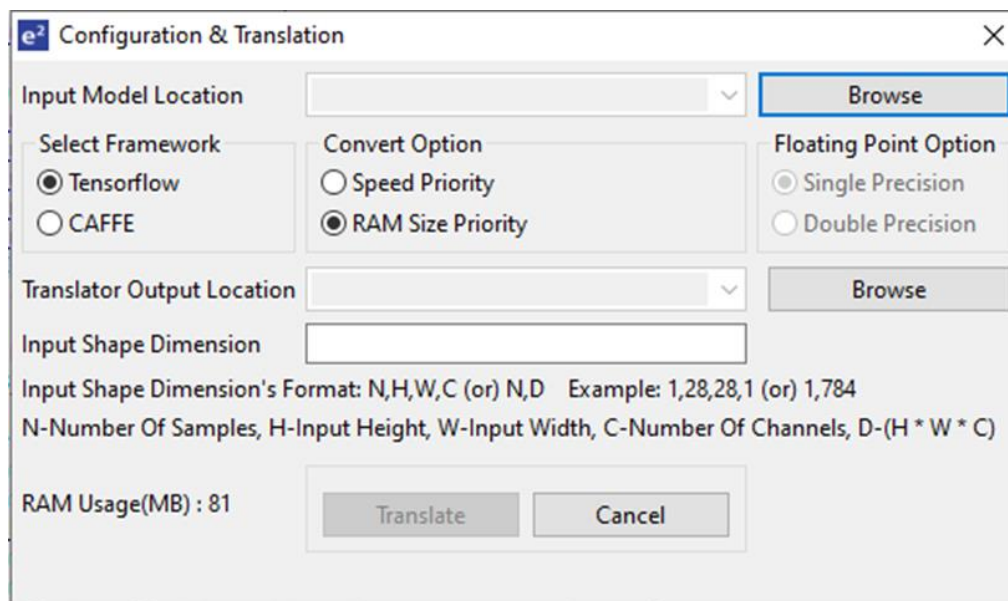


**Figure 27. Translating AI Model**

- Select framework as tensorflow and convert option as RAM size priority

- Please specify the output folder as project/application/pre_processing_engine folder.

- Give the input shape dimensions as 1, (2(number of extracted peaks*) + TOTAL_RPM_LIST (total trained RPM)).

- Translate option will be enabled, click on the translate button it will show the translation status

*The number extracted peaks are 1st and 3rd harmonic which are constant for RX24U.

## 5.4  Integrate to e2 studio project

### 5.4.1  Update of the trained DNN

Modify the input_image_0.h in the path  project/application/pre_processing_engine/translator using following code.

#include "Translator/Typedef.h"

#include "eai_solution_config.h"

TPrecision data_in[ 2 + TOTAL_RPM_LIST ] = { 0.0 };

The following **perform_prediction**function call inference processing API (dnn_compute function) output by e-AI Translator, besides, it executes pre-processing which extracts peaks from frequency spectrum before inference processing.

```
static int8_t perform_prediction()
{
  int8_t AIresult;
  uint8_t u8_idx = 0u;
```

```
    for (u8_idx = 0; u8_idx <  TOTAL_FREQUENCY_COUNT; u8_idx++)
    {
    if (u8_idx < (POLEPAIRCOUNT - 1))
    {
        data_in[u8_idx] = g_af_magnitude[u8_idx];
    }
    else
    {
        data_in[u8_idx] = g_af_magnitude[u8_idx + 1];
    }
    }

    for(u8_idx = 0; u8_idx <  TOTAL_RPM_LIST; u8_idx++)
    {
    if(g_u32_speed_rpm == g_cn_au16_rpm_list[u8_idx][0])
    {
        data_in[TOTAL_SELECTED_PEAKS + u8_idx] = 1;
    }
    else
    {
        data_in[TOTAL_SELECTED_PEAKS + u8_idx] = 0;
    }
    }
    g_u8_test_prediction =1;
    g_pf_prediction = (TPrecision *) dnn_compute(data_in);

    return AIresult;

    }
```

### 5.4.2 Update of the Configuration file with parameters from training tool to e2 studio project.

- The configuration file with parameters will generated along with DNN files as shown in Figure 28.

| | | | |
|---|---|---|---|
| checkpoint | 28-02-2020 11:43 | File | 1 KB |
| eai_solution_config.h | 28-02-2020 11:38 | H File | 4 KB |
| epoch.data-00000-of-00001 | 28-02-2020 11:43 | DATA-00000-OF-0... | 7 KB |
| epoch.index | 28-02-2020 11:43 | INDEX File | 1 KB |
| epoch.meta | 28-02-2020 11:43 | META File | 53 KB |
| epoch.png | 28-02-2020 11:43 | PNG File | 52 KB |
| SortedTrainingDataFile.csv | 28-02-2020 11:41 | Microsoft Excel C... | 17 KB |

**Figure 28. Parameter header file from training tool**

- Move the the eai_solution_config.h file to e2 studio roject\application\pre_processing_engine\

- Build the project.

- Flash the firmware and run the demo to get the AI inference.

## 6.  Overview of Tool

| Tool Name | Tool Usage |
|---|---|
| Data Collection Tool | This tool is used as waveform monitor tool. |
| Training Tool | This tool is used to train and test the model. |

### 6.1  Data collection Tool

The Data collection tool is used to receive the U-phase shunt current data, Frequency spectrum and AI inference value from the RX24U MCU via a serial communication and to display those results.

An overview of the waveform monitor tool operations is as follows.

- Control display start/stop
- Display data sent from RX24U
    - o U-phase shunt current waveform data
    - o Waveform data after U-phase shunt current data is FFT processed
    - o AI inference result
        - Moving average waveform
        - Actual value bar (0~100% displayed in increments of 10)
        - Numerical value
    - o Peak current value
    - o Motor rotation speed

#### 6.1.1  Function Explanations

The following describes each waveform monitor tool function in detail. The tool has a **View tab** for displaying all categories of information and a **Setting tab** for setting up operations.
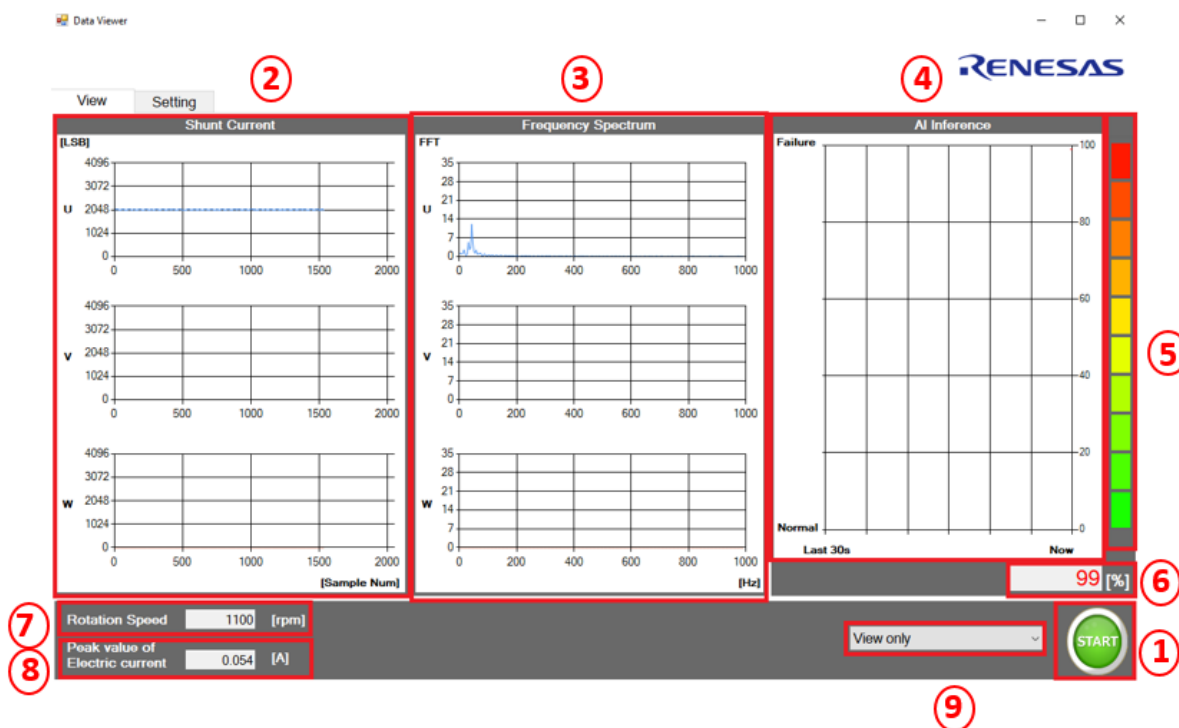


**Figure 29. View Tab Display Specifications**

### 6.1.1.1 View Tab

Figure 29 shows the display layout used in the View tab. The numbers in the figure  29 correspond to the numbers in the function descriptions below.

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

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

    - When the START button is pushed:
      - o 'Data Send Request Commands' are sent from the PC to RX24U, and data is sent from RX24U to the PC.
      - o Received data is displayed in real time.
    - When the STOP button is pushed:
      - o 'Data Send Stop Command' is sent from the PC to RX24U and data acquisition ends.

2.  **3 shunt current waveform data -**

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

    Currently only U-phase shunt current is displayed while V and W phase current will be zero always

3.  **Frequency characteristics –**

    The U-phase shunt current waveform data in (2) above are transformed into the frequency spectrum via FFT 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.

    Currently only U-phase shunt current frequency spectrum is displayed while V and W phase current will be 0 always.

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 U-phase 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 drop down list. The CSV file is stored in the "CSV Location" folder immediately under the C drive in the initial settings.

    - View only
      - o Only monitors various data.
    - Save to CSV (divided)
      - o This option is not supported.
    - Save to CSV (combined)

> o   This option is not supported.

- Save to CSV (RPM combined)

> o   This option outputs data for each axis in a single CSV file along with RPM information while viewing the data.

#### 6.1.1.2  **Setting Tab**

Figure 30 shows the display specifications for the Setting tab. The numbers in the figure 30 correspond to the numbers in the function descriptions below.
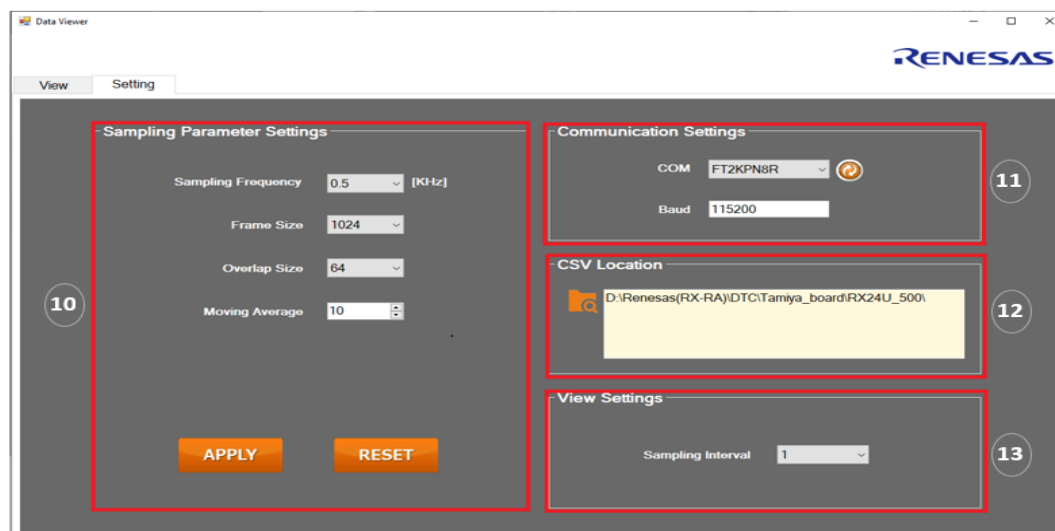


**Figure 30. Setting Tab Display Specifications**

**10.  Sampling parameter setting –**

The learned DNN in this example is optimized to the default setting except for the      moving average.

- **Sampling Frequency –** Specifies the sampling frequency (0.5/1/2/4/8 kHz, default: 2 kHz)

  **Note:** - For RX24U select Sampling Frequency as "0.5" only

- **Frame Size -** Specifies the FFT frame size (128/256/512/1024, default: 512).

  **Note:** - For RX24U select Frame Size as "1024" only

- **Overlap Size -** Specifies the FFT frame overlap size (16/32/64/128, default: 64).

- **Moving Average -** Specifies the moving average of the graph for the AI inference result (specified range: 1 to 10 times, default: 10).

**11.  Communication setting –**

- **COM -** Acquires and displays the name of the FTDI device connected to the PC.

- **Baud -** Specifies the Baud rate for communications between the MCU and PC (range: 9600 to 5000000, default: 5000000).

  **Note:** - For RX24U give Baud rate as "115200" only

**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).

### 6.1.2  **Operations**

#### 6.1.2.1  **Startup**

Connect the Demo device and PC with a USB cable. Open the DataCollectionTool.exe as shown in the figure 31.
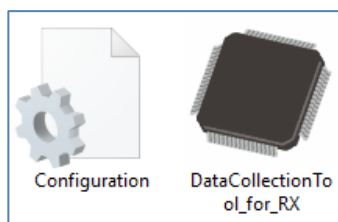


**Figure 31. DataCollectionTool.exe file**

If you open the exe file before connecting the demo device to the PC with a USB cable, you will get an error warning, as shown in Figure 32.
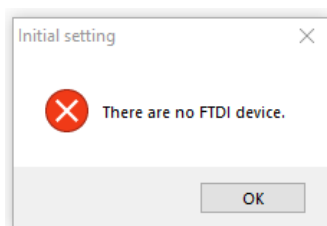


**Figure 32. Error Display**

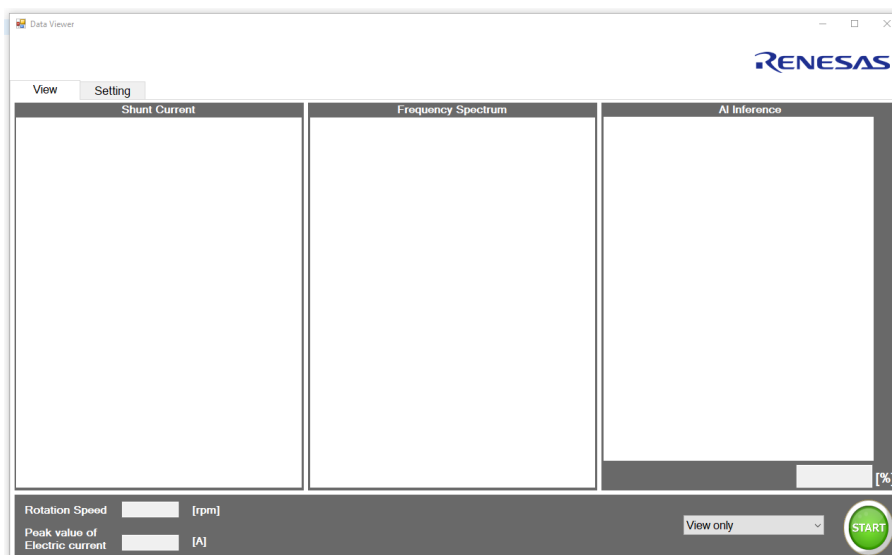The screen shown in Figure 33 appears when you open the exe file.



**Figure 33. Initial Startup Screen**

#### 6.1.2.2  **Start data acquisition**

Push the START button shown in Figure 34, to start data acquisition.

**Figure 34. Data Acquisition Start Button**

After data acquisition starts, the acquired data is displayed in the View screen.
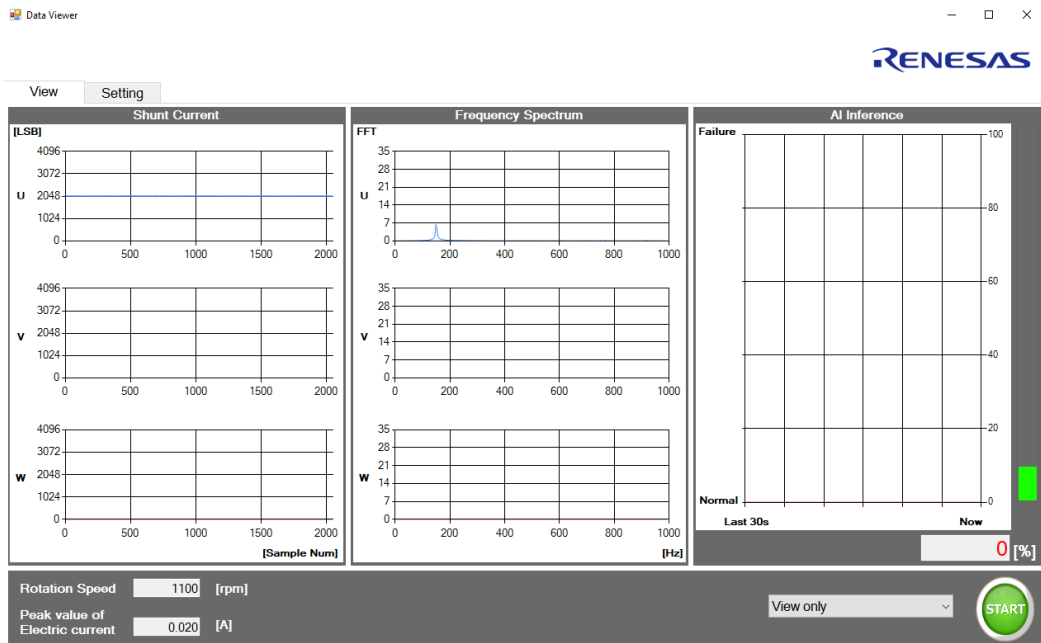


**Figure 35. Screen During Data Acquisition**

### 6.1.2.3 Stop data acquisition

Push the STOP button shown in Figure 36, to stop data acquisition.



**Figure 36. Data Acquisition Stop Button**

By selecting "Save to CSV(RPM combined)" from the drop-down list shown in "View tab," you can output data for each axis in a single CSV with RPM information file while viewing the data. Figure 37 shows the files created for saving the output data.



**Figure 37. File Created when "Save to CSV(combined)" is selected**

## 6.2   Training Tool

The Training tool is used to train and create the model using the collected data from Data collection tool An overview of the Training tool operations is as follows.

- Preforming pre-processing on collected data from Data collection tool.

- Training the Model using pre-processed data.

- Creating the trained Model.

- Testing of trained Model.

- Generating Parameters header file(eai_solution_config.h).

### 6.2.1   Functions Explanations

The following describes each Training tool function in detail using  number indicated in Figure 38.

The modes in Training Tool are Training and Testing.

1. **Training**: This mode will train the Model with Training data set, performs pre-Processing and creates Model with parameter header file.

2. **Testing**: This mode will test the created model with the input data set.

3. **Training Data Set**: This will the path of collected data from Data collection tool for training the model.

4. **Output AI Model**: this will the path for the created model.

5. **Loading and Preprocessing**: This is the progress bar for pre-processing.

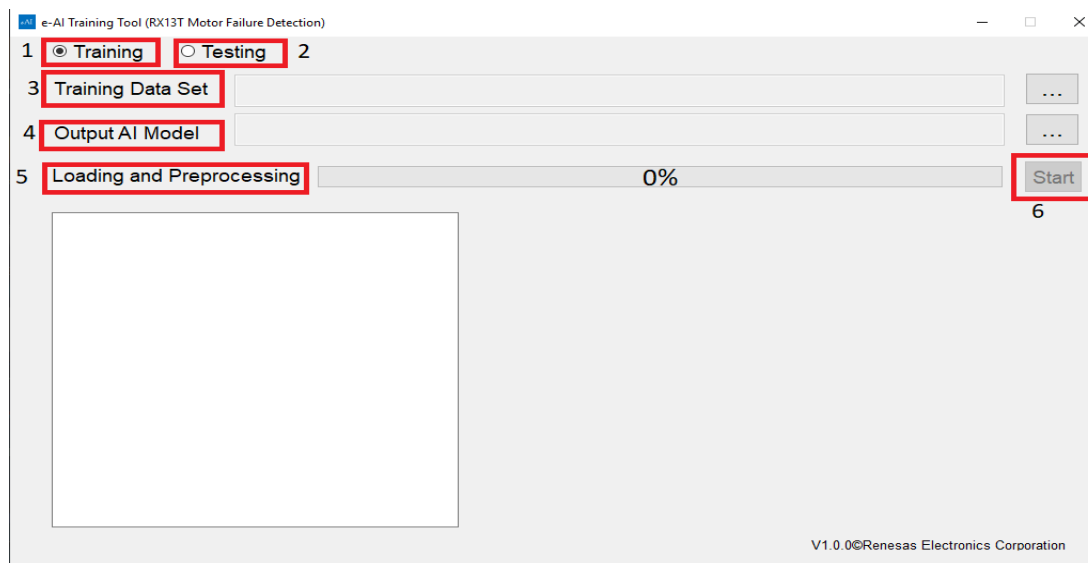6. **Start**: This is the button to start the training or testing.



**Figure 38. Training tool functionality**

### 6.2.2   Operations
### 6.2.2.1   Run the exe



**Figure 39. Training tool exe**

### 6.2.2.2  Training tool "Training Mode"

Train the AI model in the sequence shown in Figure 40. If accuracy does not increase, please re-train the AI model.
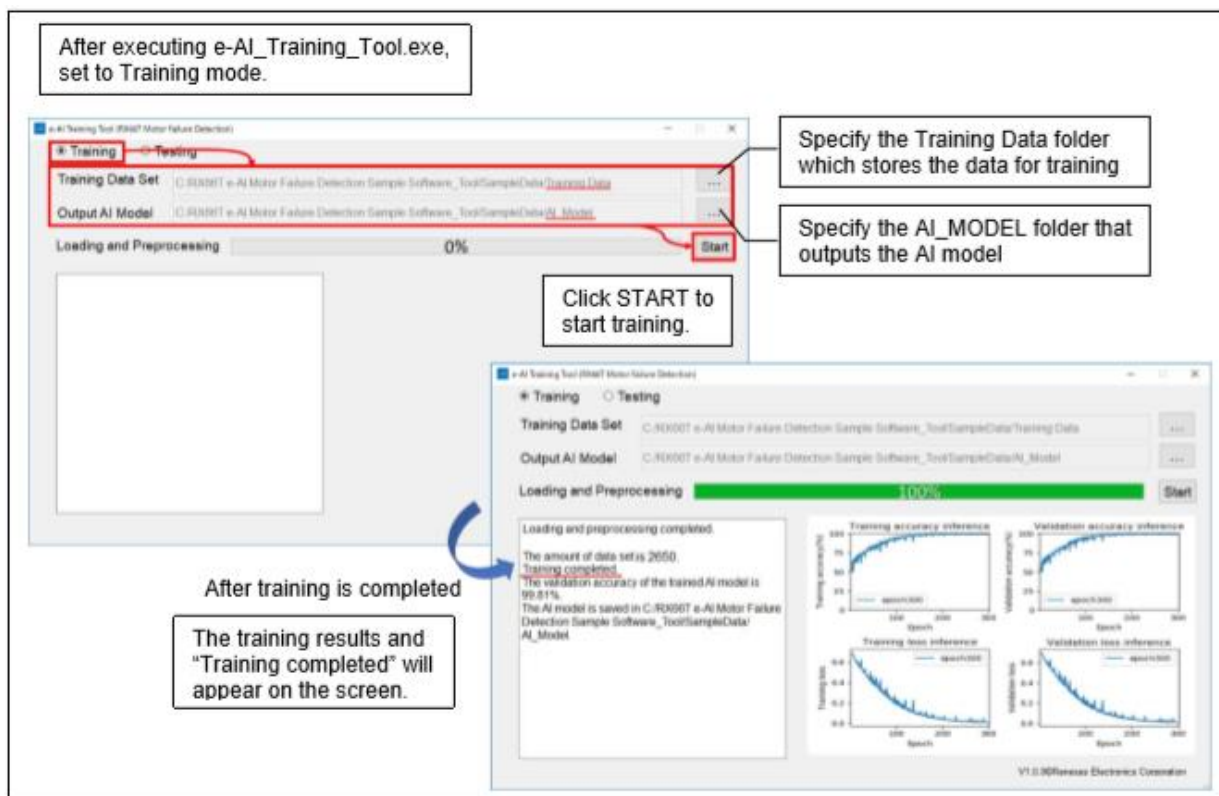


**Figure 40. Training Tool (Training)**

Confirm that the files shown in Figure 41  have been generated in the AI_MODEL folder.

| Name | Date modified | Type | Size |
|---|---|---|---|
| checkpoint | 25-02-2020 19:08 | File | 1 KB |
| eai_solution_config.h | 25-02-2020 19:04 | H File | 4 KB |
| epoch.data-00000-of-00001 | 25-02-2020 19:08 | DATA-00000-OF-0... | 7 KB |
| epoch.index | 25-02-2020 19:08 | INDEX File | 1 KB |
| epoch.meta | 25-02-2020 19:08 | META File | 53 KB |
| epoch.png | 25-02-2020 19:08 | PNG File | 52 KB |
| SortedTrainingDataFile.csv | 25-02-2020 19:07 | Microsoft Excel C... | 15 KB |

**Figure 41. Model File**

### 6.2.2.3  Training tool "Testing Mode"

Continue to operate e-AI_Training_Tool.exe. Test the trained AI Model by following the sequence shown in Figure 42.
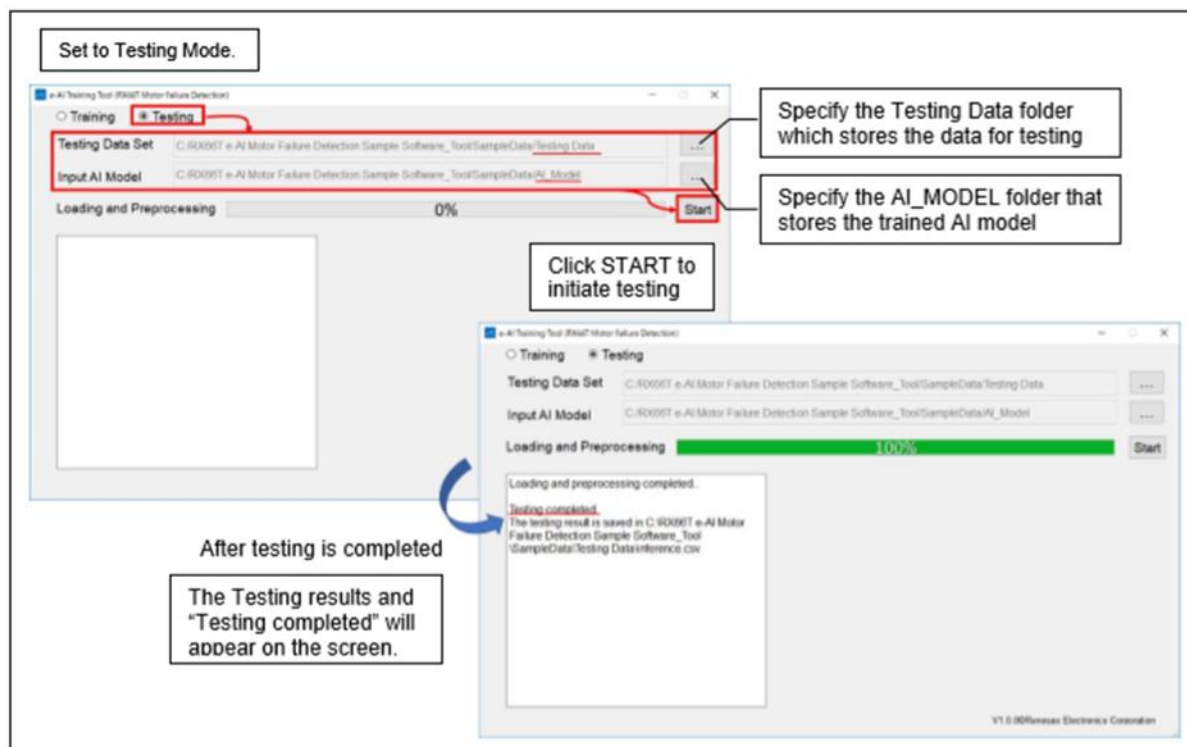


**Figure 42. Training Tool (Testing)**

The Testing results will be output as "inference.csv" in the Testing Data folder. Figure 43 shows how to confirm the Testing results.



| 1st Harmonic | 3rd Harmonic | RPM (hot encoding Format) | Normal or Abnormal file Normal =0 Abnormal=1 | Normal output | Abnormal output |
| --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 |
| 2.361972 | 3.706163 | 0 | 1 | 5.80E-05 | 0.999942 |
| 3.921628 | 5.35094 | 0 | 1 | 4.03E-09 | 1 |
| 2.307823 | 4.455802 | 0 | 1 | 1.69E-07 | 1 |
| 3.029894 | 2.57609 | 0 | 1 | 1.93E-05 | 0.999981 |
| 1.909905 | 3.328314 | 0 | 1 | 0.001633 | 0.998367 |
| 3.738885 | 5.047849 | 0 | 1 | 2.49E-08 | 1 |

**Figure 43. Inference Results**

## 7.  Reference Documents

[1] RX24U Group User's Manual: Hardware (R01UH0658)
[2] RX24T/RX24U Sensorless Vector Control for Permanent Magnetic Synchronous Motor (Implementation) (R01AN3788)
[3] Renesas Solution Starter Kit 24V Motor Control Evaluation System for RX23T User's Manual (R20UT3697)
[4] RX24U CPU Card User's Manual (R12TU0018)
[5] RX Family RX DSP Library Version 5.0 (R01AN4359)
[6] RX Family Sample Program for Performing FFT on Analog Input Signals (R01AN4015)
[7] User's Manual for e-AI Translator (R20UT4135)
[8] e2 studio Integrated Development Environment User's Manual: Getting Started Guide (R20UT4374)

Appendix1. MCU Software: detailed information

## 1. Memory Usage

| Item | Total Size | Description |
|---|---|---|
| RAM | 40.02KB | — |
| ROM | 41.87KB | Romdata section : 11.36 KB<br>Program section : 30.51 KB |

## 2. CPU Load

| Item | Processing Time |
|---|---|
| FFT process (FFT()) (*) | 20ms |
| e-AI inference process (Perform_prediction()) (*) | 1ms |

## 3. Smart Configurator Setting

### i. Clock settings

Table lists the clock settings for the sample software.

| Item | Description |
|---|---|
| VCC | 5V |
| Main clock | Select ON |
| Oscillation source | Oscillator |
| Frequency | 8MHz |
| PLL circuit division ratio | x1 |
| PLL circuit multiplication ratio | x20.0 |
| SCKCR(FCLK[3:0]) | x1/4 (FCLK=32MHz) |
| SCKCR(ICLK[3:0]) | x1 (ICLK=80MHz) |
| SCKCR(PCLKA[3:0]) | x1/2 (PCLKA=80MHz) |
| SCKCR(PCLKB[3:0]) | x1/4 (PCLKB=40MHz) |
| SCKCR(PCLKD[3:0]) | x1/4 (PCLKD=40MHz) |
| SCKCR(BCK[3:0]) | x1/4 (BCLK=40MHz) |

### ii. SCI5

Use SCI5 to carry out serial communication between RX24U and the pc. FTDI's USB – serial converter cable serves as the serial communication interface

*1: Multi-Protocol Synchronous Serial Engine

Table lists the settings for SCI5.

| Item | Description |
|---|---|
| Communication method | SCI asynchronous mode |
| Start bit detection | When RXD8 pin is Low |
| Data bit length | 9 bits |
| Parity | NONE |
| Stop bit | 1 bit |
| Data transfer direction | LSB first |
| Transfer clock | Internal clock (PCLKB) |
| Bit rate | 115200 |
| Bit modulation function | Enabled |
| SCK5 pin function | Clock output |

| Send data processing | Processed in interrupt service routine |
|---|---|
| Receive data processing | Processed in interrupt service routine |
| Callback function setting | Send complete, receive complete, receive error |
| Pins used | RXD5-PB6 pin-38<br>TXD5-PB5 pin-39 |

Notes1.    Default values are used for settings not listed here.

Notes2.    This is defined in the sample code in Reference Document [2]. Please do not change.


The following code is added to set the 9th bit of send data initial value to '0'.

Target file: Config_SCI5_user.c(output by smart configurator)

Target function : R_Config_SCI5_Create_UserInit function (created by smart configuration)


```
   void R_Config_SCI5_Create_UserInit(void)
{
    /* Start user code for user init. Do not edit comment generated here */
    /*Set 9th bit to "0"*/
    SCI5.TDRHL.BYTE.TDRH = 0xFE;
    /* End user code. Do not edit comment generated here */
}
```


## 4.   Functions

Table lists the functions used in main processing.

| Function Name | Description |
|---|---|
| board_ui | User interface using board |
| ics_ui() | User interface using ICS (Analyzer) |
| ics2_init() | Initialize ICS |
| software_init() | Initialize private global variables |
| clrpsw_i() | Interrupt disable |
| R_MTR_InitHardware() | Initializes peripheral functions |
| R_MTR_InitBoardUi() | Initializes peripheral functions for user intrface |
| R_MTRInitControl() | Initialze motor FOC control |
| mtr_board_led_control() | LED control |
| SCI5_init() | Initialize SCI5 |
| R_MTR_ExecEvent() | Execute event of FOC event |
| setpsw_i() | Interrupt enable |
| R_MTR_ChargeCapacitor() | Wait for charging capacitor |
| R_SCI5_Serial_Receive | Starts SCI reception(asynchronous mode) |
| R_MTR_SetUserifMode() | Set User interface mode |
| R_MTR_ClearWdt | Clears Watch dog timer |
| SCI_TX_U16 | This function transmits SCI5 data as unit16 type. |
| SCI_TX_U8 | This function transmits SCI5 data as unit8 type. |
| fPcCommandRecv() | Analyzes PC Command |
| mtr_set_com_variables | |
| R_MTR_GetStatus() | Get status of motor control |

Below table lists the functions used in pre-processing.c

| Function Name | Description |
| --- | --- |
| fUpadteMovingAverage() | Updates AI inference value moving average data |
| fGetMovingAverage() | Acquires AI inference value moving average data |
| fPcGetData() | Processing of data received from PC |
| SCI_TX_U32 | This function transmits SCI5 data as unit32 type |
| SCI_TX_U8 | This function transmits SCI5 data as unit8 type |
| Perform_pre_processing_FFT_algorithm | This routine performs the FFT on raw data, calculates the high magnitudes for the selected fundamental frequency. |
| find_highest_peak_magnitude_in_range | This rountine calculated the 1$^{st}$ and 3$^{rd}$ hormonic required for AI inference by using fundamental frequency. |
| perform_prediction | This routine performs the Inference using the peaks calculated by find_highest_peak_magnitude_in_range function. |
| FindPeaksForPrediction | This routine finds the fundamental peak which is the maximum peak in the spectrum. |

Below table lists the functions used in SCI.c

| Function Name | Description |
| --- | --- |
| SCI5_init() | This function receives SCI5 data. |
| SCI_RX_U16 | This function receives SCI5 data. |
| SCI_TX_STRING(const char *pcS) | This function transmits SCI5 data as string type. |
| SCI_TX_U16(uint8_t *pu8_buffer) | This function transmits SCI5 data as uint16 type. |
| SCI_TX_U8(uint8_t *pu8_buffer) | This function transmits SCI5 data as uint8 type. |
| R_SCI5_Serial_Send(const uint8_t * u8TXBuf, uint16_t u16TXNum) | This function transmits SCI5 data. |
| R_SCI5_Serial_Receive(uint8_t * const pu8RXBuff, uint16_t u16RXNum) | This function starts SCI reception (asynchronous mode). |
| r_sci5_callback_receiveerror(void) | Performs processing in response to the receive error interrupts. |
| SCI_TX_U32(uint8_t *pu8_buffer) | This function transmits SCI5 data as uint32 type. |

## 5. Variables

Table lists the global variables used for main.c

| Type Name | Variable Name | Description |
|---|---|---|
| uint8_t | u1_reset_req | Reset request flag |
| uint8_t | g_au8_rx_char[] | SCI5 receive buffer |
| uint8_t | g_u8_test | |
| uint8_t | gv_PcCommand | Receive command from PC |
| uint16_t | g_16_samples_cnt1 | Number of samples transmitted |
| uint16_t | g_u8_ShuntCurrentBuffer | ADC data buffer |
| float | FD_X_axis | Preprocessing input buffer |
| uint16_t | gv_MonitorStatus | Measurement start/stop status |
| uint8_t | start_Flag | Start Acquisition flag |
| uint8_t | g_u1_motor_status | Motor status |
| uint8_t | com1_u1_sw_userif | User interface switch |
| uint8_t | g_u1_sw_userif | User interface switch |
| uint8_t | com_u1_mode_system | System Mode |
| uint8_t | g_u1_mode_system | System Mode |
| float | g_u32_speed_rpm | Current RPM |
| | g_u2_max_speed_rpm | Max RPM of Motor |
| uint16_t | g_u16_idx_test | Index of ADC ring buffer |

Below table lists the global variables used for pre-processing.c

| Type Name | Variable Name | Description |
|---|---|---|
| float | g_current | Shunt current at that instance |
| float | g_u32_peak_current | Maximum Shunt current at that instance |
| float | g_u32_moving_avg | Moving average |
| char | g_u32_tx_buff[DEBUG_PRINT_BUFF_LEN] | Char buffer used to send string. |
| unit16_t | gv_PcRecvStatus | Status of receive from PC |
| unit32_t | gv_PcRecvTimer | Monitor timer for receive from PC |
| unit8_t | gv_PcCommand | Receive command from PC |
| unit16_t | gv_MonitorStatus | Measurment start/stop status |
| float | g_af_magnitude[TOTAL_FREQUENCY_COUNT] | Buffer for Peaks of FFT data |
| unit16_t | g_au16_freq_bins[TOTAL_FREQUENCY_COUNT] | Buffer for bins of Peaks |
| unit16_t | g_au16_freq_bins_highest_idx[TOTAL FREQUENCY COUNT] | Buffer for fundamental,first and second hormonics |

Below table lists the global variables used for r_mtr_interrupt_carrier.c

| Type Name | Variable Name | Description |
|---|---|---|
| uint8_t | g_u8_100usec_cnt | Number of samples transmitted |
| uint32_t | g_u32_sampling_time_100_usec | Sampling time in micro seconds |
| uint16_t | g_16_samples_cnt | Index of Ring buffer |
| uint16_t | g_u8_ShuntCurrentBuffer[] | Shunt current buffer |
| uint16_t | g_u16_adc_iu_ad | U-phase current value [A] |
| unit32_t | g_u32_acquisition_delay_1msec_cnt | Acquisition delay one millisecond counter |

Below table lists the global variables used for SCI.c

| Type Name | Variable Name | Description |
|-----------|---------------|-------------|
| uint8_t | *gp_u8_sci5_tx_address | SCI5 transmit buffer address |
| uint8_t | *gp_u8_sci5_rx_addres | SCI5 receive buffer address |
| uint16_t | g_u16_sci5_rx_count | SCI5 receive data number |
| uint16_t | g_u16_sci5_rx_length | SCI5 receive data length |
| uint8_t | g_u8_sci5_tx_done | SCI5 transmit flag |
| uint8_t | g_au8_rx_char[] | SCI5 receive buffer |
| uint8_t | g_u8_rx_flag | SCI5 receive flag |

## Revision History

| Rev. | Date | Description |  |
|------|------|------|------|
|  |  | **Page** | **Summary** |
| 1.00 | Apr. 23, 2020 | - | First Edition issued |

# 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.

# Notice

## Corporate Headquarters

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

## Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

## 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/.