RX13T 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 RX13T.

The sample software comes with learned Deep Neural Network (DNN). The e-AI system operations can be immediately confirmed on the required hardware described in this document.

Target Device

RX13T (RX13TLQFP48).

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.

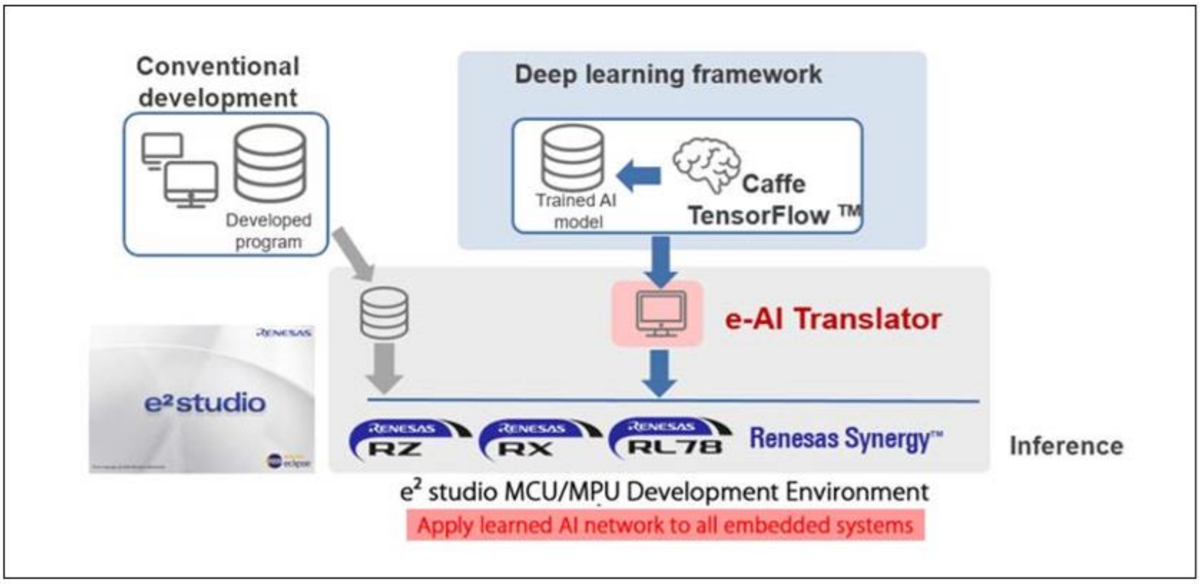


Figure 1

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.

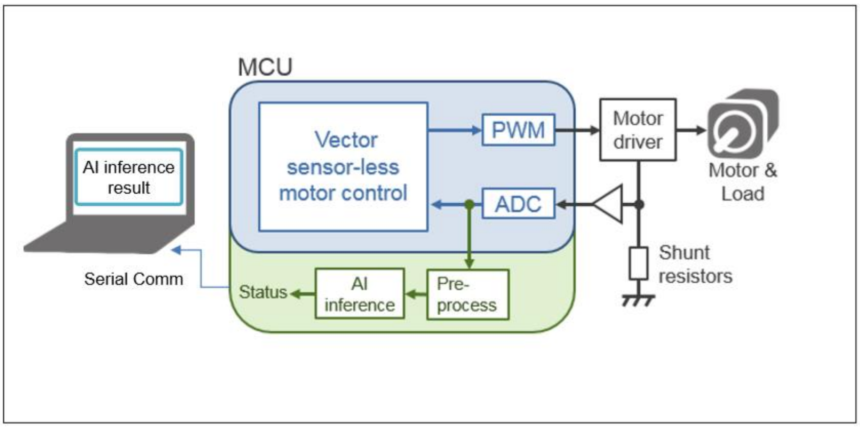


Figure 2. System Block Diagram

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.

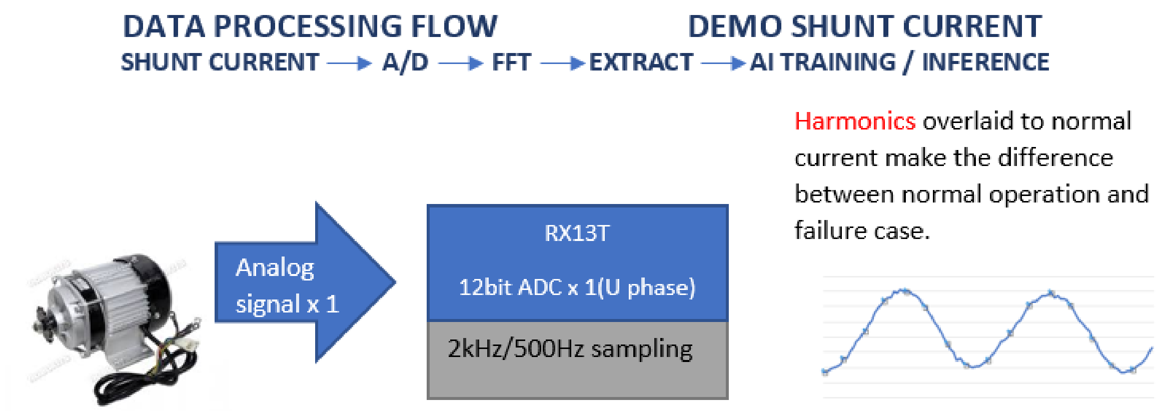


Figure 3. Data Processing Flow (1/2)

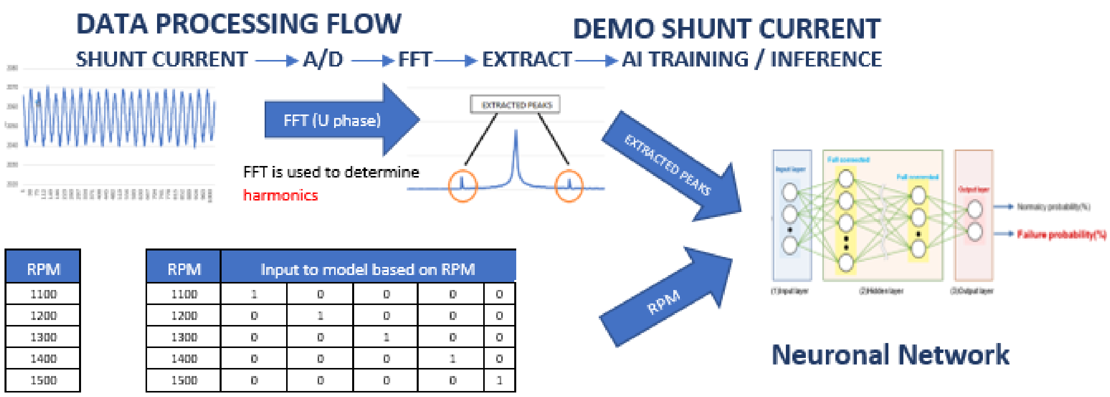


Figure 4. Data Processing Flow (2/2)

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.

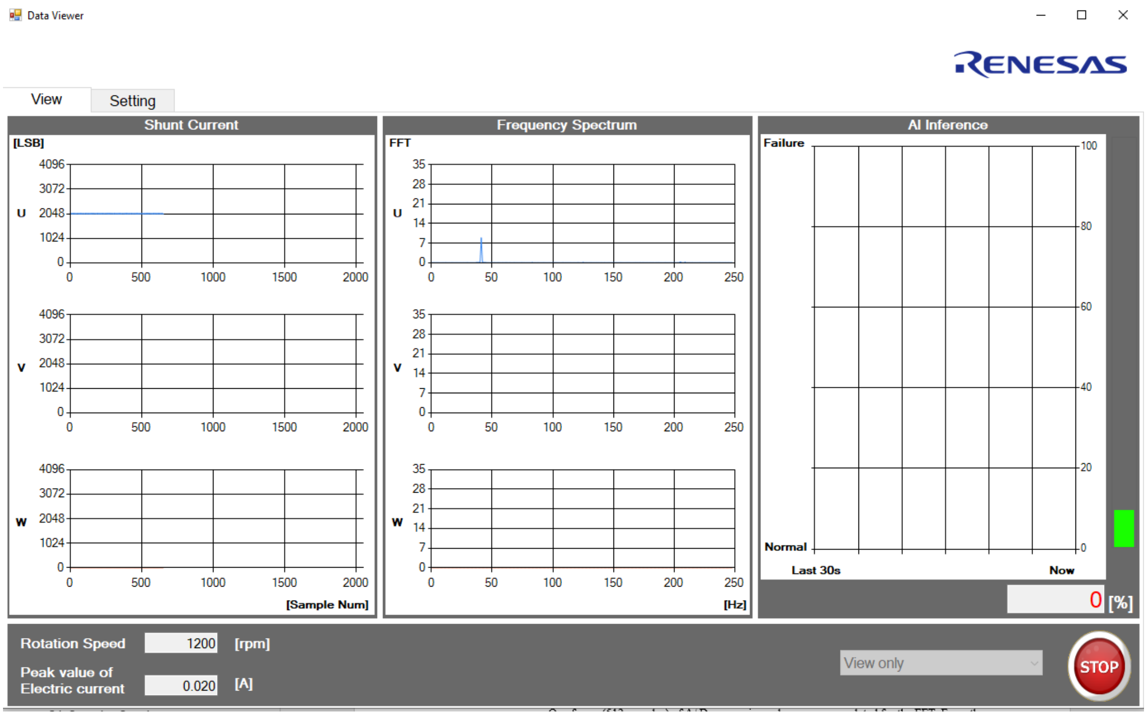


Figure 5. DataCollectionTool - Appearance

3. Specification

3.1 Operating Conditions

Item	Description
MCU	RX13T
Operating Frequency	Main clock oscillator frequency: 1 to 20 MHZ CPU clock (ICLK): 32MHz Peripheral module clock A (PCLKA): 40MHz Peripheral module clock B (PCLKB): 32MHz Peripheral module clock D (PCLKD): 40MHz FlashIF clock (FCLK): 32MHz
Operating Voltage	5.0V
Operating Mode	Single-chip mode
Endian	Instructions: Little Endian 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 • Compiler option: -isa=rxv1
Emulator	Renesas Electronics E2 lite emulator (Fine interface). The emulator such as E2, E2 lite, etc. also can be used.
Middleware	RX13T Data Pre Processing Lib
e-AI Development Environment	Renesas Electronics e-AI Translator V1.0.2
Evaluation Board	Renesas Electronics • 24V Motor Control Evaluation System for RX23T (RTK0EM0006S01212BJ) • RX13T CPU card (RTK0EMAXA10C00000BJ)

3.2 Operations Overview

3.2.1 Workbench Setup of demo

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

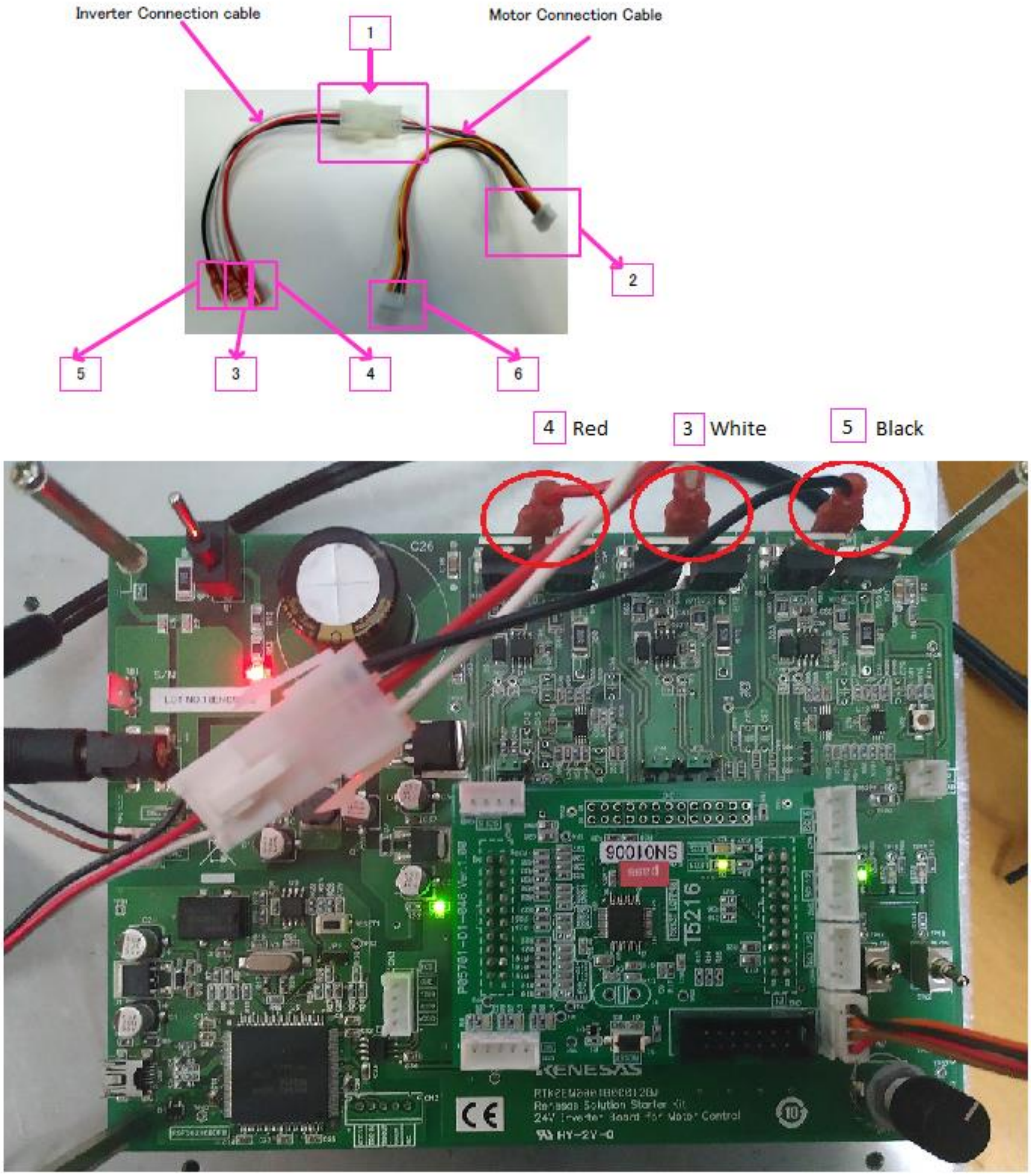


Figure 6 Motor Connection to 24V inverter board

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 RX13T CPU card**

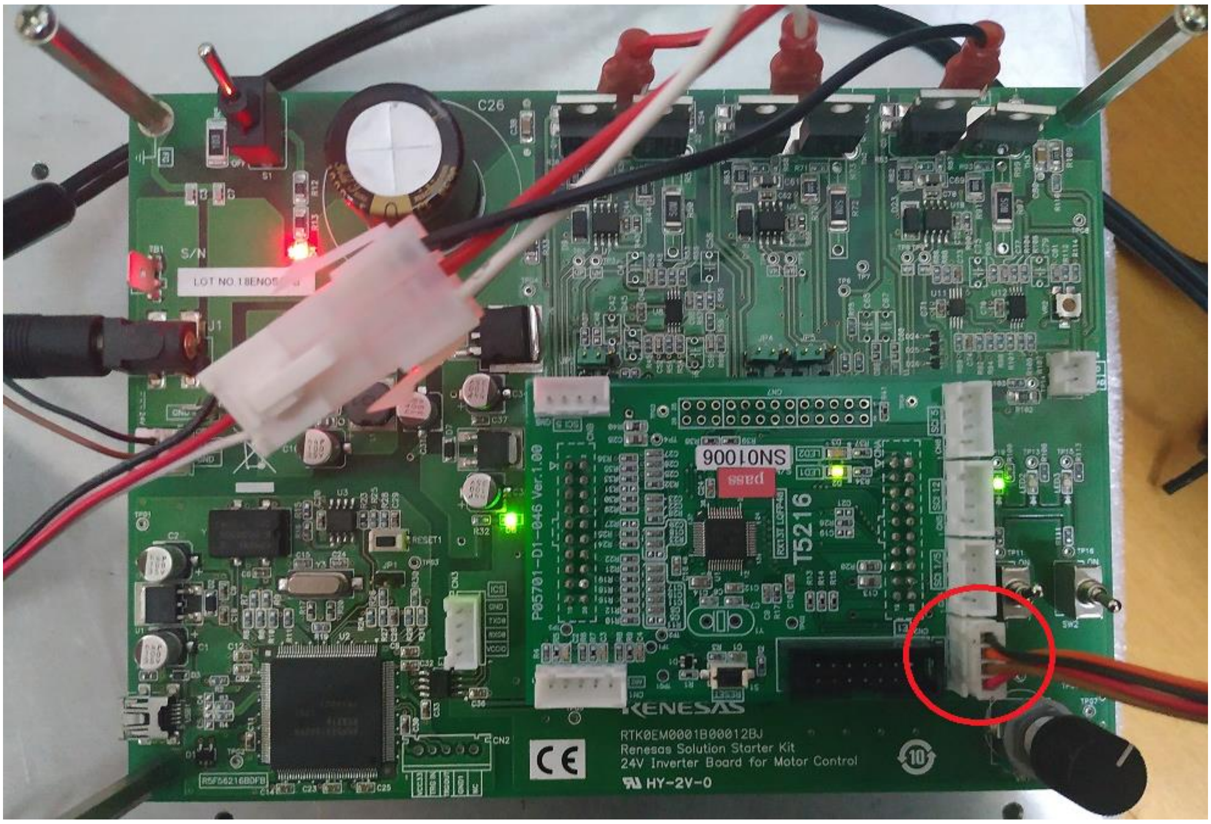


Figure 7 SCI connection of RX13T

SC1 of RX13T is used for serial communication with the Data Collection Tool.

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

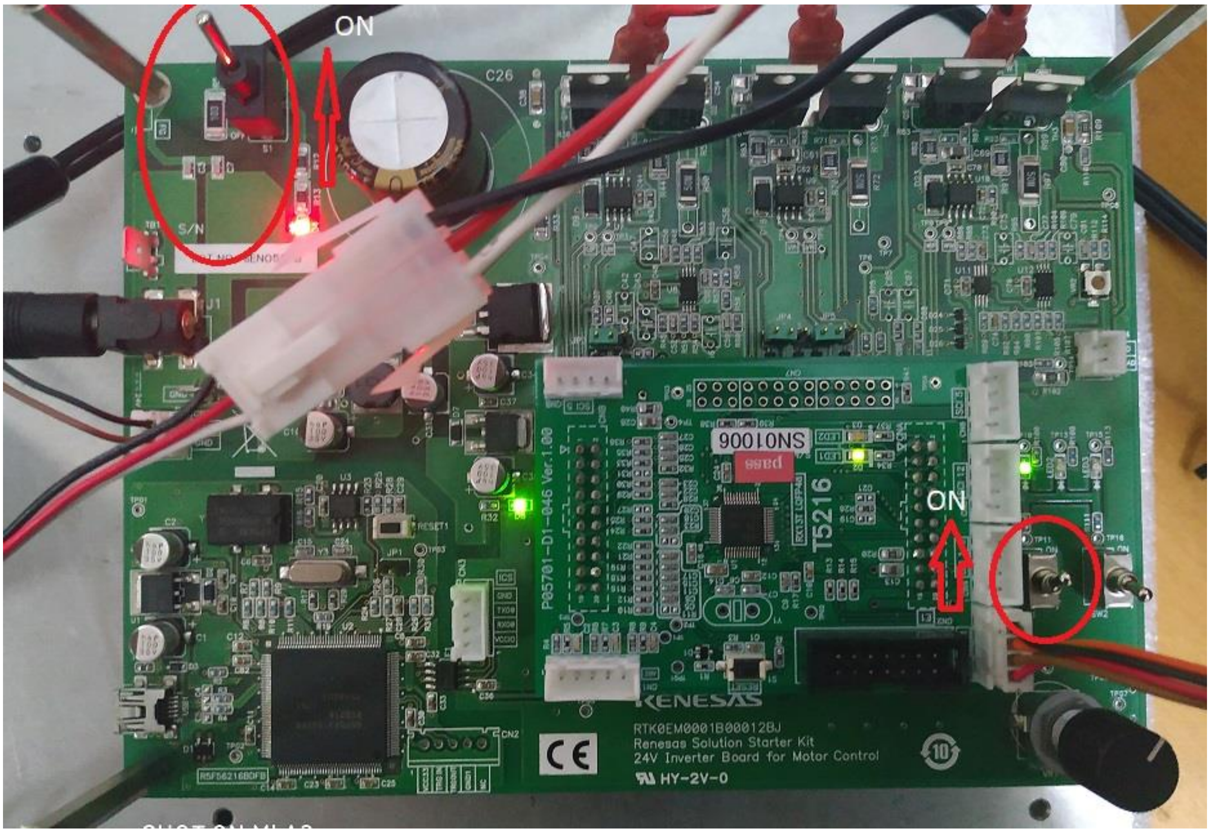


Figure 8 Switch Conditions

State	S1 switch	SW1 switch	LED1 (inverter board)	LED1 (RX13T)	LED2 (inverter board)	LED2(RX13T)
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 RX13T and the PC.

cable C232HM-EDHSL-0 pin assigned

Function	Wire color
TXD	Orange
RXD	Yellow
GND	Black

Figure 9 USB-Serial Cable

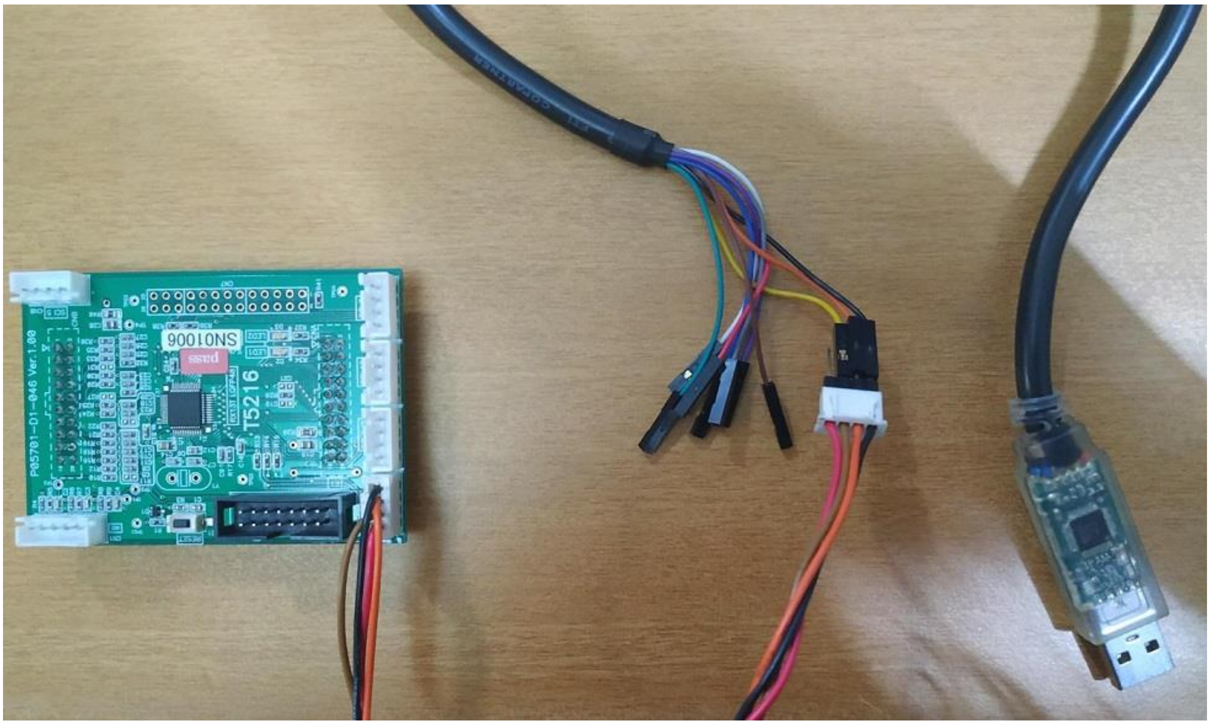


Figure 10 USB-Serial Cable connection to RX13T

As SCI1 is used for serial communication in RX13T, 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 RX13T.**

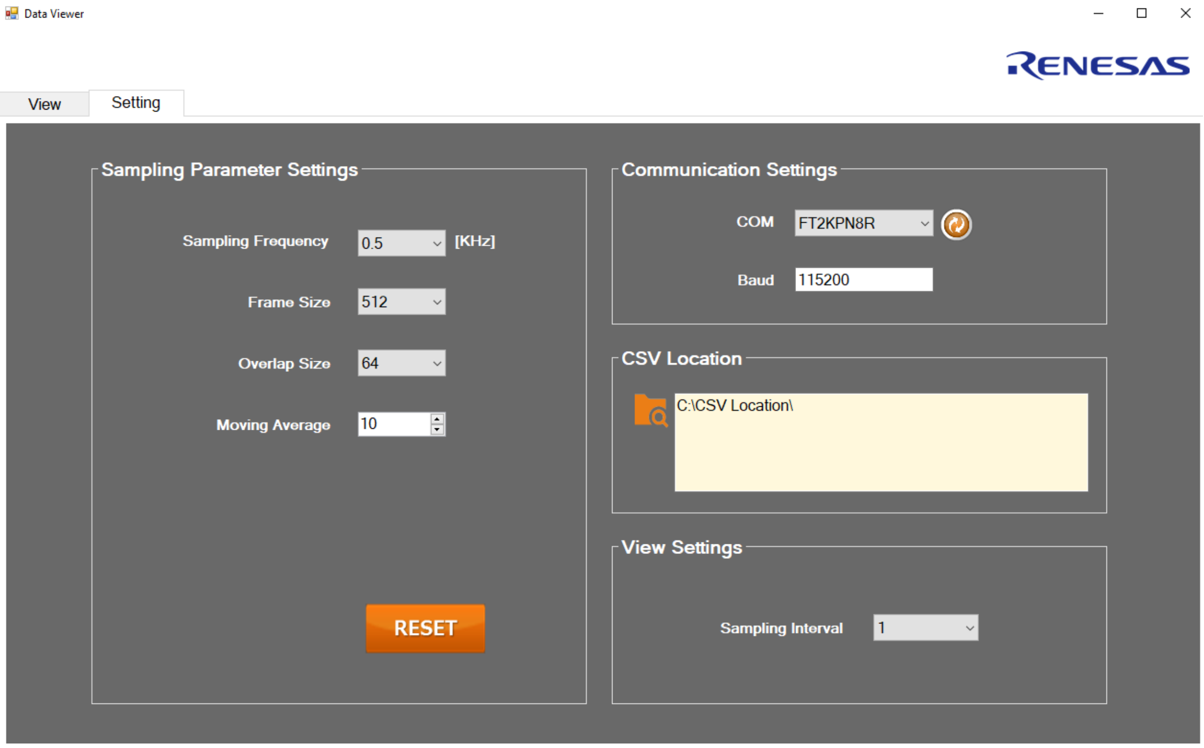


Figure 11 Data Collection Tool Parameter settings

Please select Sampling frequency as 0.5KHz, Frame Size as 512 and Baud rate as 115200.

- **Figure 12 shows the results of RX13T demo.**

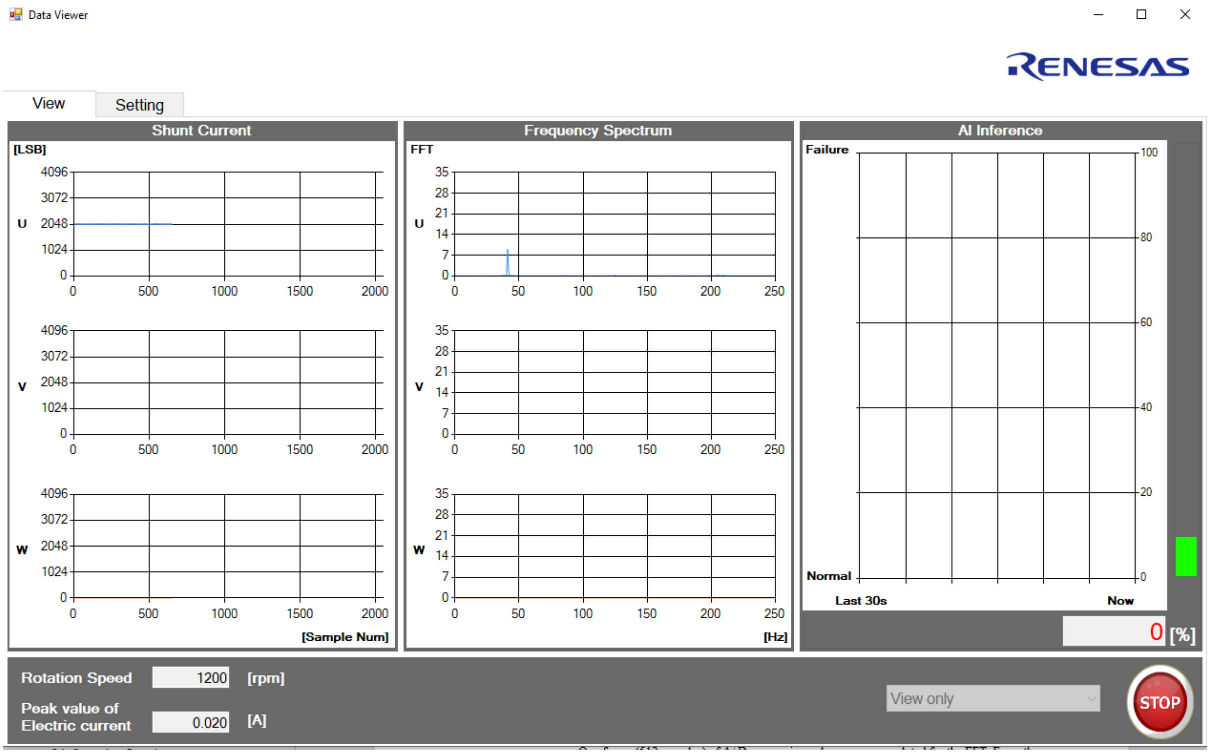


Figure 12 Data collection Tool display

3.2.3 System operation flow

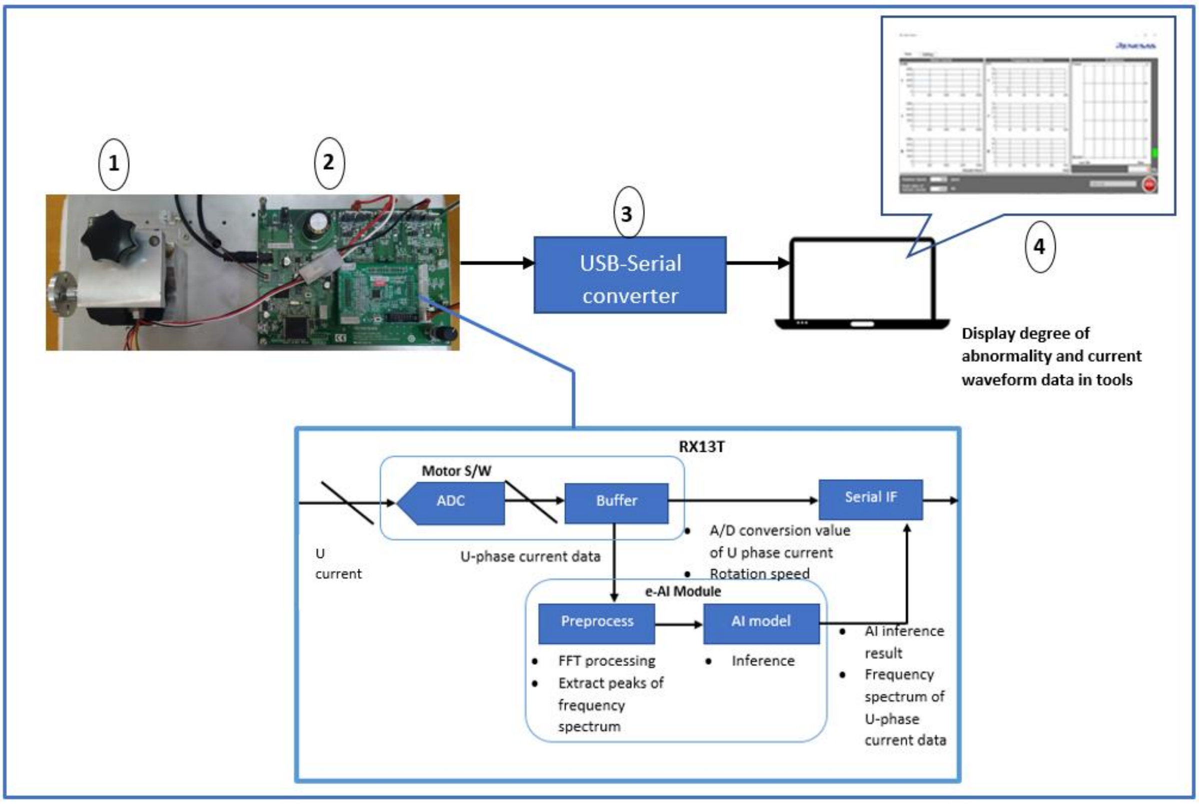


Figure 13 System Operation Flow.

Figure 13 shows the system operation flow. The numbers in the Figure 14 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 RX13T 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 (512 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 is 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 SCI1, 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 Data Collection Tool (GUI tool) run on the PC.

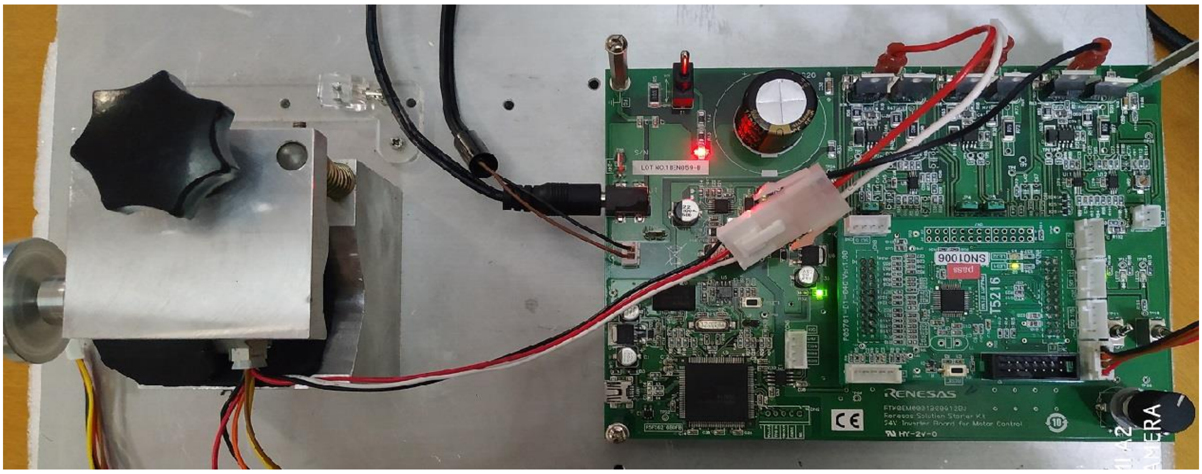


Figure 14 Enlarged Image of RX13T demo setup

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 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.

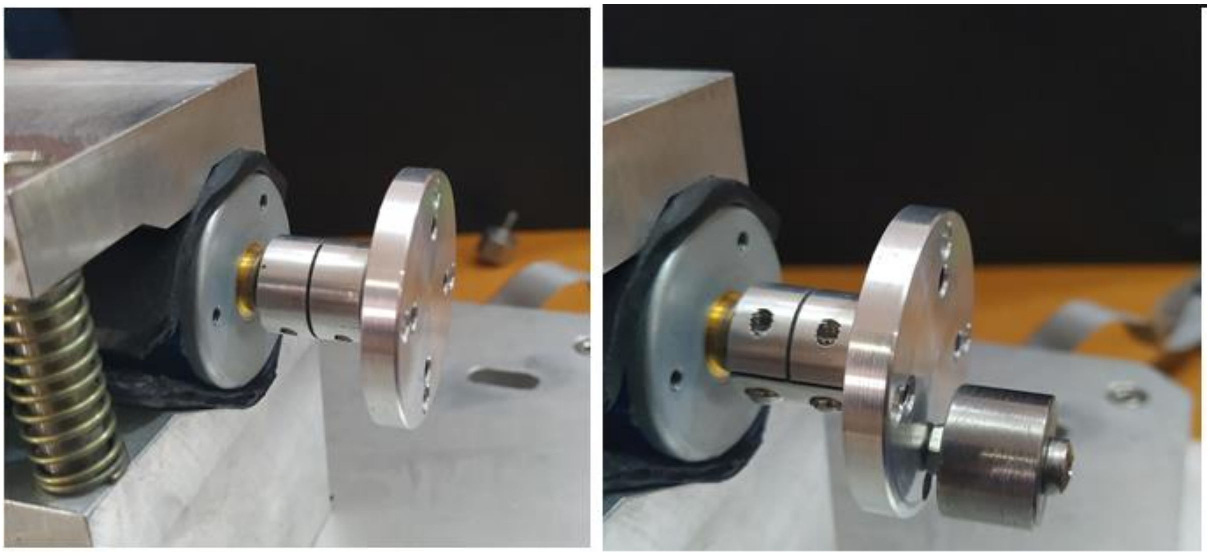


Figure 15. Normal State (left side) and Abnormal State (right side)

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.

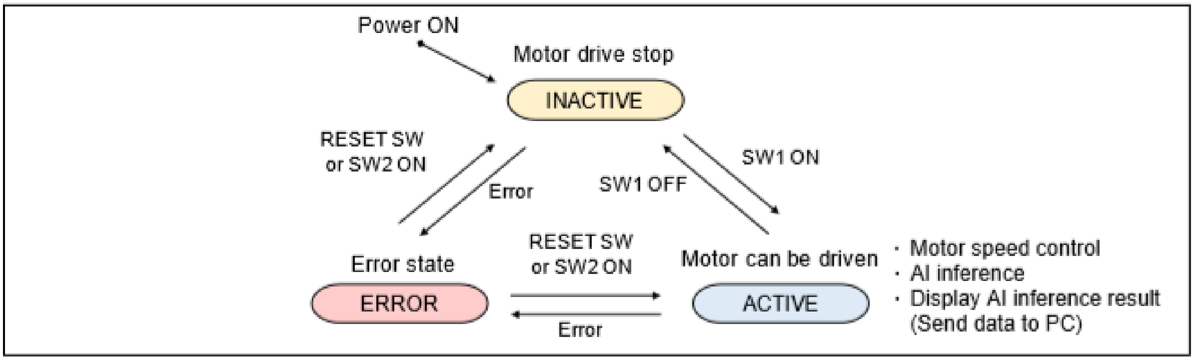


Figure 16. State Transition

(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

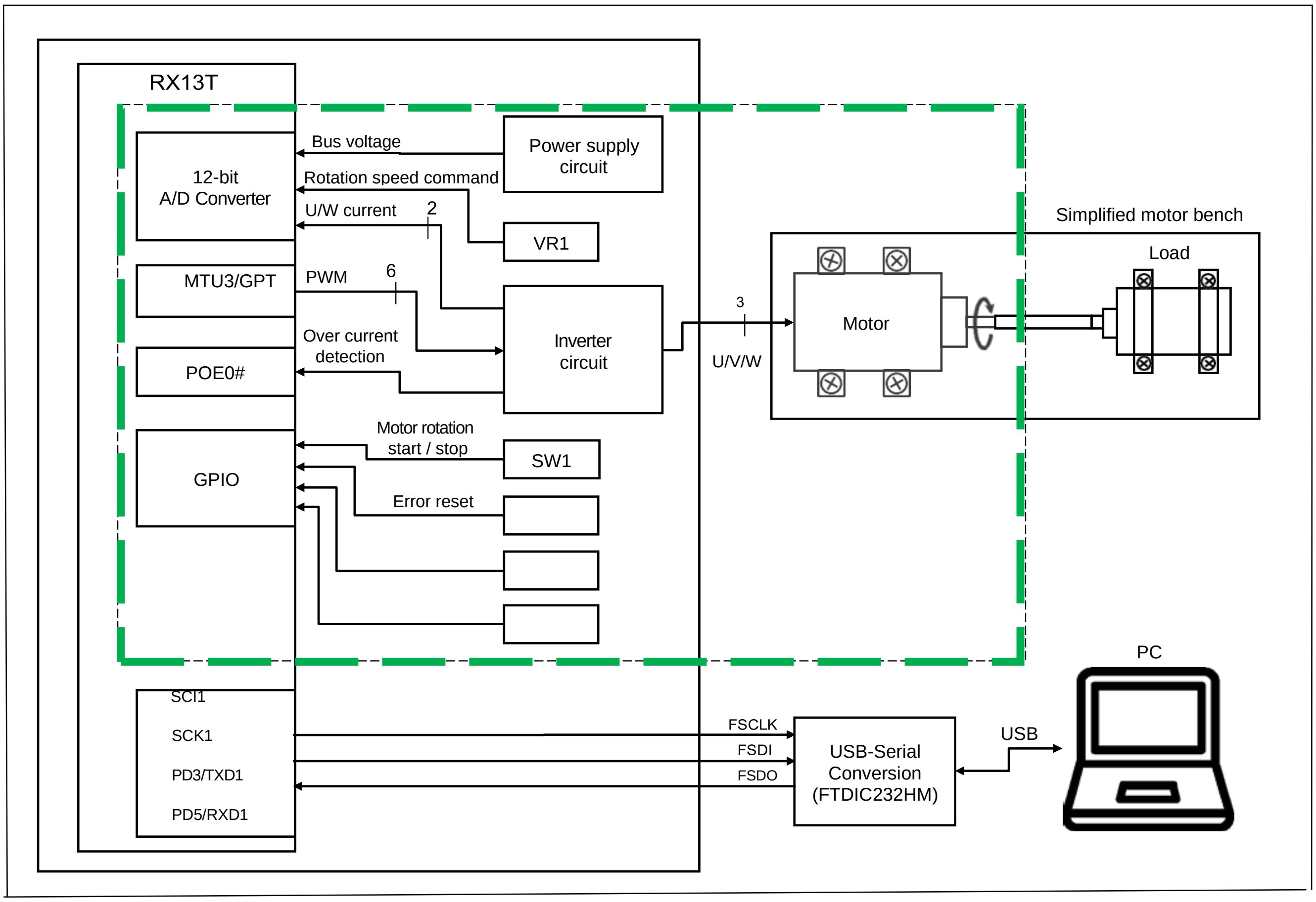


Figure 17 Hardware diagram

4. MCU Software Explanation

4.1 Software Configuration

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

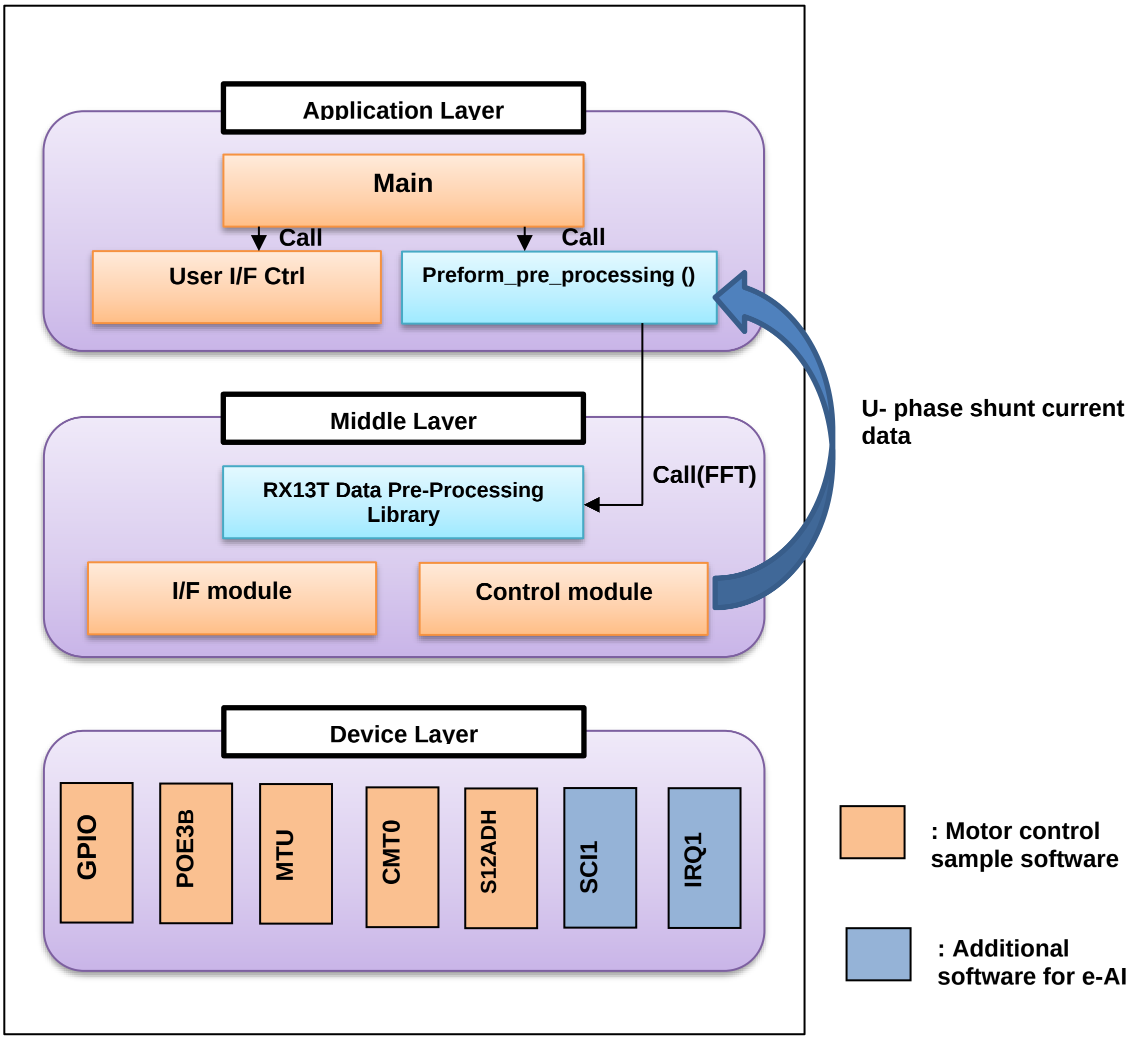


Figure 18. Software Configuration Overview

4.2 Directory Configuration

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

```
RX13T_MRSSK_SPM_LESS_FOC_CSP_RV040
│   .cproject
│   .project
│   RX13T_MRSSK_SPM_LESS_FOC_CSP_RV040 Debug.launch
│   RX13T_MRSSK_SPM_LESS_FOC_CSP_RV040 HardwareDebug.launch
│   RX13T_MRSSK_SPM_LESS_FOC_CSP_RV040.x.launch
│
├─.settings
│   │   com.renesas.cdt.managedbuild.renesas.ccrx.prefs
│   │   Dependency_Scan_Preferences.prefs
│   │   e2studio_project.prefs
│   │   renesasPGModel.xml
│   │
│   └─CodeGenerator
│           cgprojectDatas.datas
│
├─application
│   ├─main
│   │       main.c
│   │       main.h
│   │
│   ├─pre_processing_engine
│   │   │   eai_solution_config.h
│   │   │   Pre_defined.h
│   │   │   pre_processing.c
│   │   │   pre_processing.h
│   │   │   Validation.h
│   │   │
│   │   └─Translator
│   │           checker_log_output.txt
│   │           dnn_compute.c
│   │           input_image_0.h
│   │           layer_graph.h
│   │           layer_shapes.h
│   │           network.c
│   │           network_description.txt
│   │           Typedef.h
│   │           weights.h
│   │
│   └─user_interface
│       ├─board
│       │       r_mtr_board.c
│       │       r_mtr_board.h
│       │
│       └─ics
│               ICS2_RX13T.h
│               ICS2_RX13T.lib
│               RX13T_MRSSK_SPM_LESS_FOC_CSP_RV040.map
│               RX13T_MRSSK_SPM_LESS_FOC_E2S_RV101.rmt
│               r_mtr_ics.c
│               r_mtr_ics.h
│
├─config
│       r_mtr_config.h
│       r_mtr_control_parameter.h
│       r_mtr_inverter_parameter.h
│       r_mtr_motor_parameter.h
│
├─driver
│   ├─inverter
│   │       r_mtr_ctrl_mrssk.c
│   │       r_mtr_ctrl_mrssk.h
│   │
│   └─mcu
│       │   r_mtr_ctrl_mcu.h
│       │   r_mtr_ctrl_rx13t.c
│       │   r_mtr_ctrl_rx13t.h
│       │   r_mtr_interrupt.c
│       │   SCI.c
│       │   SCI.h
│       │
│       └─auto_generation
│               dbsct.c
│               hwsetup.c
│               intprg.c
│               iodefine.h
│               resetprg.c
│               sbrk.c
│               sbrk.h
│               stacksct.h
│               typedefine.h
│               vect.h
│               vecttbl.c
│
└─middle
    ├─common
    │       r_mtr_common.h
    │       r_mtr_filter.c
    │       r_mtr_filter.h
    │       r_mtr_fluxwkn.h
    │       r_mtr_fluxwkn.obj
    │       r_mtr_mod.c
    │       r_mtr_mod.h
    │       r_mtr_pi_control.c
    │       r_mtr_pi_control.h
    │       r_mtr_statemachine.c
    │       r_mtr_statemachine.h
    │       r_mtr_transform.c
    │       r_mtr_transform.h
    │       r_mtr_volt_err_comp.h
    │       r_mtr_volt_err_comp.obj
    │
    ├─control
    │       r_mtr_bemf_observer.h
    │       r_mtr_bemf_observer.obj
    │       r_mtr_ctrl_gain_calc.obj
    │       r_mtr_foc_action.c
    │       r_mtr_foc_control_less_speed.c
    │       r_mtr_foc_control_less_speed.h
    │       r_mtr_foc_current.c
    │       r_mtr_foc_current.h
    │       r_mtr_foc_speed.c
    │       r_mtr_foc_speed.h
    │       r_mtr_interrupt_1ms.c
    │       r_mtr_interrupt_carrier.c
    │       r_mtr_opl2less.h
    │       r_mtr_opl2less.obj
    │       r_mtr_opl_damp_ctrl.h
    │       r_mtr_opl_damp_ctrl.obj
    │       r_mtr_parameter.h
    │
    └─interface
            r_mtr_driver_access.c
            r_mtr_driver_access.h
```

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 • Inverter bus line voltage measurement • Rotation speed command value input
CMT0	1ms interval timer
MTU3	Complementary PWM output
GPIO	• Switch (SW1, SW2) input • LED (LED1, LED2) ON/OFF control

SCI1	Transfers data to send data register based on SCI1 transfer request

4.3.2 Interruptions

Table provides a list of interrupt process.

Interrupt request generation source (peripheral module)	**Name (peripheral module name)**	**Interrupt priority level (IPR)**	**Description**
Group Interrupt 1 （POE3） （SCI1）	GROUPBL1 （OEI4） （TEI1,ERI1)	15 (highest)	• Motor interrupt processing (overcurrent detected) • • Data send/receive to/from PC - SCI1 send complete interrupt (TEI1) -SCI1 receive error interrupt (ERI1)
PERIA（MTU3）	INTA209 (TCIV4)	12	Motor interrupt processing(1ms)
SCI1	RXI1	12	Data receive from PC
CMT0	CMI0	11	Motor interrupt processing
SCI1	TXI1	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.

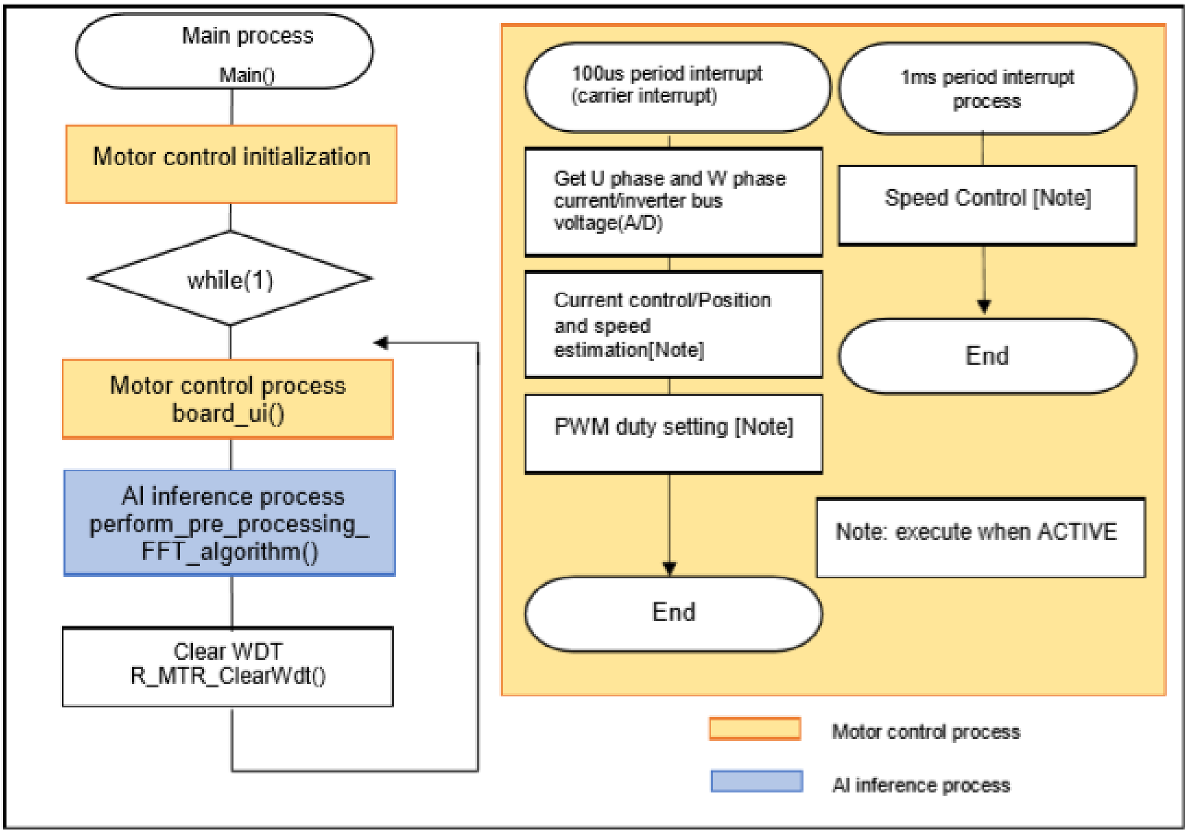


Figure 20. Main Processing Flow Chart

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 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. pstInOutData - Pointer for input/output buffer to store the FFT computation.

The following shows the FFT source code-

```
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].fIma
g,2)))*2.0;
}
```

4.7 AI Inference Processing

4.7.1 Flowchart

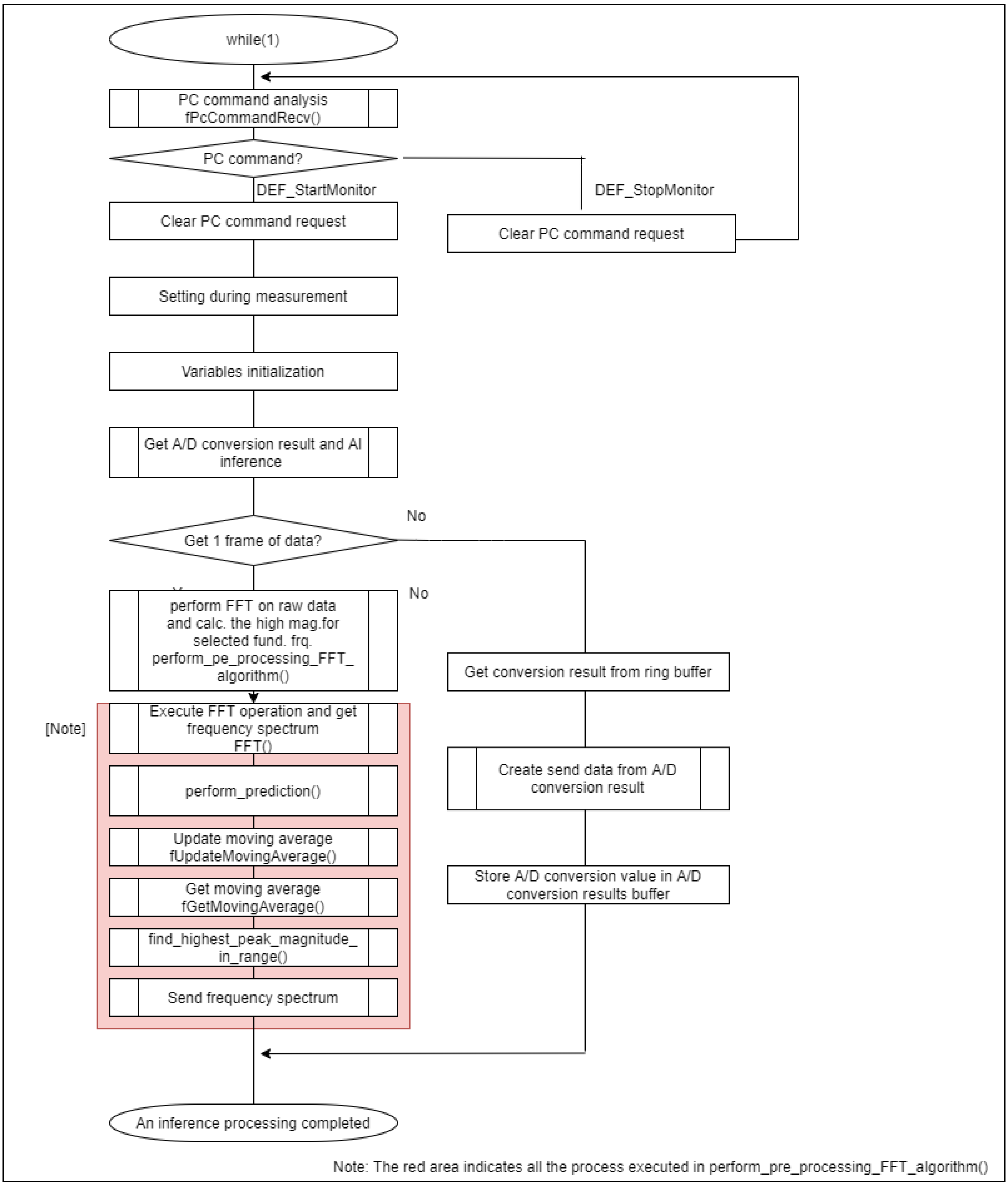


Figure 21. AI Inference Processing Flowchart

4.7.2 Data Flow

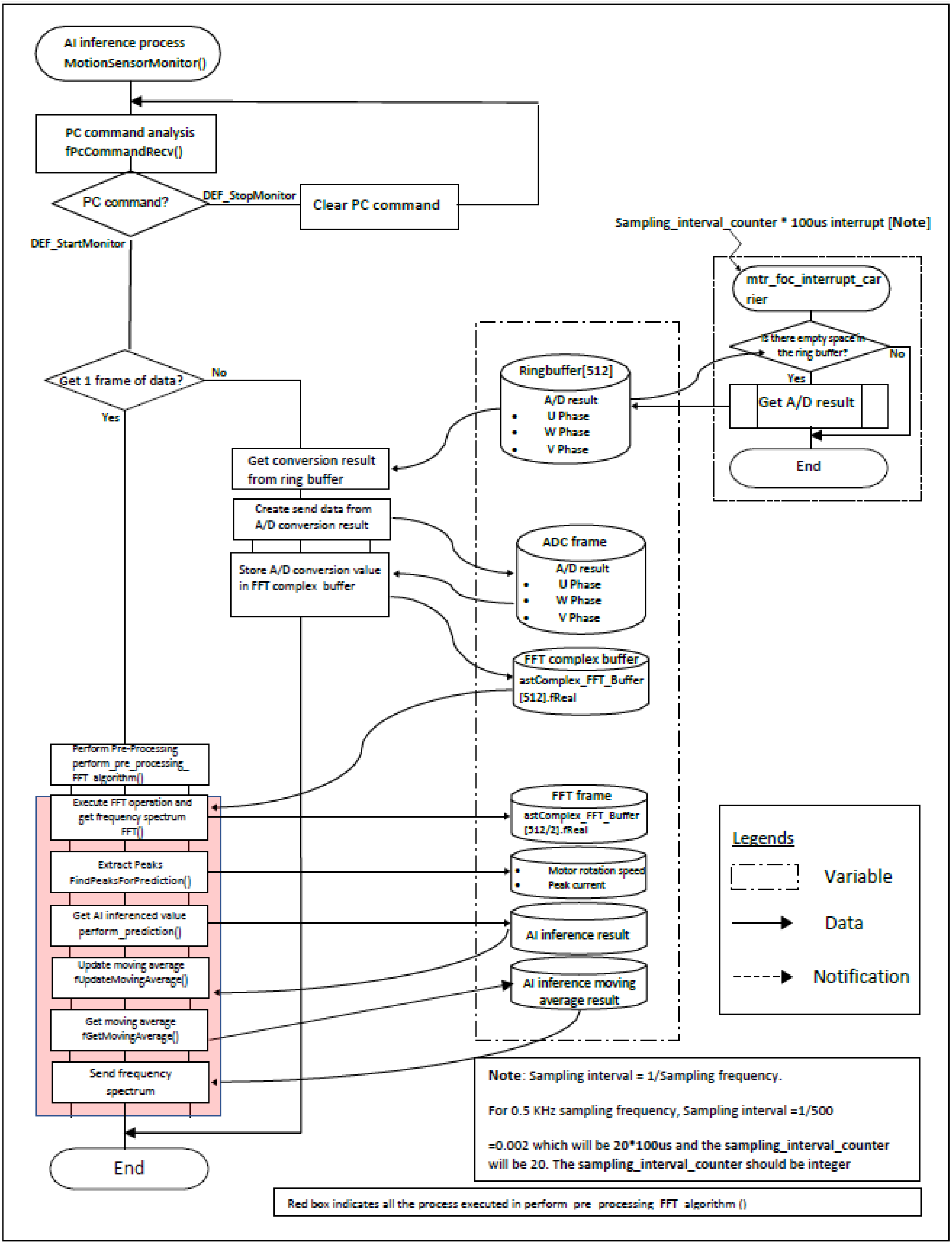


Figure 22 AI inference processing data flow

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, axis deviation is defined as abnormal.

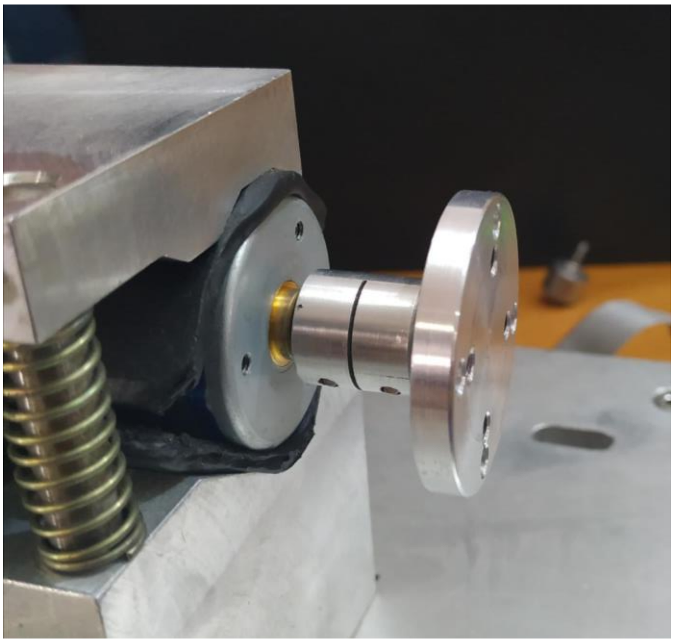


Figure 23. Normal State

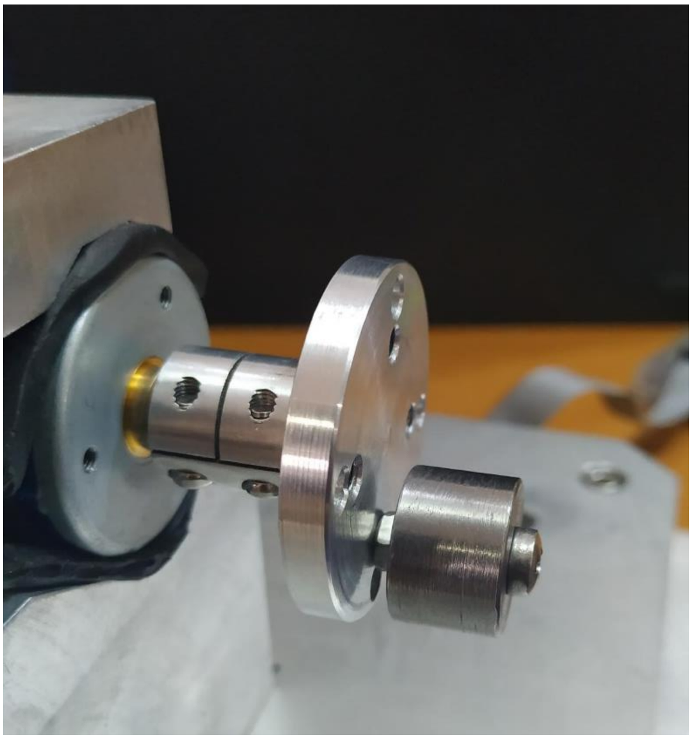


Figure 24. Abnormal State

Figure 25 shows the AI model configuration

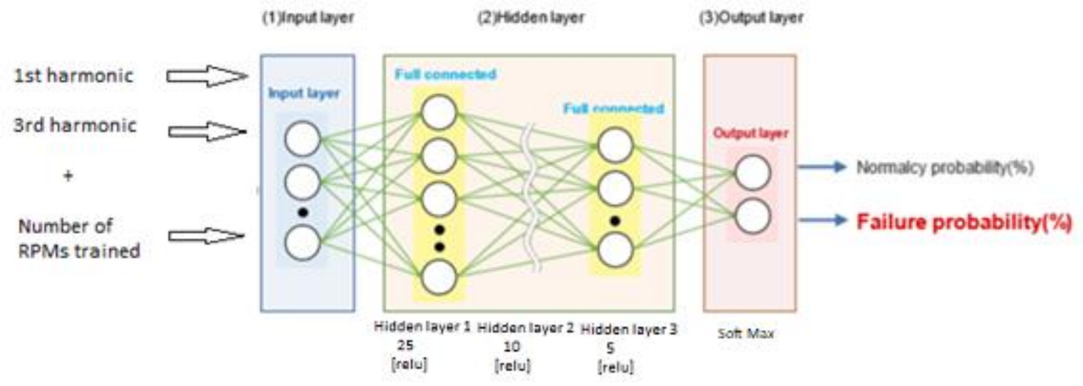


Figure 25. AI Model Configuration

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 section 6.2.2.2
- Choose the training data set path as data collected 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).

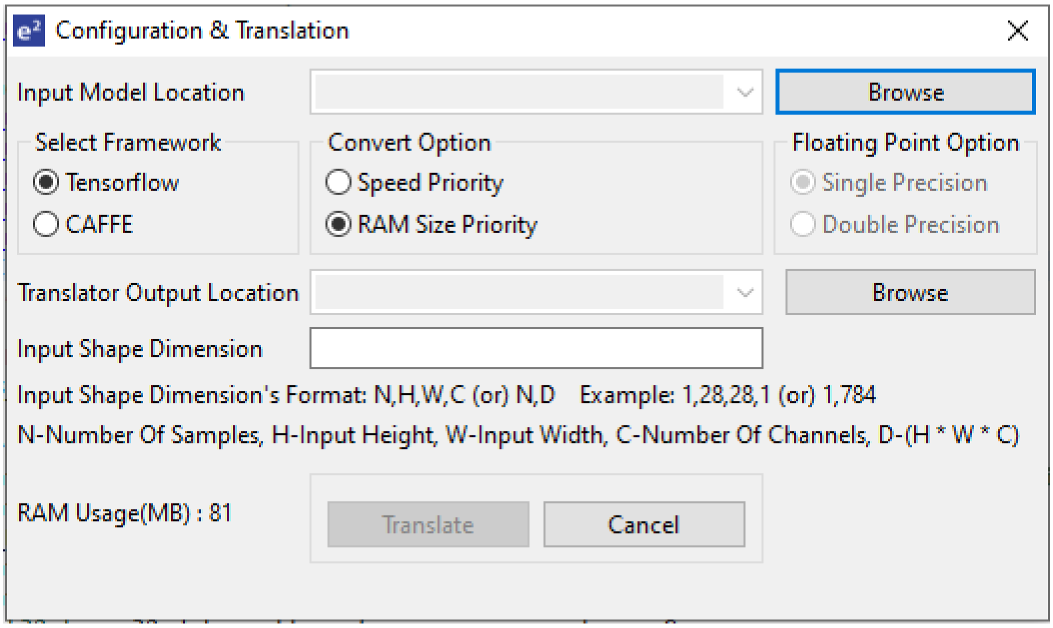


Figure 27 Translating AI Model

- 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 RX13T.

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.

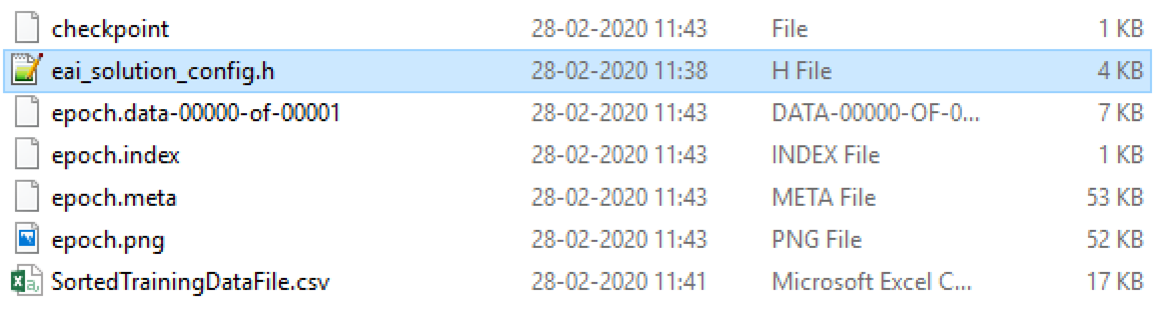


Figure 28 Parameter header file from training tool

- Move the eai_solution_config.h file to e2 studio project\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 RX13T MCU via a serial communication and to display those results.

An overview of the Data collection tool operations is as follows.

- Control display start/stop
- Display data sent from RX13T
    - U-phase shunt current waveform data
    - Waveform data after U-phase shunt current data is FFT processed
    - AI inference result
        - Moving average waveform
        - Actual value bar (0~100% displayed in increments of 10)
        - Numerical value
    - Peak current value
    - Motor rotation speed

6.1.1 Function Explanations

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

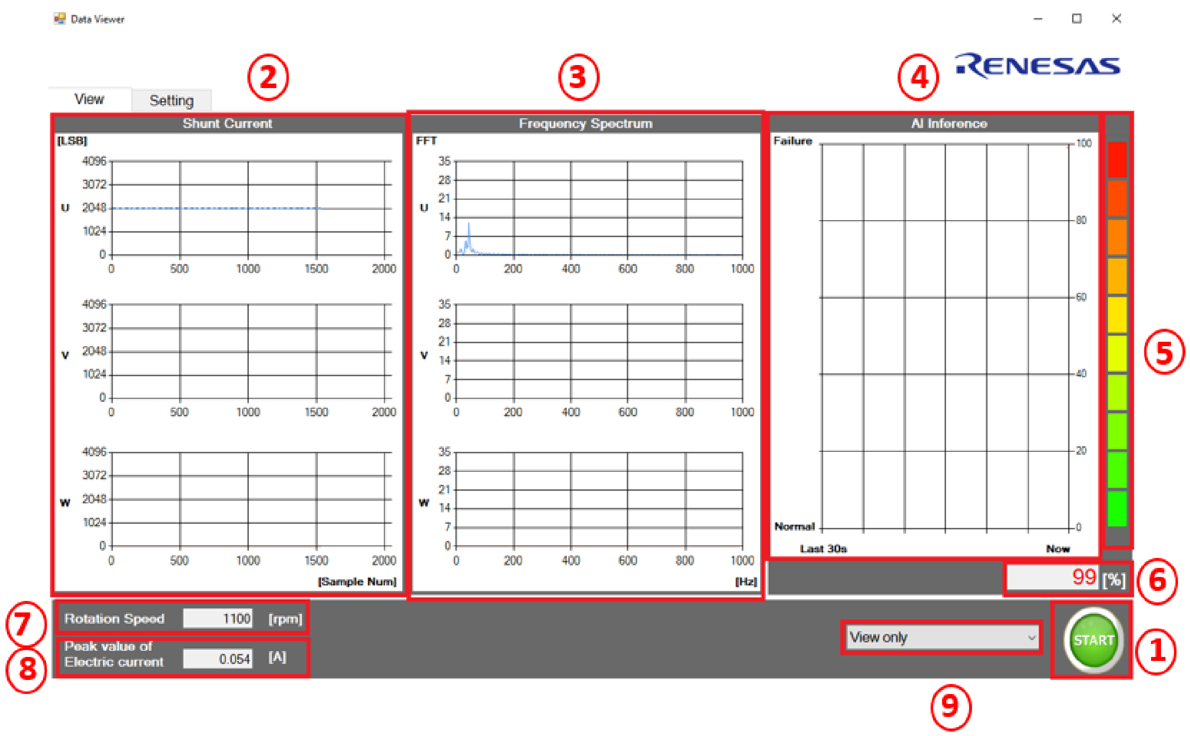


Figure 29. View Tab Display Specifications

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:
     - 'Data Send Request Commands' are sent from the PC to RX13T, and data is sent from RX13T to the PC.
     - Received data is displayed in real time.
   - When the STOP button is pushed:
     - 'Data Send Stop Command' is sent from the PC to RX13T 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 frequency spectrum is displayed while V and W phase current will be 0 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 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
     - Only monitors various data.
   - Save to CSV (divided)
     - This option is not supported.

- Save to CSV (combined)
    - This option is not supported.
- Save to CSV (RPM Combined)
    - 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.

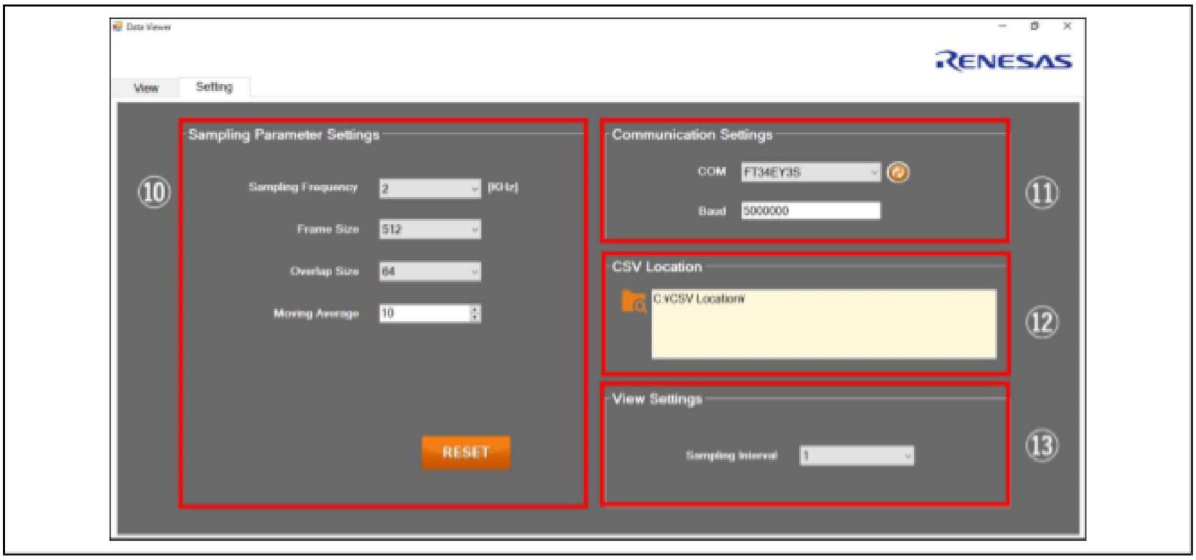


Figure 30. Setting Tab Display Specifications

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: - Select Sampling Frequency as 0.5 for RX13T

- **Frame Size -** Specifies the FFT frame size (128/256/512/1024, default: 512).
- **Note:** - For RX13T select Frame Size as “512” 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 RX13T 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.

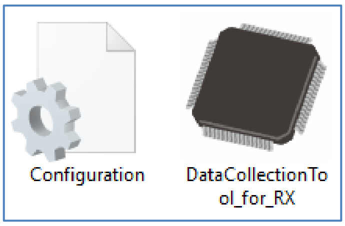


Figure 31. DataCollectionTool.exe file

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.

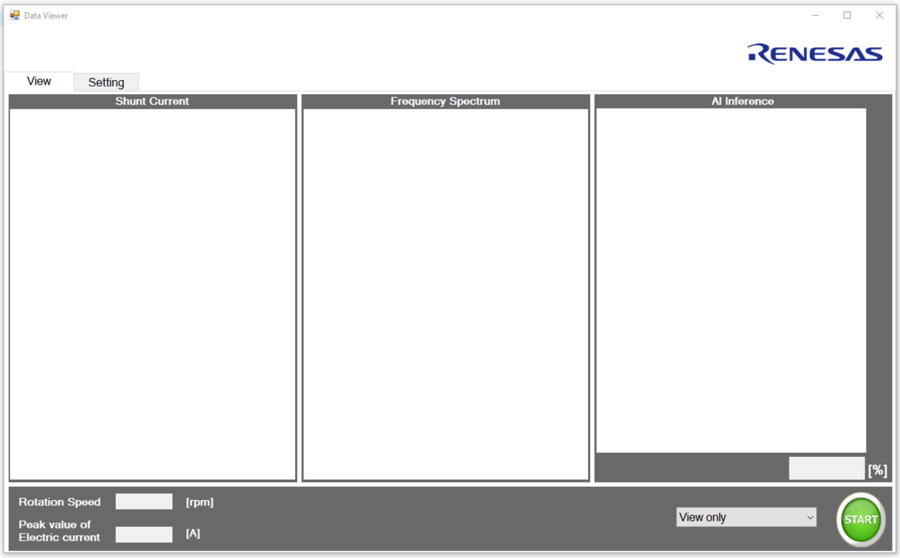


Figure 33. Initial Startup Screen

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.

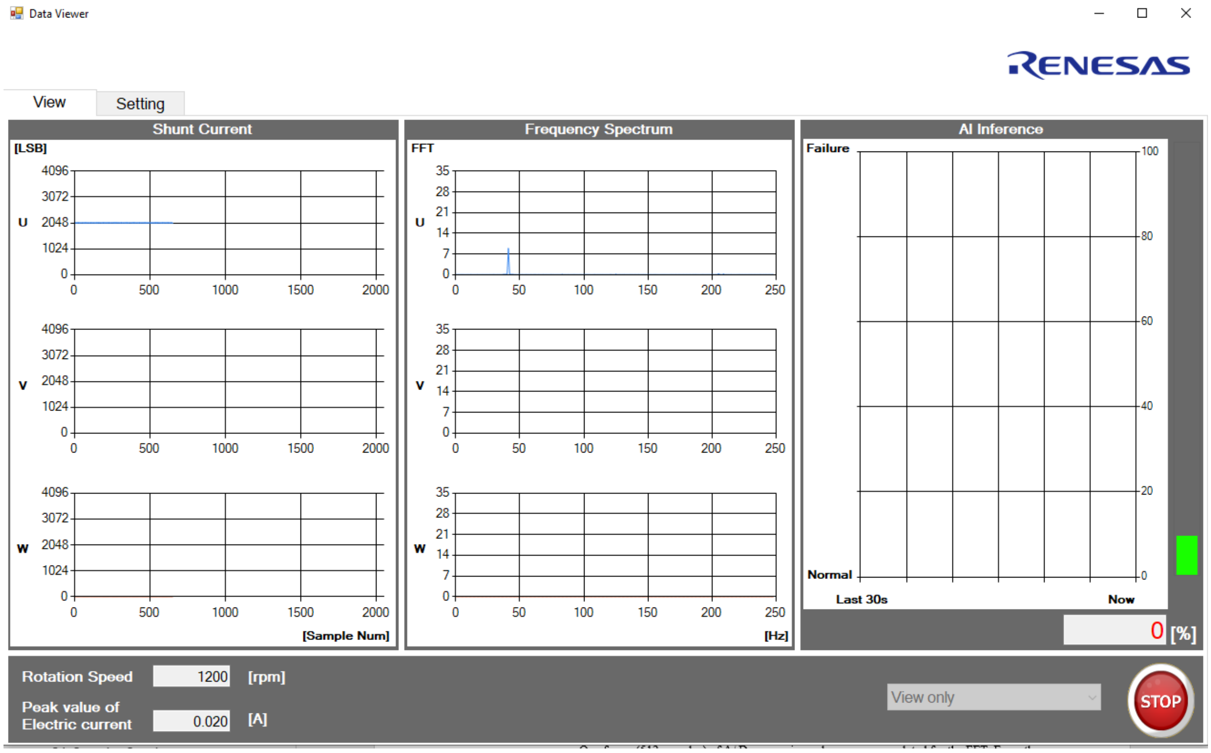


Figure 35. Screen During Data Acquisition

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 file along with RPM information while viewing the data. Figure 37 shows the file created for saving the output data.

Figure 37. File Created when” Save to CSV (RPM 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 Function 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. **1.** Training: This mode will train the Model with Training data set, performs pre-Processing and creates Model with parameter header file.
2. **2.** Testing: This mode will test the created model with the input data set.
3. **3.** Training Data Set: This will the path of collected data from Data collection tool for training the model.
4. **4.** Output AI Model: this will the path for the created model.
5. **5.** Loading and Preprocessing: This is the progress bar for pre-processing.
6. **6.** Start: This is the button to start the training or testing.

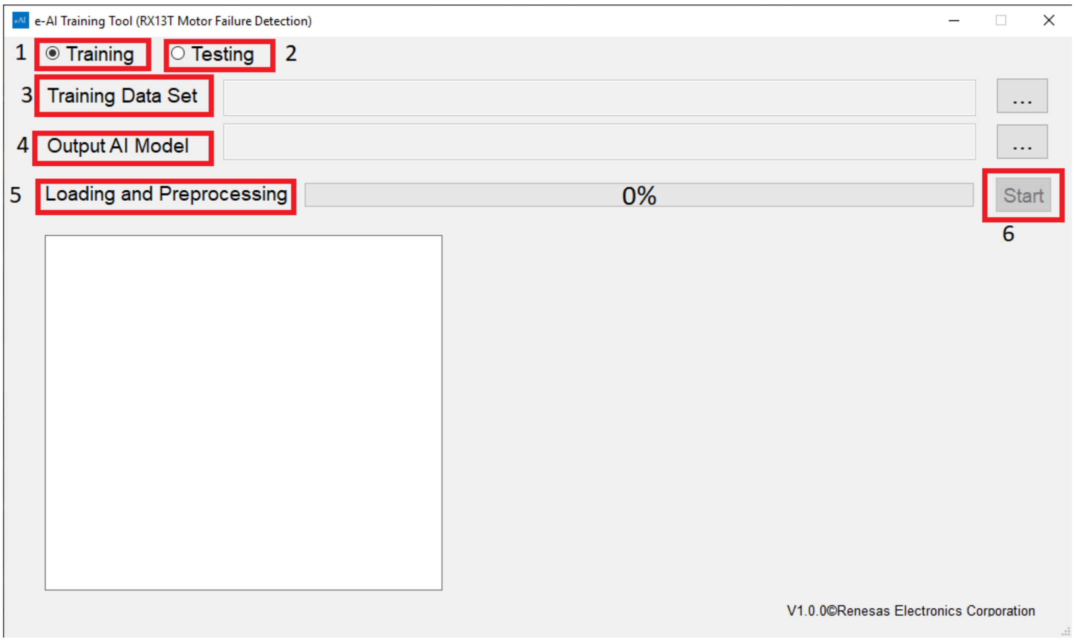


Figure 38 Training tool functionality

6.2.2 Operations

6.2.2.1 Run the exe

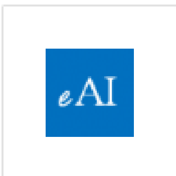


Figure 39 Training tool exe

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.

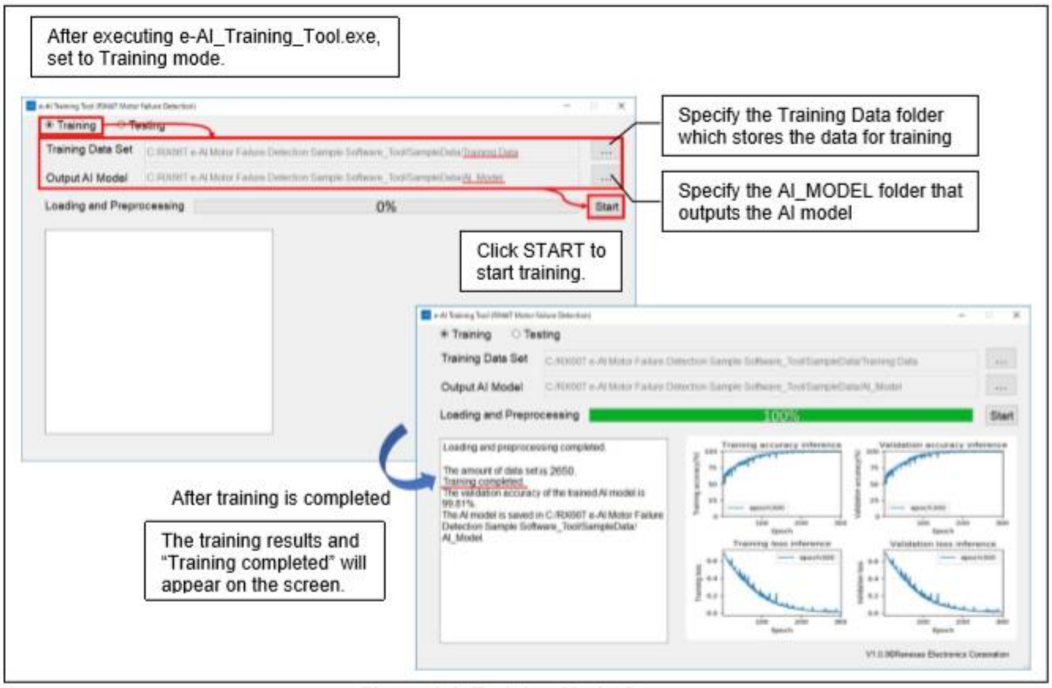


Figure 40. Training Tool (Training)

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.

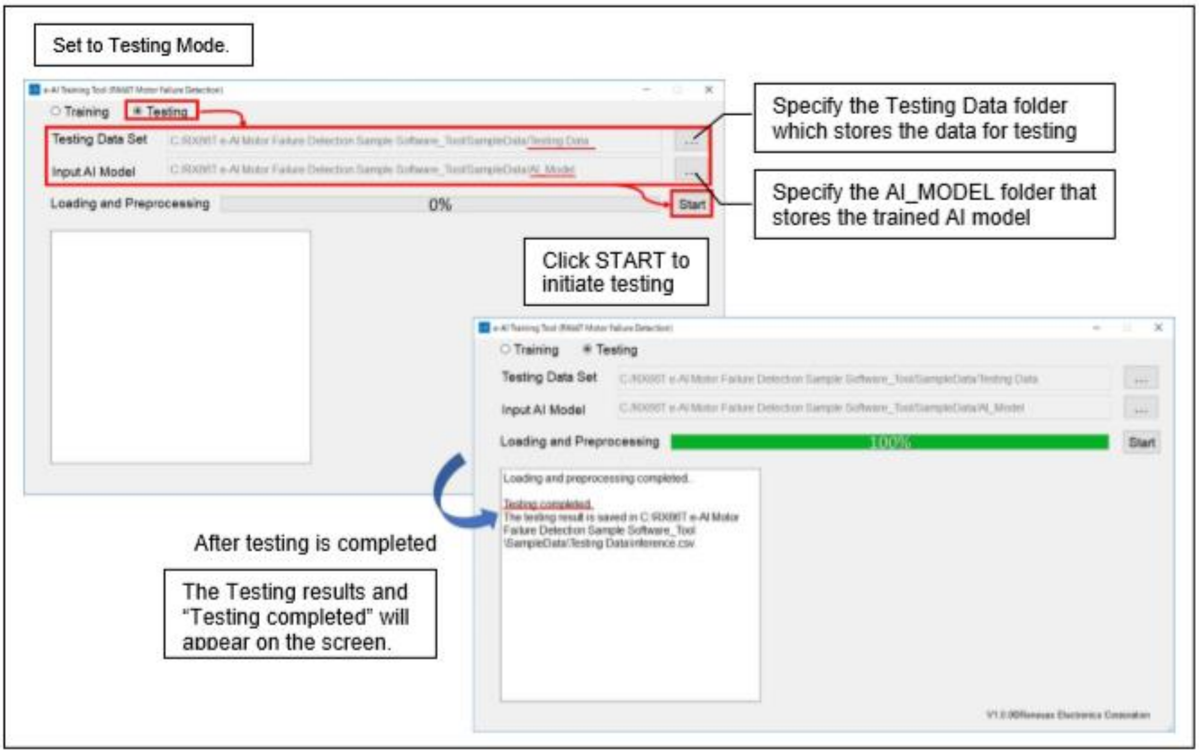
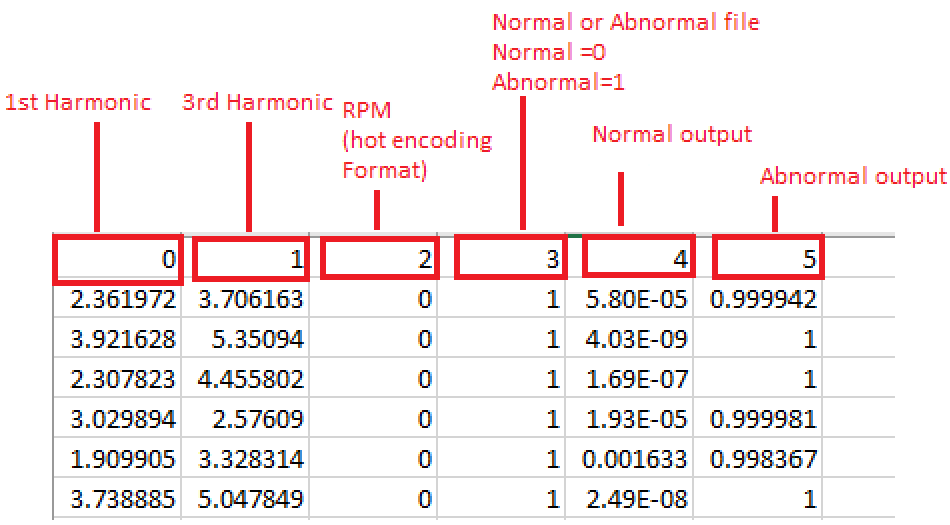


Figure 42. Training Tool (Testing)

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



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] RX13T Group User’s Manual: Hardware (R01UH0822)
[2] RX13T Sensorless Vector Control for Permanent Magnetic Synchronous Motor (Implementation) (R01AN4967)
[3] Renesas Solution Starter Kit 24V Motor Control Evaluation System for RX23T User’s Manual (R20UT3697)
[4] RX13T CPU Card User’s Manual (R12UZ0051)
[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
RAM	12KB
ROM	128KB

2. CPU Load

Item	Processing Time
FFT process (FFT()) (*)	92ms
e-AI inference process (Perform_prediction()) (*)	2ms

3. Smart Configurator Settings

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=32MHz)
SCKCR(PCLKA[3:0])	x1/2 (PCLKA=80MHz)
SCKCR(PCLKB[3:0])	x1/4 (PCLKB=32MHz)
SCKCR(PCLKD[3:0])	x1/4 (PCLKD=40MHz)
SCKCR(BCK[3:0])	x1/4 (BCLK=40MHz)

ii. SCI1

Use SCI1 to carry out serial communication between RX13T and the pc. FTDI's USB – serial converter cable serves as the serial communication interface, and transfers are carried out in the MPSSE(*1) mode. To enable this, the SCI1 must be set to asynchronous mode, and clock output from the SCK8 pin. Also, use the CTS#8 pin for flow control.

*1: Multi-Protocol Synchronous Serial Engine

Table lists the settings for SCI1.

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
SCK1 pin function	Clock output

Send data processing	Processed in interrupt service routine
Receive data processing	Processed in interrupt service routine
Receive error interrupt enable (ERI5)	Enabled
Callback function setting	Send complete, receive complete, receive error
Pins used	RXD1 - PD5 TXD1 - PD3

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_SCI1_user.c(output by smart configurator)

Target function : R_Config_SCI1_Create_UserInit function (created by smart configuration)

```
void R_Config_SCI1_Create_UserInit(void)
{
    /* Start user code for user init. Do not edit comment generated here */
    /*Set 9th bit to “0”*/
    SCI1.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
SCI1_init()	Initialize SCI1
R_MTR_ExecEvent()	Execute event of FOC event
setpsw_i()	Interrupt enable
R_MTR_ChargeCapacitor()	Wait for charging capacitor
R_SCI1_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 SCI1 data as unit16 type.
SCI_TX_U8	This function transmits SCI1 data as unit8 type.
fPcCommandRecv()	Analyzes PC Command
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 SCI1 data as unit32 type
SCI_TX_U8	This function transmits SCI1 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 1st and 3rd 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**
SCI1_init()	This function receives SCI1 data.
SCI_RX_U16	This function receives SCI1 data.
SCI_TX_STRING(const char *pcS)	This function transmits SCI1 data as string type.
SCI_TX_U16(uint8_t *pu8_buffer)	This function transmits SCI1 data as uint16 type.
SCI_TX_U8(uint8_t *pu8_buffer)	This function transmits SCI1 data as uint8 type.
R_SCI1_Serial_Send(const uint8_t * u8TXBuf, uint16_t u16TXNum)	This function transmits SCI1 data.
R_SCI1_Serial_Receive(uint8_t * const pu8RXBuff, uint16_t u16RXNum)	This function starts SCI reception (asynchronous mode).
r_sci1_callback_receiveerror(void)	Performs processing in response to the receive error interrupts.
SCI_TX_U32(uint8_t *pu8_buffer)	This function transmits SCI1 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[]	SCI1 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

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
float	g_current	Shunt current at that instance

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_sci1_tx_address	SCI1 transmit buffer address
uint8_t	*gp_u8_sci1_rx_addres	SCI1 receive buffer address
uint16_t	g_u16_sci1_rx_count	SCI1 receive data number
uint16_t	g_u16_sci1_rx_length	SCI1 receive data length
uint8_t	g_u8_sci1_tx_done	SCI1 transmit flag
uint8_t	g_au8_rx_char[]	SCI1 receive buffer
uint8_t	g_u8_rx_flag	SCI1 receive flag
unit8_t	g_u8_cmd_received	

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

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.
   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
6. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.4.0-1 November 2017)

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/.