RX66T 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 RX66T.
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.

The software described in this application note is for reference use only. Operations are not guaranteed by Renesas Electronics. When using the software described in this application note, fully evaluate in an applicable environment before use.

Target Device

Operations with the software described in this application note have been confirmed for the following device.

- RX66T (R5F566TEADFP)

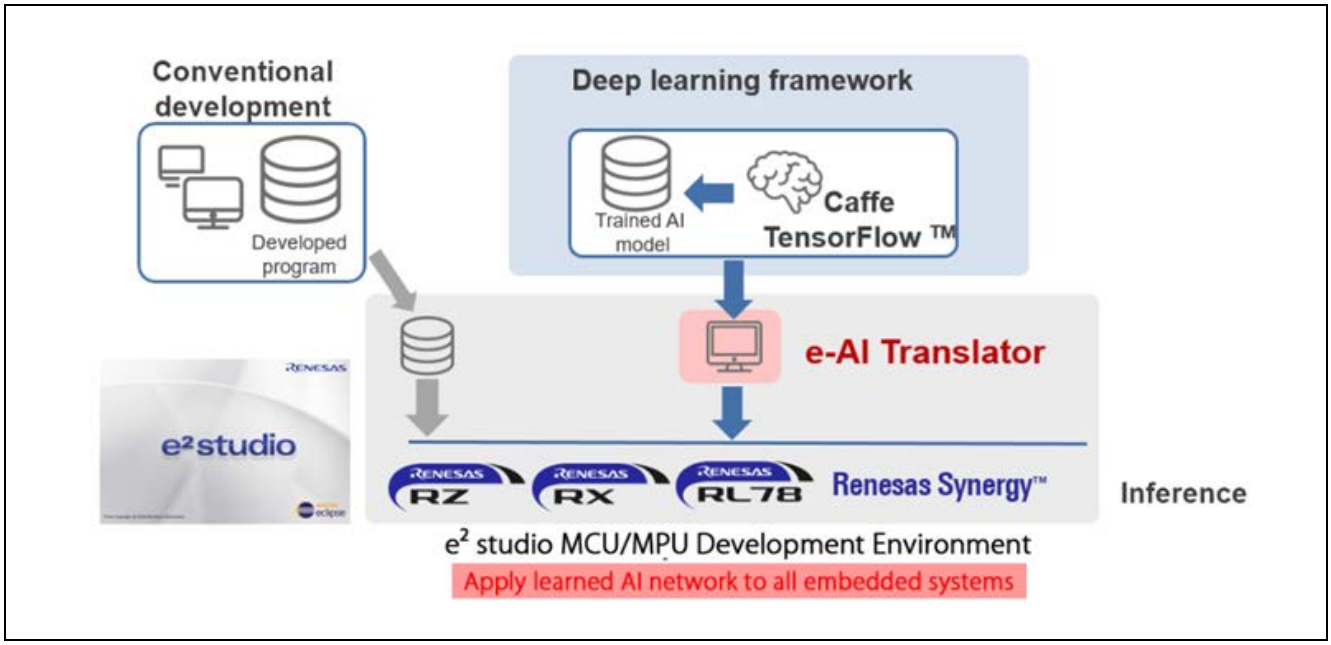
Contents

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.



2. Overview

Figure 2-1 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.

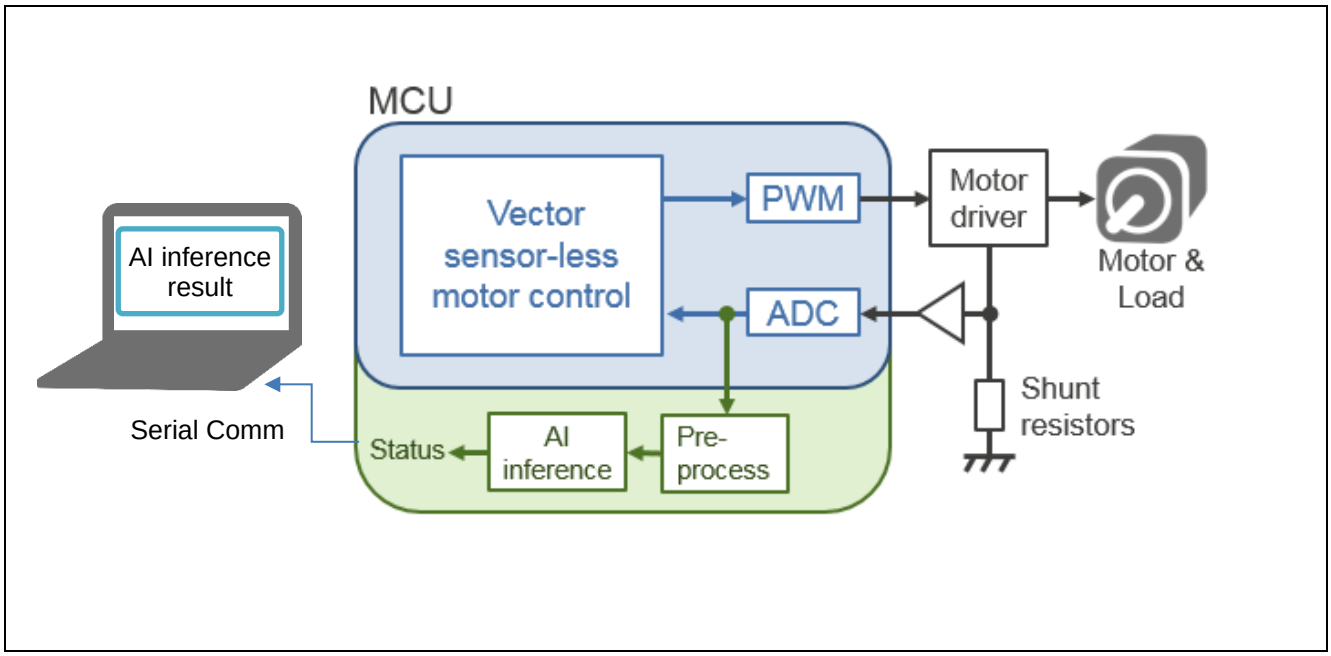


Figure 2-1. System Block Diagram

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

1. Collect A/D conversion values of the three-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 sensorless vector control method to monitor the 3 shunt current control with the A/D converter. In this system, focusing on the fact that the waveform of the 3 shunt current changes depending on the state of the motor, this 3 shunt current is used as the input of trained DNN. 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 3 shunt current3 shunt current waveform. The FFT inputs data units with coefficient of 2 as one frame, but also generates a frame to partially overlap the preceding and following frame to detect changes at frame breaks. This is a common method often referred to as "overlap analysis." In addition, in the e-AI system with limited storage area, reduction of the DNN network layer is a benefit, allowing extraction and use of areas around the input data feature point.

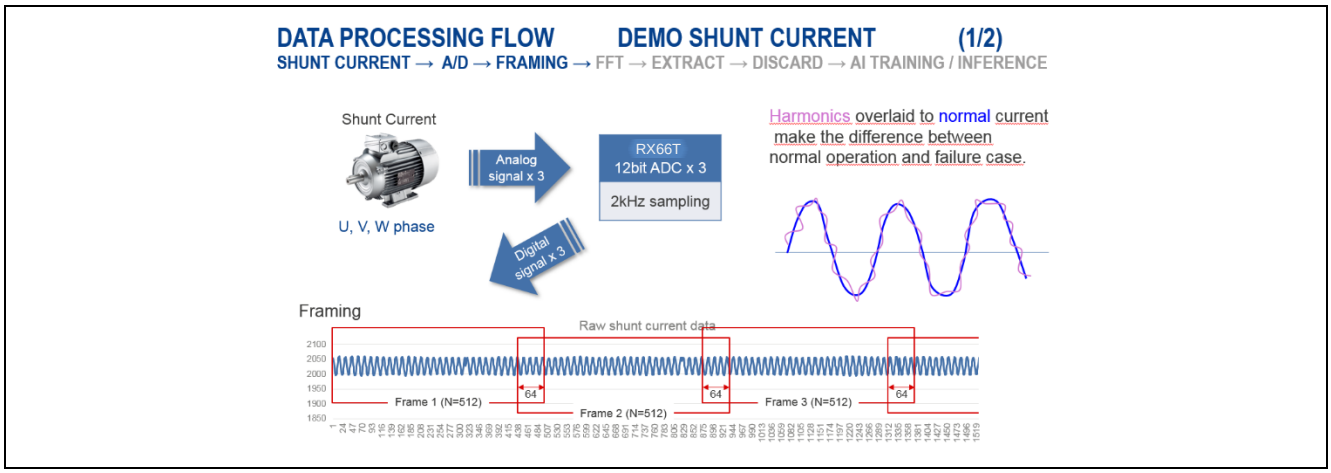


Figure 2-2. Data Processing Flow (1/2)

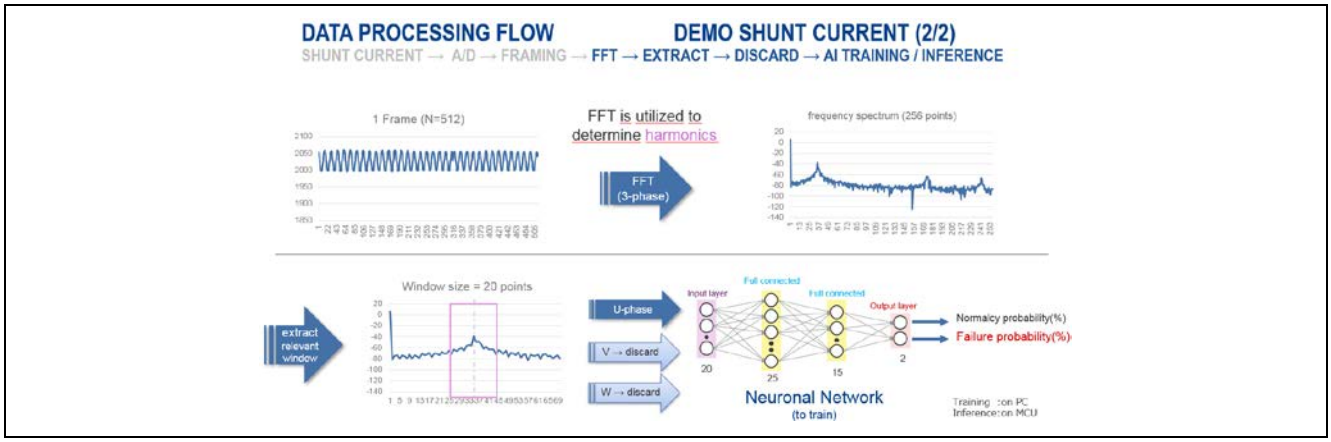


Figure 2-3. Data Processing Flow (2/2)

AI inference results are displayed in the PC software (DataCollectionTool). In addition to the AI inference results, this sample software also displays the 3 shunt current A/D conversion value and the spectrum waveform. The system's waveform data log function allows the system to accumulate NN training data while displaying waveforms.

Further details concerning the DataCollectionTool can be found in section 5. Waveform Monitoring Tool.

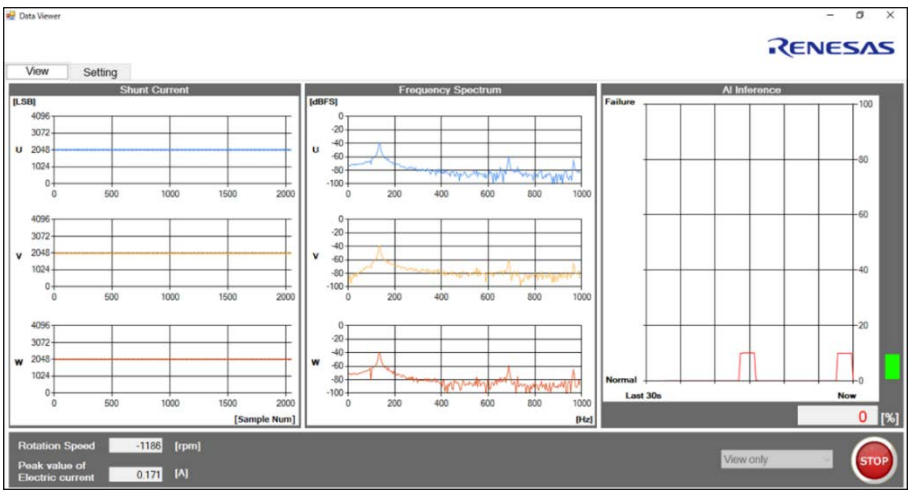


Figure 2-4. DataCollectionTool - Appearance

3. Specification

3.1 Operating Conditions

Table 3-1 lists the confirmed operation conditions.

Table 3-1. Operating Conditions

Item	Description
MCU	RX66T (R5F566TEADFP)
Operating Frequency	Main clock: 8 MHz crystal oscillator CPU clock (ICLK): 160MHz Peripheral module clock A (PCLKA): 80MHz Peripheral module clock B (PCLKB): 40MHz Peripheral module clock C (PCLKC): 160MHz Peripheral module clock D (PCLKD): 40MHz FlashIF clock (FCLK): 40MHz
Operating Voltage	5.0V (RX66T)
Operating Mode	Single-chip mode
Endian	Little Endian
Integrated Development Environment	Renesas Electronics e^{2}studio V7.2.0
C Compiler	Renesas Electronics CC-RX: V3.00.00 • Compiler option: -isa=rxv3 -fpu -save_acc
Auto generation tool	Renesas Electronics RX Smart Configurator V7.2.0
Emulator	Renesas Electronics E2 Lite emulator
Middleware	Renesas Electronics RX Family RX DSP Library Version 5.0
e-AI Development Environment	Renesas Electronics e-AI Translator V1.0.2 Renesas Electronics e-AI Checker V1.0.2
Evaluation Board	Renesas Electronics 24V Motor Control Evaluation System for RX23T (RTK0EM0006S01212BJ) • 24V inverter board Renesas Electronics RX66 CPU card (RTK0EMX870C00000BJ)
Motor	Tsukasa Electronics Co. Ltd. TG55-L (RTK0EM0006S01212BJ bundled) Simplified Motor Bench (see note)

Note: For more details, refer to Appendix entitled “Simplified Motor Bench Assembly Guide.”

3.2 Hardware Diagram

Figure 3-1shows the hardware block diagram of the e-AI Motor Abnormality Detection Sample Software. This sample is based on the hardware structure described in Reference Document [2]: “RX66T Application Note for Sensorless Vector Control of Permanent Magnetic Synchronous Motor,” and is enhanced with a communication function and drive motor load device. For details on the hardware structure that serves as the base, please refer to Reference Document [2].

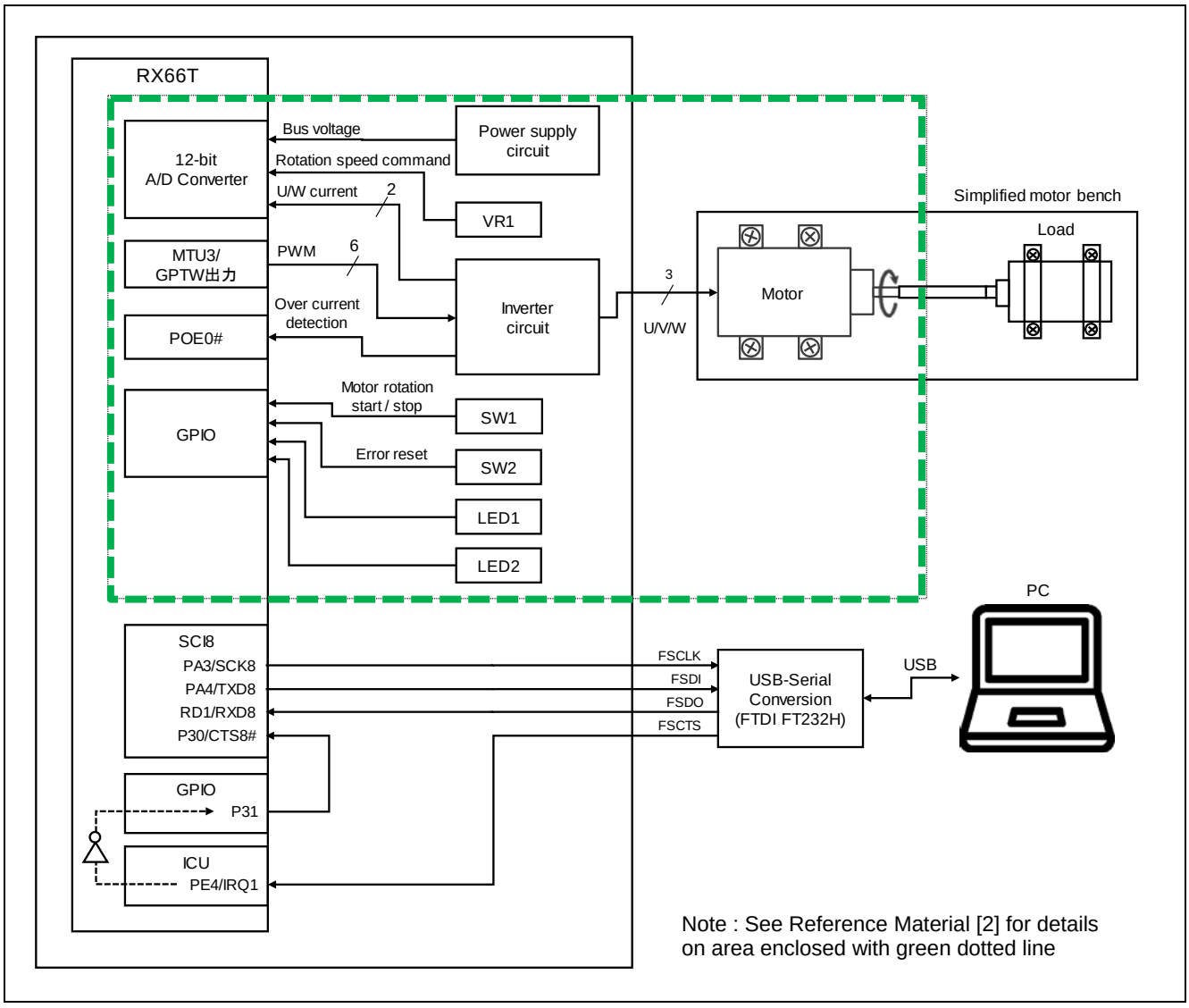


Figure 3-1. Hardware Block Diagram

3.3 Operations Overview

Figure 3-2 shows the system operation flow. The following is an overview of operations.

① Execute sensorless vector control on motor

When power is applied to the 24V inverter board, it is also applied to the RX66TCPU board, which starts the motor driver operations. See Reference Document [2] for processing details and Reference Document [3] for details on board operations.

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

1. A/D conversion value accumulation
   CMT1 generates the 2kHz sampling frequency and acquires the A/D conversion value of the motor 3 shunt current. The 3 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. From the next frame on, A/D conversion values are accumulated by overlapping 64 samples of the previous frame.
2. Data pre-processing
   The MCU performs the FFT operation using the RX DSP library. The frequency spectrum resulting from the FFT operation is converted into dBFS. This sample software defines 0dB = 4095LSB @ S12ADH Full Scale. Next, the peak value of the frequency spectrum (excluding the DC component) and the previous and successive 10 samples (A/D conversion values) are eliminated to extract the frequency spectrum feature points.
3. AI inference
   The extracted feature points are 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 SCI8 in the SPI mode, data is transferred to the PC using a USB-serial converter cable. DMAC0 is used to transfer data to the SCI8's transfer data register (TDR)

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.

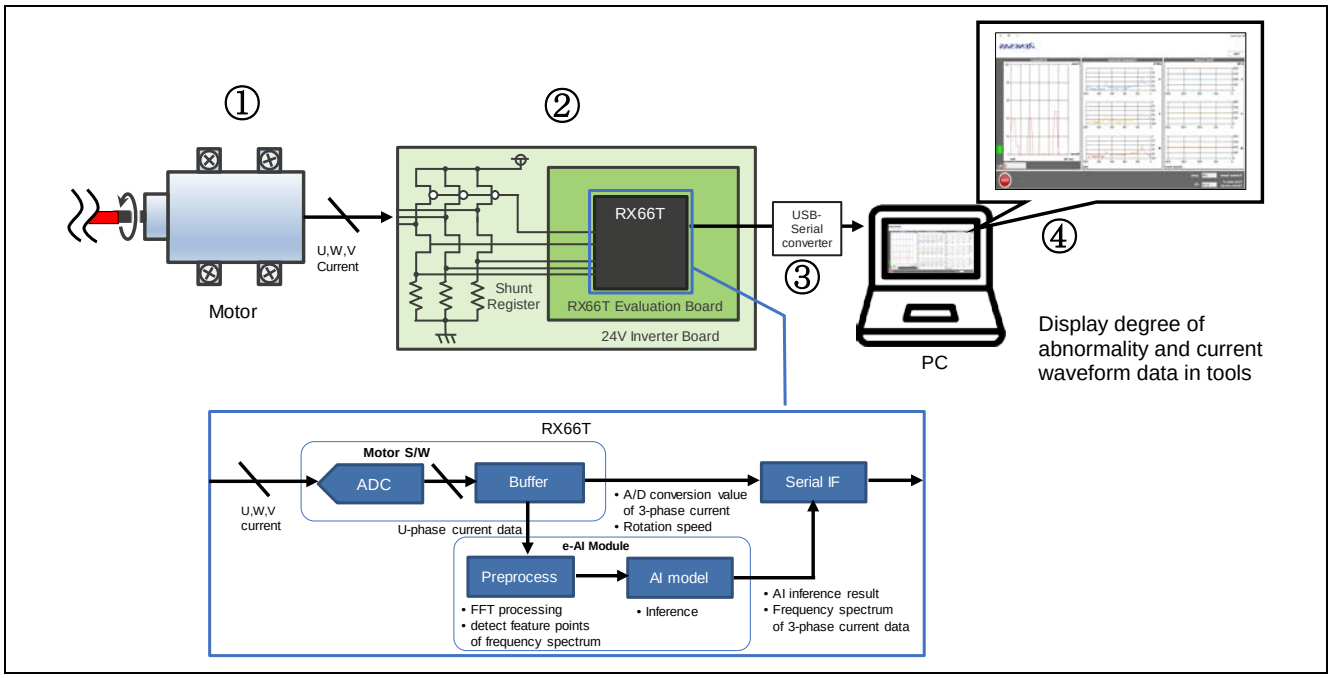


Figure 3-2. System Operation Flow

Figure 3-3 shows images of the system in normal and abnormal states. Normal state is defined as when the drive motor and load motor shafts form a single line and abnormal state is defined as when the axis of the two shafts is deviated. In this example, normal and abnormal states are recreated using a simple motor bench, coupling the drive motor and load motor shafts with a tube.

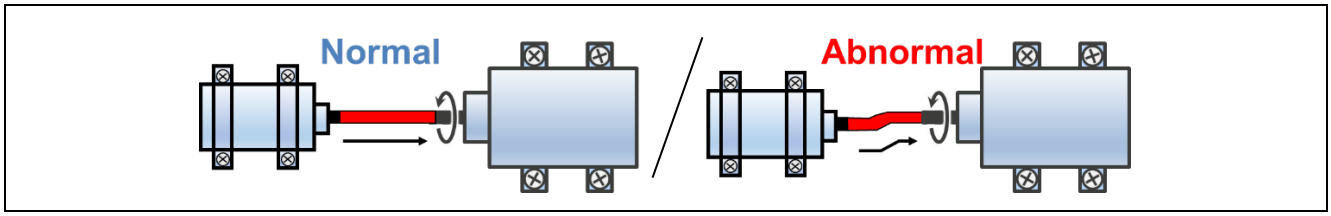


Figure 3-3. Images of Normal and Abnormal States

3.4 State Transition

Figure 3-4 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 for RX23T. RESET button refers to a device mounted on the RX66 CPU card.

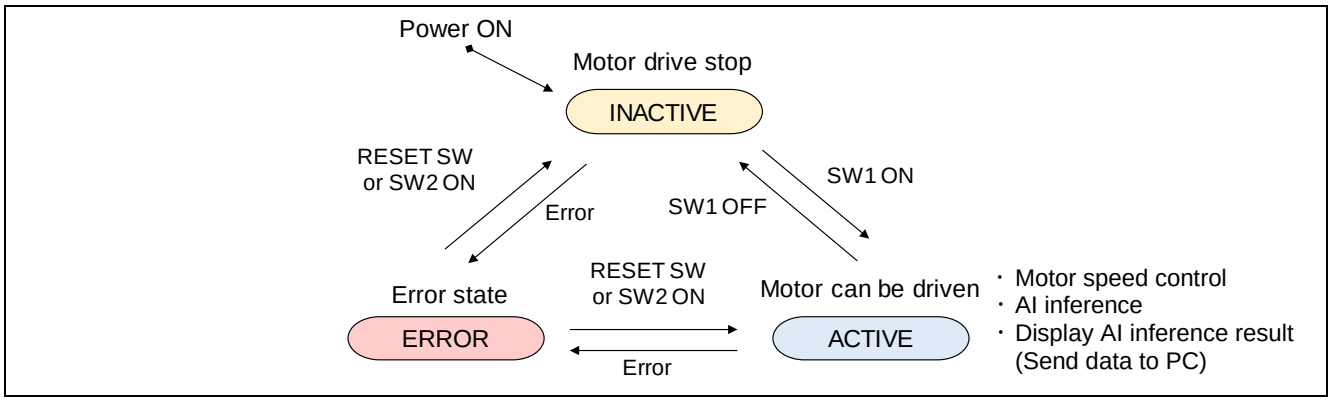


Figure 3-4. State Transition

(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.

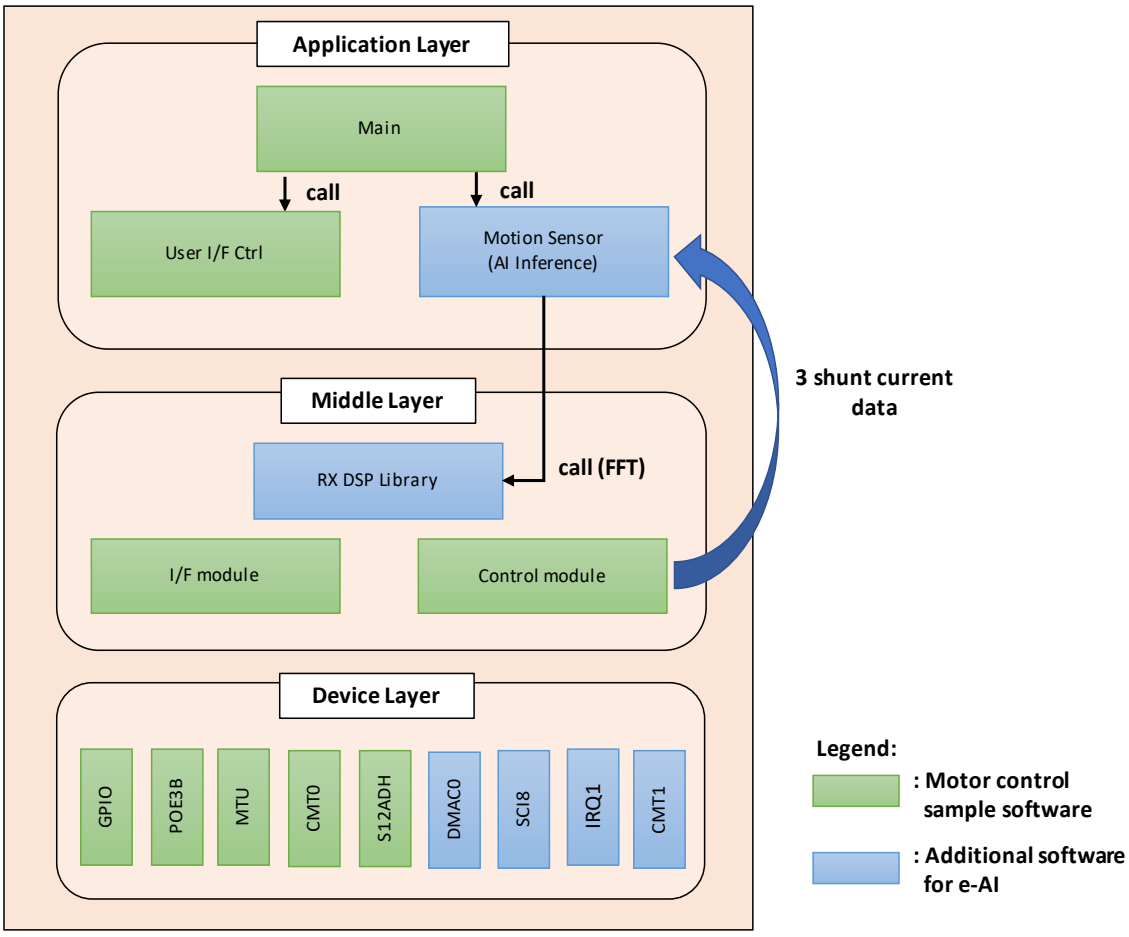
(3) ERROR

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.

4. MCU Software Explanation

4.1 Software Configuration

Figure 4-1 shows the software configuration of the e-AI Motor Abnormality Detection Sample Software. This sample is based on the sample software described in Reference Document [2]: “RX66T Application Note for Sensorless Vector Control of Permanent Magnetic Synchronous Motor,” and is enhanced with AI inference logic, related drivers, and RX Family RX DSP Library. For details on the sample software, please refer to Reference Document [2].



Note: Blue area indicated new features: For details on all other features and functions see Reference Material [2].

Figure 4-1. Software Configuration Overview

4.2 Directory Configuration

Figure 4-2 shows the directory configuration of the e-AI Motor Abnormality Detection Sample Software.

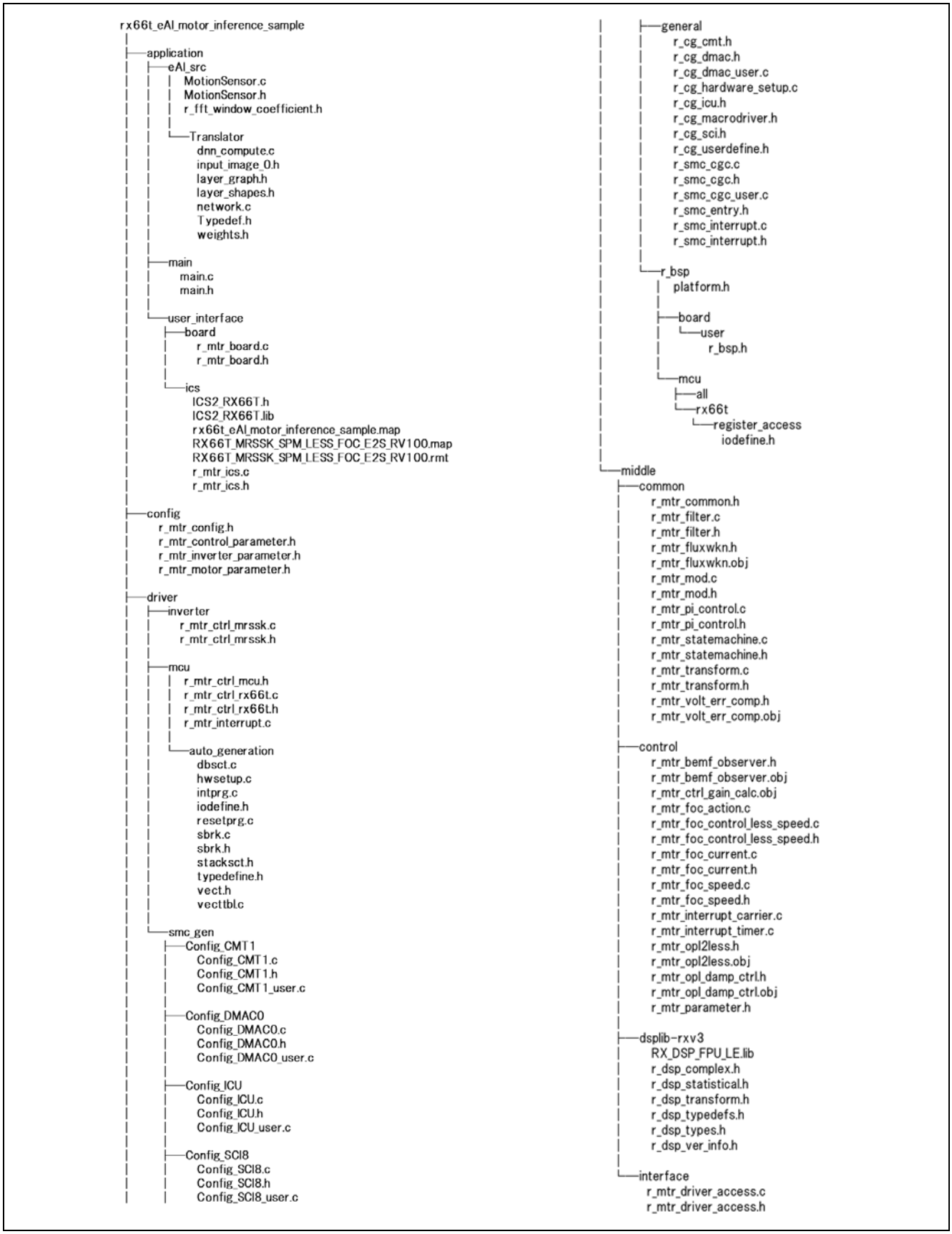


Figure 4-2. Directory Configuration

4.3 Resources

4.3.1 Resource List

Table 4-1 provides a list of resources used in the sample software. The items in bold are added peripheral functions unique to this software. Refer to Reference Document [2] for details on peripheral functions other than those in bold lettering.

Table 4-1. Resource List

Item	Description
Clock generation circuit	Generates operating clock from external oscillator.
S12ADH	• 3 shunt current measurement • Inverter bus line voltage measurement • Rotation speed command value input
CMT0	500μs interval timer
MTU3	Complementary PWM output
POE3B	Overcurrent detection, output short circuit detection
GPIO	• Switch (SW1, SW2) input • LED (LED1, LED2) ON/OFF control
CMT1	Generates 3 shunt current value sampling frequency (2kHz)
SCI8	Serial communication (SPI)
DMAC0	Transfers data to send data register based on SCI8 transfer request

4.3.2 Interruptions

Table 4-2 provides a list of interrupt processings. The items in bold are added peripheral functions unique to this sample software. Refer to Reference Document [2] for details on interrupts other than those in bold lettering.

Table 4-2. Interrupt Processings

Interrupt request generation source (peripheral module)	Name (peripheral module name)	Interrupt priority level (IPR)	Description
Group Interrupt 1 （POE3） （SCI8）	GROUPBL1 （OEI1） （TEI8、ERI8）	15 (highest)	• Motor interrupt processing (overcurrent detected) • Data send/receive to/from PC — SCI8 send complete interrupt (TEI8) — SCI8 receive error interrupt (ERI8)
PERIA（MTU3）	INTA209 （TCIV4）	12	Motor interrupt processing (500 μs)
SCI8	RXI8	12	Data receive from PC
CMT0	CMI0	11	Motor interrupt processing (50 μs)
SCI8	TXI8	8	Data send to PC (DMAC transfer trigger)
DMAC0	DMACI0	8	Clear transfer complete flag
CMT1	CMI1	5	3 shunt current sampling frequency (2kHz)

The following shows the interrupt processing change point for this sample software.

① File name: intprg.c
The following codes are commented out.

```
void Excep_CMT1_CMI1(void){ }
void Excep_SCI8_RXI8(void){ }
void Excep_SCI8_TXI8(void){ }
void Excep_DMAC_DMAC0I(void){ }
```

② File name: r_mtr_interrupt.c
Processings added to external variable and external function declarations, over-current detection interrupt process (Group 1 interrupts) and 50[μs] period interrupt (carrier interrupt) process.

```
/* Used for e-AI inference processing */
extern float g_current = 0;
extern void r_Config_SCI8_transmitend_interrupt(void);
extern void r_Config_SCI8_receiveerror_interrupt(void);

------------------------------------------------------------------
#pragma interrupt (mtr_over_current_interrupt(vect = VECT_ICU_GROUPBL1))
static void mtr_over_current_interrupt(void)
{
  /* Overcurrent current detection */
  if (1 == ICU.GRPBL1.BIT.IS9){
      mtr_over_current(&g_st_foc);
  }

  // SCI8 TEI8 (Send complete)
  if (1 == ICU.GRPBL1.BIT.IS24){
    r_Config_SCI8_transmitend_interrupt();
  }

  // SCI8 ERI8 (Receive error)
  if (1 == ICU.GRPBL1.BIT.IS25){
    r_Config_SCI8_receiveerror_interrupt();
  }
} /* End of function mtr_over_current_interrupt */


------------------------------------------------------------------
#pragma interrupt (mtr_tciv4_interrupt(vect = VECT_PERIA_INTA208))
static void mtr_tciv4_interrupt(void)
{
    mtr_foc_interrupt_carrier(&g_st_foc);
    mtr_ics_interrupt_process();

    /* Peak current acquisition (for e-AI inference processing) */
  if(g_current < g_st_foc.f4_iu_ad)
  {
    g_current = g_st_foc.f4_iu_ad;
  }
}
```

4.4 Main Processing

Figure 4-3 shows the flow chart for main processing. Motor control interrupt processing is also included to help clarify the relationship between main processing and motor control processing.
Main processing executes system initialization, then executes a loop of user interface processing and AI inference processing. User interface processing uses the 24V inverter board devices (SW1, SW2, VR1) to control the system, converts the input device information to system state and motor rotation speed command values, and passes this on to the motor control processing.
Motor drive control is performed by 2 types of motor interrupt processing. The motor can be driven when the system state is ACTIVE.

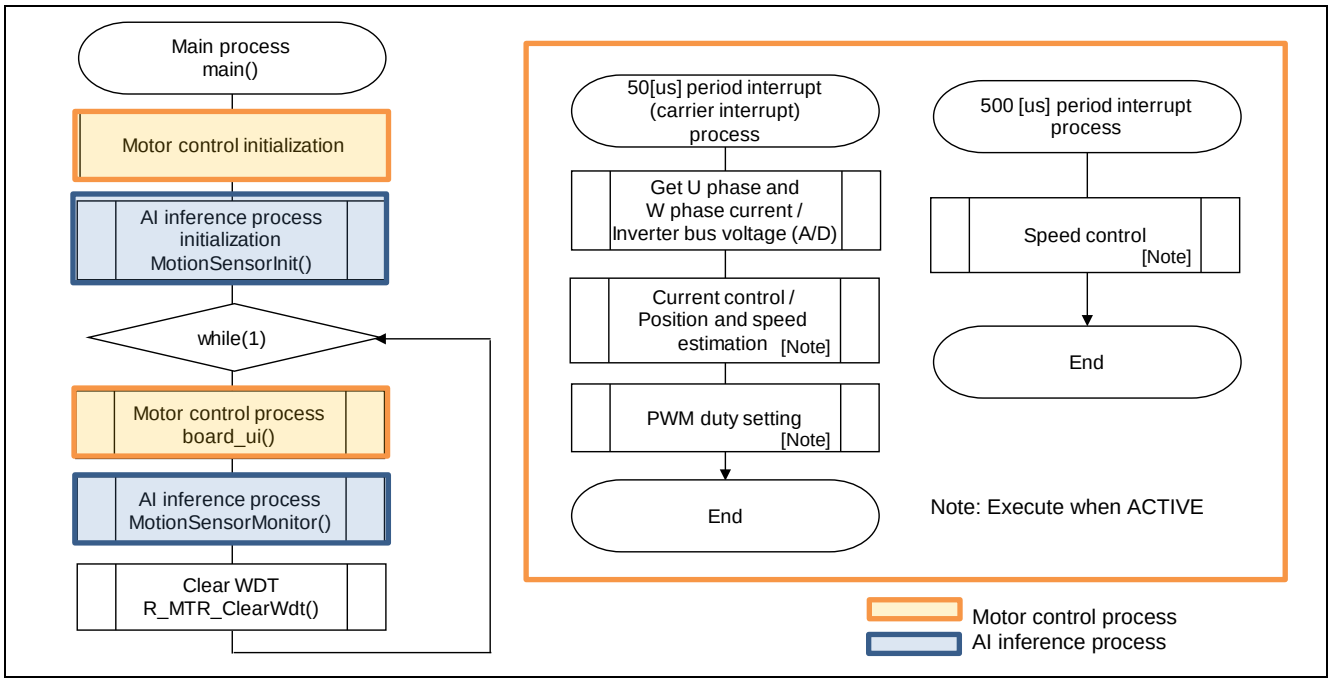


Figure 4-3. Main Processing Flow Chart

4.5 Motor Control Processing

This section describes only areas of the Reference Document [2] “RX66T Application Note for Sensorless Vector Control of Permanent Magnetic Synchronous Motor,” that have been changed for this sample software. Please refer to Reference Document [2] for details on all other motor control processings.
In this example, 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.

Table 4-3. Rotation Speed Command Value Conversion Rate

Item		Conversion Rate (command value: A/D conversion value)
Rotation speed command value	CW	1000[rpm]~2000[rpm] : 08C8H~0FFFH

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);
R_MTR_SetSpeed(s2_temp); /* Set speed reference */
```

4.6 FFT Processing

The RX family uses the DSP library for FFT processing of the 3 shunt current data. Table 4-4 lists the functions used in the DSP library.

Table 4-4. Functions Used in DSP Library

Function Name	Description
R_DSP_FFT_Init_i16ci32()	Initialize handle for real FFT operation
R_DSP_FFT_i16ci32()	Real FFT transform (output FFT result as complex number)
R_DSP_VecCplxMag_ci32i32()	Acquire complex magnitude value

The following shows the FFT source code used by the RX DSP library for the example.

```
#include <r_dsp_transform.h>

static void fExecuteFFT(int16_t *pInBuf, float *pOutBuf)
{
    uint32_t i;
    uint16_t SamplingHalf = gv_SamplingConditions.m_SamplingCount / 2;

    gs_fft_time.data = pInBuf;

    /* Real FFT transformation (output FFT result as a complex number)  */
    R_DSP_FFT_i16ci32( &gs_fft_handle, &gs_fft_time, &gs_fft_freq);

    /*
     * Obtain complex magnitude value
     * Analysis conditions when FFT length is 512 and the sampling frequency is 2kHz
     *  - frequency ticks: 2000Hz / 512 = 3.90625Hz)
     *  - Measurable frequency: 3.90625Hz * 255 = 996.09375Hz
     */
    R_DSP_VecCplxMag_ci32i32( &gs_fft_freq, &gs_fft_mag, SamplingHalf);

    /*
     * Replace 0th element of calculation result with gs_fft_buf[0].re
     * This process is necessary to remove the influence of real number half of the FFT
length.
     */
    *(int32_t *) gs_fft_mag.data = gs_fft_buf[0].re;

    /*
     * Convert to dBFS(decibels below full scale)
     * Step1 : Convert fixed point number (1Q15) to floating point number.
     *         Divide FFT library output result (stored in int32
gs_fft_VecCplxMagi32[]) by (2^15-1).
     *         It is half the amplitude of the A/D full scale value.
     * Step2 : Double the result of step1 to get full amplitude of A/D full scale value
     * Step3 : Convert the result of step 2 to voltage-ratio(dB).
     * Step4 : Subtract 12bit A/D full scale voltage-ratio (72.247199 dB) to convert to
dBFS.
     * Note : The maximum magnitude that can be used for dBFS is 0dBFS.
     * If an magnitude lower than this is used, a negative number will be displayed.
     */

    for (i = 0; i < SamplingHalf; i++)
    {
        if (0 != gs_fft_VecCplxMagi32[i])
        {
          pOutBuf[i] = (20.0 * log10f((float) gs_fft_VecCplxMagi32[i] / 16383)) -
gv_DecibelsBelowFullScale;
        }
        /* When the complex absolute value is 0 */
        else
        {
            pOutBuf[i] = -gv_DecibelsBelowFullScale;
        }
    }
}
```

The following shows the FFT initialization source code, which uses window function (hanning window) for the example.

```
static void fInitFft (void)
{
    uint16_t Count;

    /* Initialize FFT handle data */
    gs_fft_handle.n = gv_SamplingConditions.m_SamplingCount;
    gs_fft_handle.work = NULL;
    gs_fft_handle.options = (R_DSP_FFT_OPT_SCALE | R_DSP_FFT_BIT_REVERSAL_DEFAULT |
R_DSP_FFT_OPT_TWIDDLE_DEFAULT);
    gs_fft_handle.bitrev = gs_fft_bitrev;
    gs_fft_handle.twiddles = gs_fft_twiddles;
    gs_fft_handle.window = NULL;

    gs_fft_time.n = gv_SamplingConditions.m_SamplingCount;
    gs_fft_freq.n = gv_SamplingConditions.m_SamplingCount / 2;
    gs_fft_mag.n = gv_SamplingConditions.m_SamplingCount / 2;

    /* Set window coefficient according to FFT length */
      switch(gv_SamplingConditions.m_SamplingCount)
    {
        case DEF_SamplingCount1024:
            gs_fft_handle.window = (void *) i16_hanning_fft1024;
            break;
        case DEF_SamplingCount512:
            gs_fft_handle.window = (void *) i16_hanning_fft512;
            break;
        case DEF_SamplingCount256:
            gs_fft_handle.window = (void *) i16_hanning_fft256;
            break;
        case DEF_SamplingCount128:
            gs_fft_handle.window = (void *) i16_hanning_fft128;
            break;
        default:
            gs_fft_handle.window = NULL;
            break;
    }

    /* Initialize FFT */
    R_DSP_FFT_Init_i16ci32( &gs_fft_handle);

    /* Sampling size */
    gv_SamplingSize = (gv_SamplingConditions.m_SamplingCount * 2) -
gv_SamplingConditions.m_SamplingOverLap;

    /* Overlap area of sampling buffer #2 and sampling buffer #1 start areas  */
    gv_Buffer2to1Overlap = gv_SamplingSize - gv_SamplingConditions.m_SamplingOverLap;

    /* End position of #1 frame  */
    gv_Frame1End = gv_SamplingConditions.m_SamplingCount - 1;

    /* End position of #2 frame  */
    gv_Frame2End = gv_SamplingSize - 1;

    /* (Sampling count/2)**2    */
    gv_SquareSamplingHalf = (float) ((gv_SamplingConditions.m_SamplingCount / 2)
            * (gv_SamplingConditions.m_SamplingCount / 2));

    /* Sampling frequency */
    Count = (uint16_t) (DEF_CMT_CLK8 / (float)
(gv_SamplingConditions.m_SamplingFrequency)) - 1;

    /* A/D conversion trigger timer setting */
    CMT1.CMCOR = Count;
}
```

4.7 AI Inference Processing

4.7.1 Flowchart

Figure 4-4 shows the flowchart for AI inference processing.

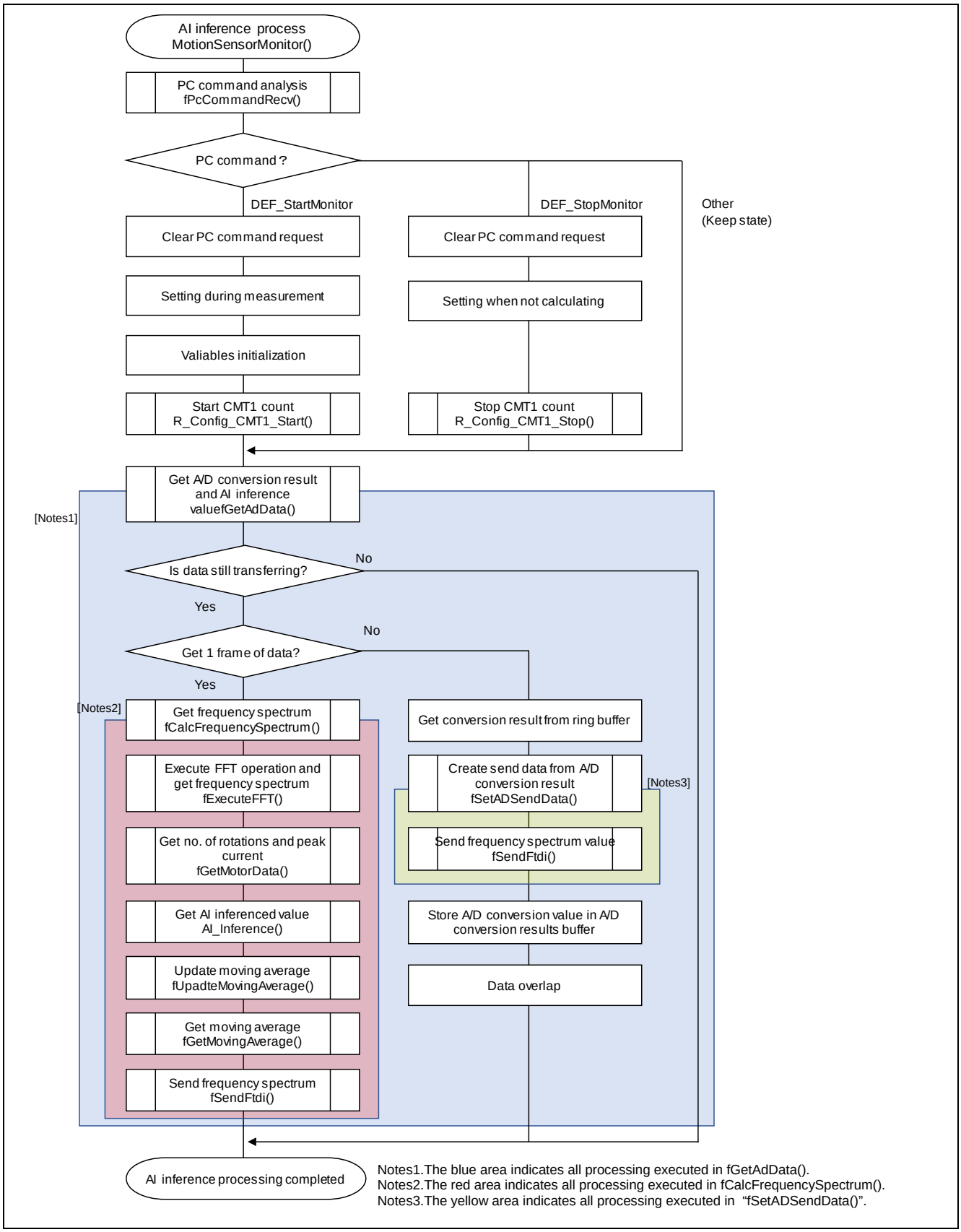


Figure 4-4. AI Inference Processing Flowchart

4.7.2 Data Flow

Figure 4-5 shows the data flow for AI inference processing.

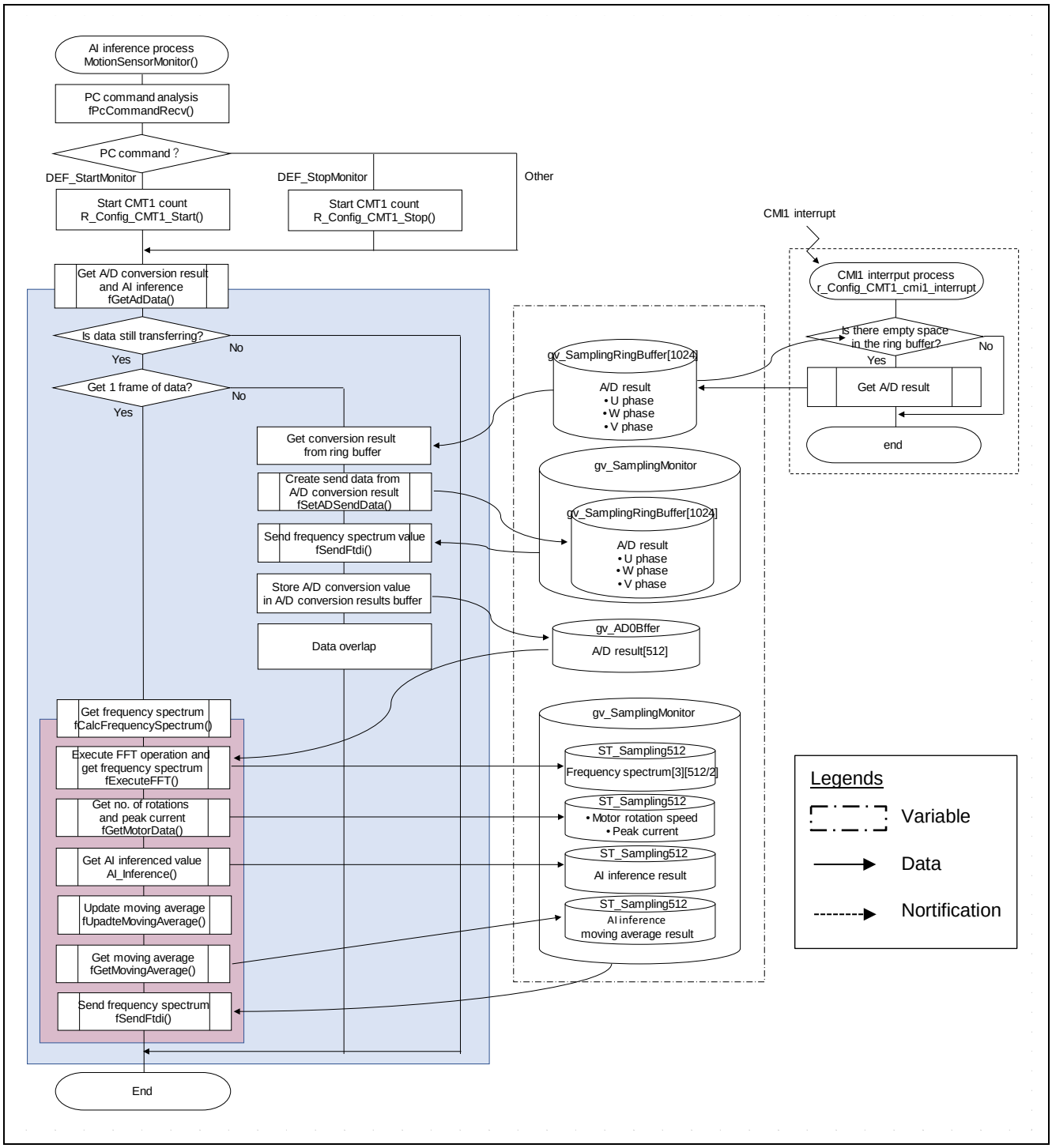


Figure 4-5. AI Inference Processing Data Flow

4.7.3 AI Model

Figure 4-6 shows the normal state and Figure 4-7 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.

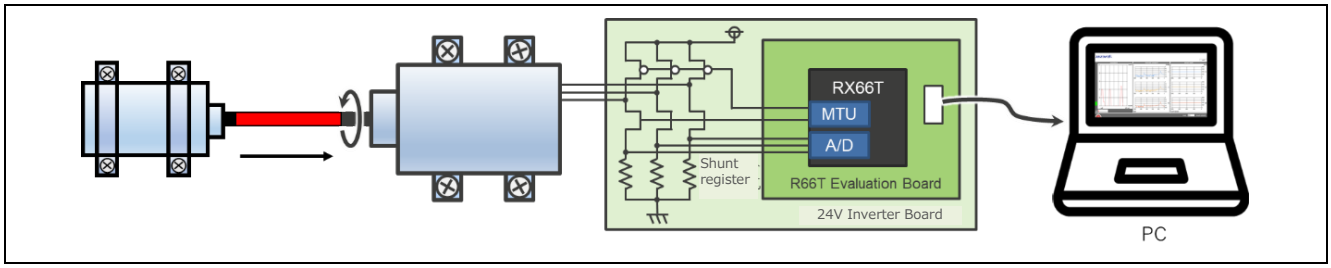


Figure 4-6. Normal State

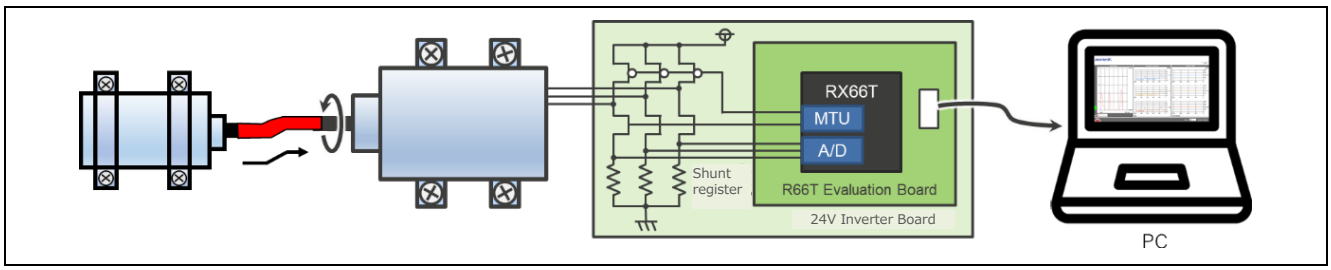


Figure 4-7. Abnormal State

Figure 4-8 shows the AI model configuration.

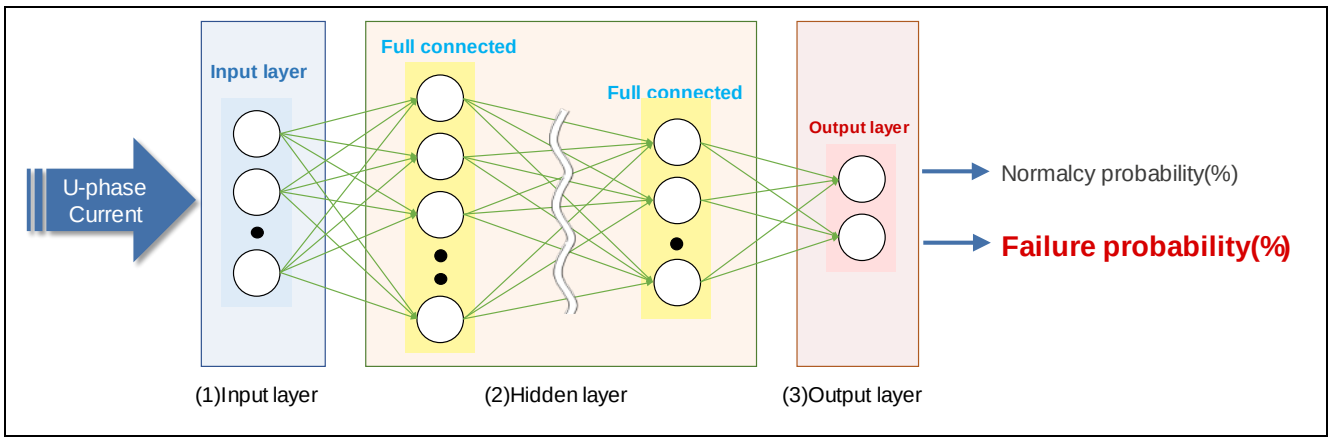


Figure 4-8. AI Model Configuration

(1) Input layer

FFT-processed 3 shunt current data is input to the input layer

(2) Hidden layer

The hidden layer uses the fully connected layer.

(3) Output layer

The output layer outputs the probability of normality and abnormality.

4.7.4 Update of the trained DNN

To update the trained DNN, please generate and train DNN with Tensorflow or Caffe, then translate the trained DNN to C source files by e-AI Translator.

When you use e-AI Translator to translate the trained DNN, specify the output folder as e-AI_src folder shown in Figure 4-2. Please refer to Reference Document [7] “User's Manual for e-AI Translator” for details

The following AI_Inference function call inference processing API (dnn_compute function) output by e-AI Translator, besides, it executes pre-processing which extracts an area defined by AI_INPUT_SIZE from frequency spectrum before inference processing. Please change AI inference processing and pre-processing related source code according to your system

```
static int8_t AI_Inference (uint16_t SamplingCount, float *pSpectrumX)
{
    TPrecision *p_prediction;
    int8_t AIresult;
    TsInt i;
    static float data_in[AI_INPUT_SIZE];

    /*The pre-processing function and dnn_compute function are called */
    for (i=0; i < AI_INPUT_SIZE; i++)
        {
            data_in[i] = pSpectrumX[i+1];
    }
   /* Call e-AI inference function */
   p_prediction = dnn_compute(data_in);
   AIresult = (int8_t) ((p_prediction[1] * 100) + 0.5);

    return AIresult;
}
```

5. Waveform Monitor Tool

5.1 Operation Overview

The waveform monitor tool is used to send the 3 shunt current data and AI inference value from the RX66T MCU in a serial communication, and to display those results. The tool does not require installation. An overview of the waveform monitor tool operations is as follows.

- Control display start/stop
- Display data sent from RX66T
  - 3 shunt current waveform data
  - Waveform data after 3 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

5.2 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.

5.2.1 View Tab

Figure5-1 shows the display layout used in the View tab. The numbers in the figure correspond to the numbers in the function descriptions below.

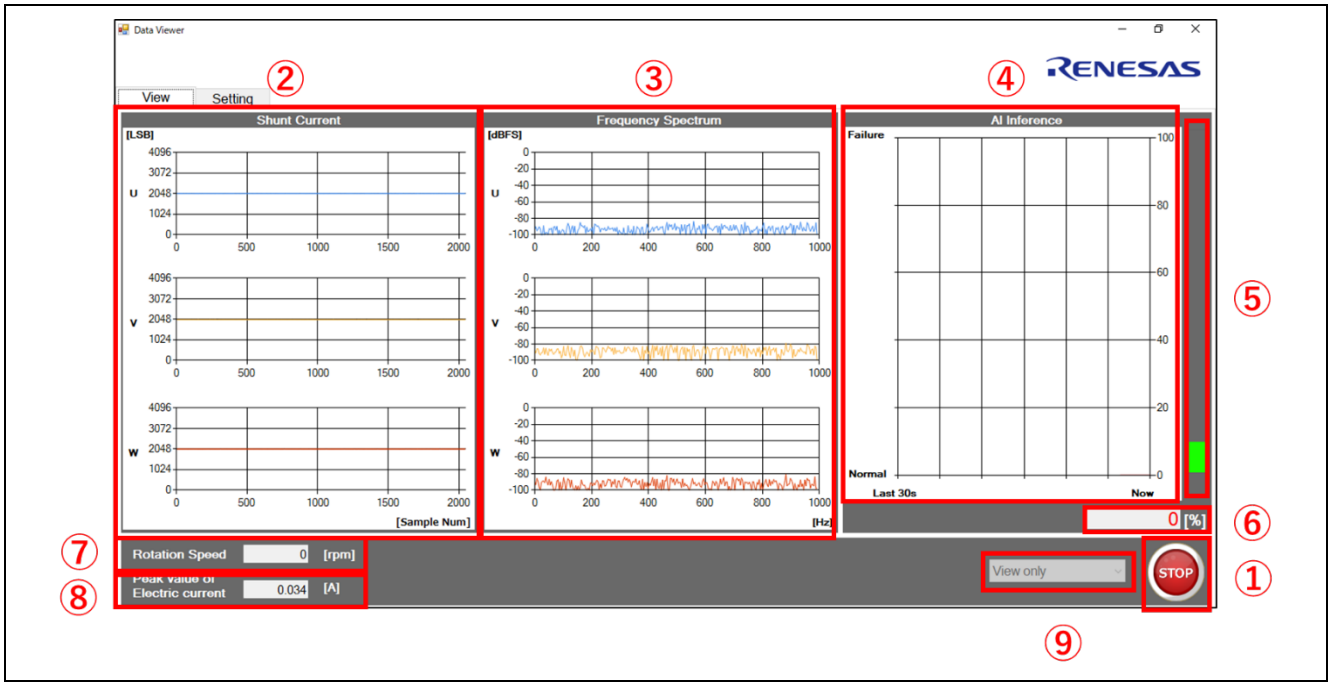


Figure5-1. View Tab Display Specifications

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 RX66T, and data is sent from RX66T 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 RX66T 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.

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

④ Moving average waveform of AI inference result
The moving average of the abnormality probability output by AI inference is generated and plotted in a waveform graph.

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

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

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

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

⑨ Log function selection
User selects whether to output log (CSV file) from 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)
  - Monitors various data and outputs logs. This setting outputs the sampling waveform and frequency spectrum (dBFS) displayed in ② and ③ to the file independently for each phase. Data is recorded after a line feed for every FFT frame.
- Save to CSV (combined)
  - Monitors various data and outputs logs. This setting outputs the sampling waveforms and frequency spectrums (dBFS) displayed in ② and ③ together in a single file. Data is added until the acquisition record is completed.

5.2.2 Setting tab

Figure5-2 shows the display specifications for the Setting tab. The numbers in the figure correspond to the numbers in the function descriptions below.

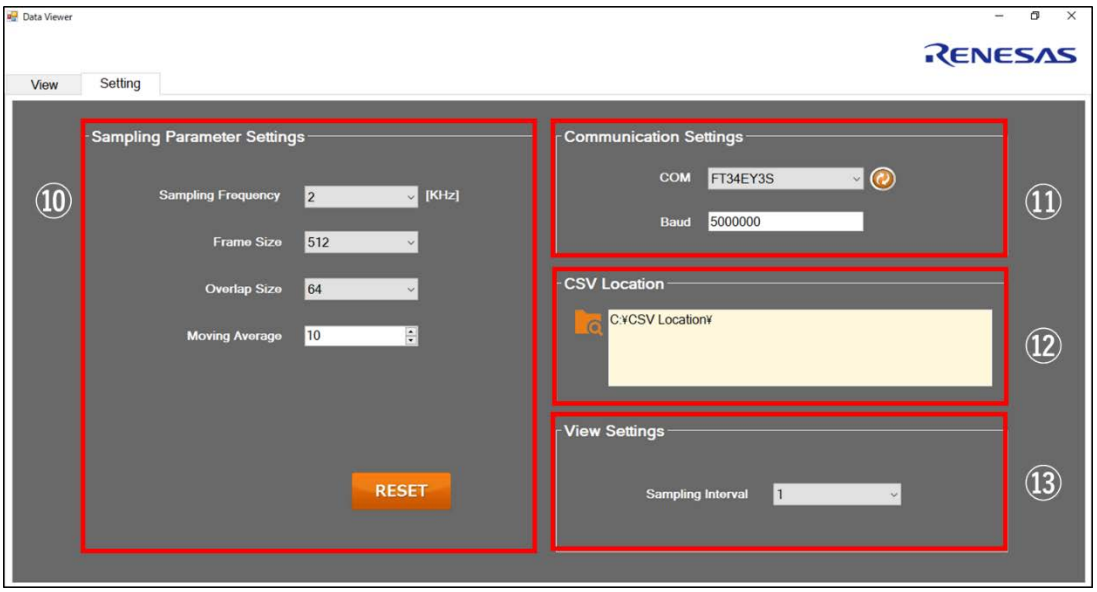


Figure5-2. Setting Tab Display Specifications

⑩ Sampling parameter setting

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

1. Sampling Frequency
   Specifies the sampling frequency (1/2/4/8 kHz, default: 2 kHz)
2. Frame Size
   Specifies the FFT frame size (128/256/512/1024, default: 512).
3. Overlap Size
   Specifies the FFT frame overlap size (16/32/64/128, default: 64).
4. 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

1. COM
   Acquires and displays the name of the FTDI device connected to the PC.
2. Baud
   Specifies the Baud rate for communications between the MCU and PC (range: 9600 to 5000000, default: 5000000).

⑫ CSV storage location setting

Specifies the CSV file output location when the View tab is set to output logs.

⑬ View settings

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

5.3 Operations

1. Startup

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

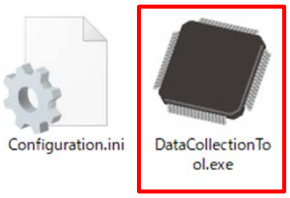


Figure 5-3. DataCollectionTool.exe file

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 Figure5-4.

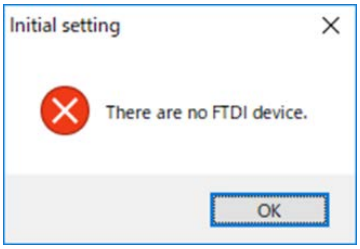


Figure5-4. Error Display

The screen shown in Figure5-5 appears when you open the exe file.

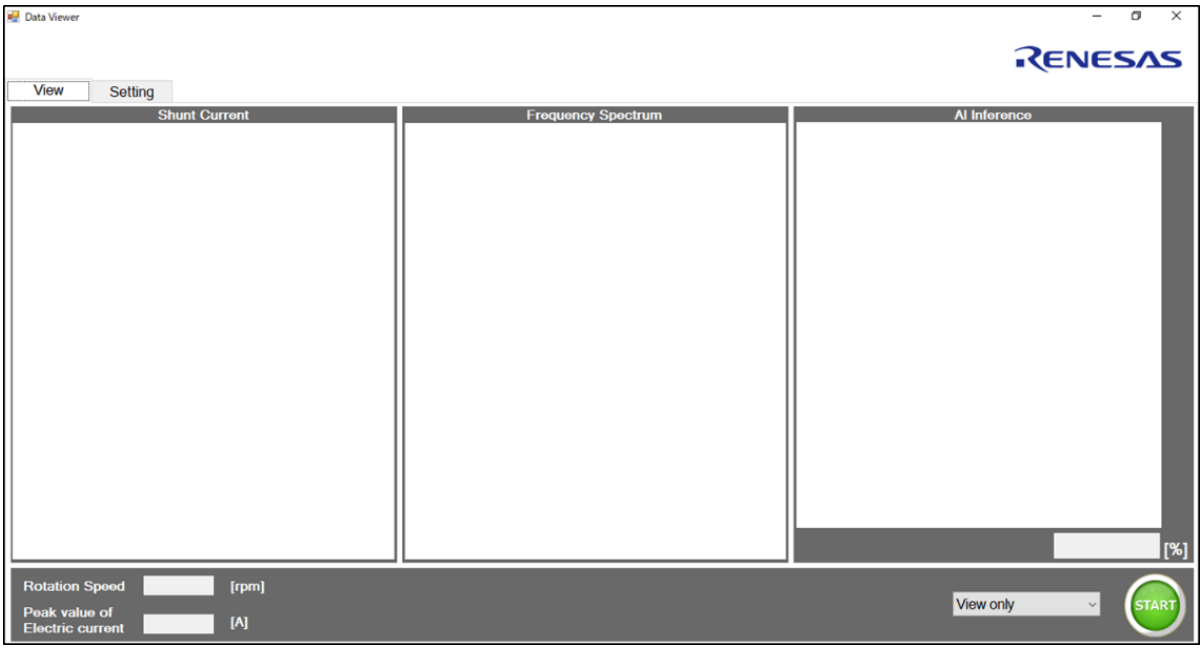


Figure5-5. Initial Startup Screen

2. Start data acquisition

Push the START button shown in Figure5-6, to start data acquisition

Figure5-6. Data Acquisition Start Button

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

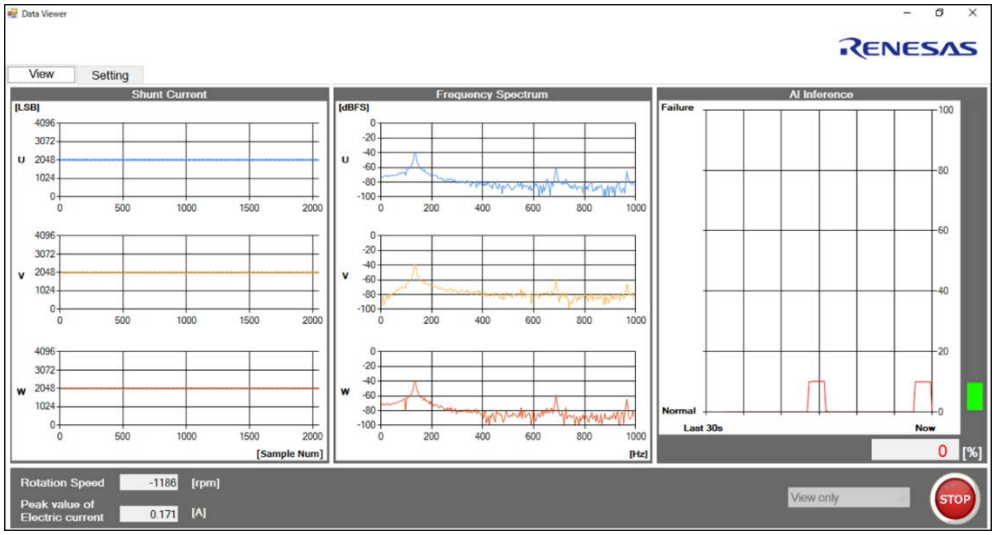


Figure5-7. Screen During Data Acquisition

3. Stop data acquisition

Push the STOP button shown in Figure 5-8, to stop data acquisition.

Figure 5-8. Data Acquisition Stop Button

By selecting “Save to CSV(divided)” from the drop-down list shown in “5.2.1 View tab,” you can output data for each axis and rotation speed/peak current data to CSV files while viewing the data. Figure5-9 shows the files created for saving the output data.

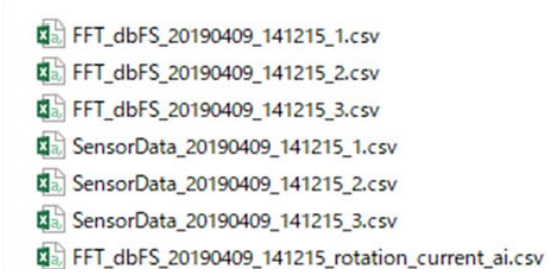


Figure5-9. Files Created when is “Save to CSV(divided)” is Selected

In the same manner, by selecting “Save to CSV(combined)” you can output data for each axis in a single CSV file while viewing the data. Figure5-10 shows the file created for saving the output data.

Figure5-10. File Created when ”Save to CSV(combined)” is selected

.

6. Reference Documents

[1] RX66T Group User’s Manual: Hardware (R01UH0749)
[2] RX66T Sensorless vector control for permanent magnetic synchronous motor (Implementation) (R01AN4244)
[3] Renesas Solution Starter Kit 24V Motor Control Evaluation System for RX23T User’s Manual (R20UT3697)
[4] RX66T CPU Card User’s Manual (R12UZ0029)
[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)
[9] Renesas Flash Programmer V3.05 Flash memory programming software User’s Manual (R20UT4307)

Appendix1. Operation Confirmation Method

1. Usage Notes

The evaluation device (Simple Motor Bench) described in this section was not constructed for operations or evaluations conducted over a prolonged period. In addition, vibrations during motor operations may cause problems, such as parts becoming loose or falling off the device. Please give due consideration to safety when confirming operations using this device.

2. Evaluation Environment

Table 1 provides details of the hardware environment required for development of based on this sample software.

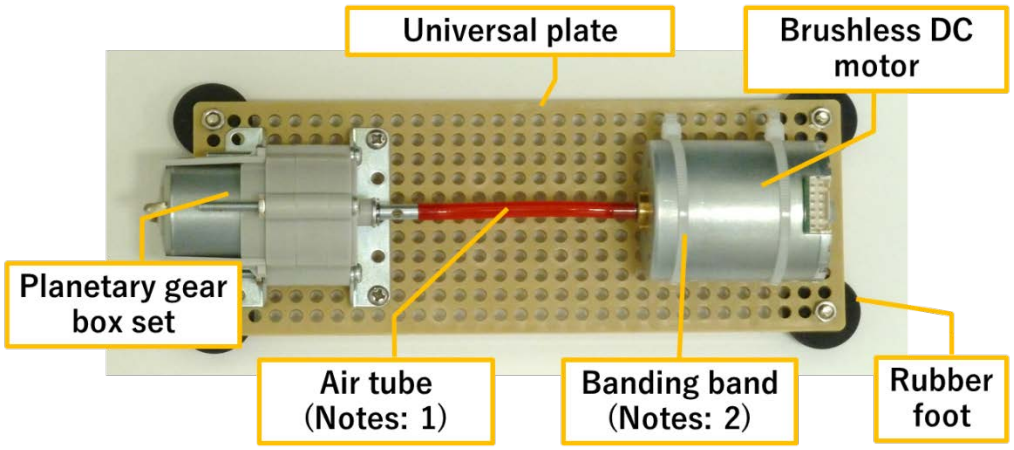
Table 1. Hardware Environment

Item	Name	Manufacturer	Spec/Model No. etc.	QTY
Evaluation board	RX66T CPU card	Renesas	RTK0EMX870C00000BJ	1
	24V inverter board	Renesas	RTK0EM0006S01212BJ	1
	24V AC adaptor	(general)	Output: over DC24V－2A Plug shape: diameter - external 5.5mm/internal 2.1mm Center plus	1
	USB serial converter cable (See note)	FTDI	C232HM-EDHSL-0 USB Hi-Speed to MPSSE Cable	1
	Pin header	(general)	2.54mm pitch, 36 positions x 2 rows	1
Drive motor	Brushless DC motor	Tsukasa Electric Co. Ltd.	TG-55L (bundled with RTK0EM0006S01212BJ)	1
Simple motor bench	Universal plate	Tamiya Inc.	Item No:70098	1
	Planetary gear box set	Tamiya Inc.	Item No:72001 Use one 4:1 gear unit	1
	Rubber foot	(general)	－	4
	Air tube	(general)	External diameter: 4mm, internal diameter 2.5mm Cut to 52mm	1
	Banding band	(general)	Width: 2mm	4
	Universal board	(general)	2.54mm pitch	1
PC for display	－	(general)	OS：Windows® 10 Processor: 1.6GHz or more Main memory: 1G Byte or more Interface: USB2.0 (for E2Lite or USB-serial cable connection)	1

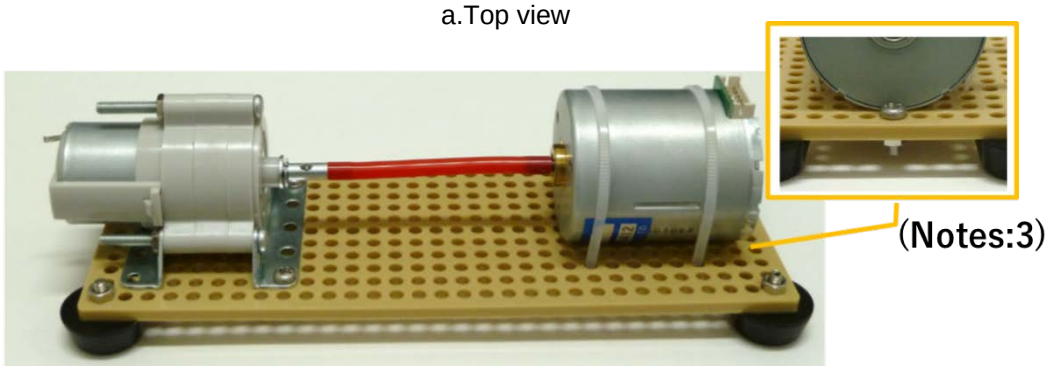
Note: Depending on the operating environment, noise may be superimposed on the transfer data. If this occurs, try various measures such as adjusting the operating environment or making the cable shorter.

3. Simple motor bench assembly

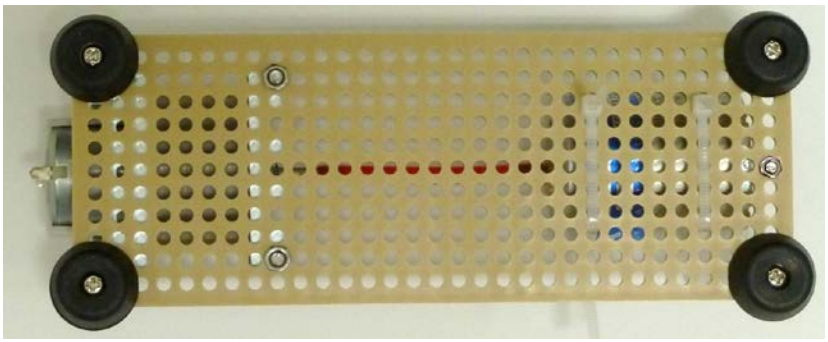
This section describes how to assemble a simple motor bench. The complete bench is shown in Figure 1.



a.Top view



b. Side view



c. Bottom view

Notes1. Connect the air tube before fixing the planetary gear box to the universal plate.
Make sure the tube is inserted securely into the base of the shaft.

Notes2. Tightly secure the brushless DC motor to the board with the bands.

Notes3. Screw as a stopper to fix the brushless DC motor.

Figure 1. Simple motor bench - Appearance

Figure 2 shows the position of the holes for the banding band to secure the load motor.

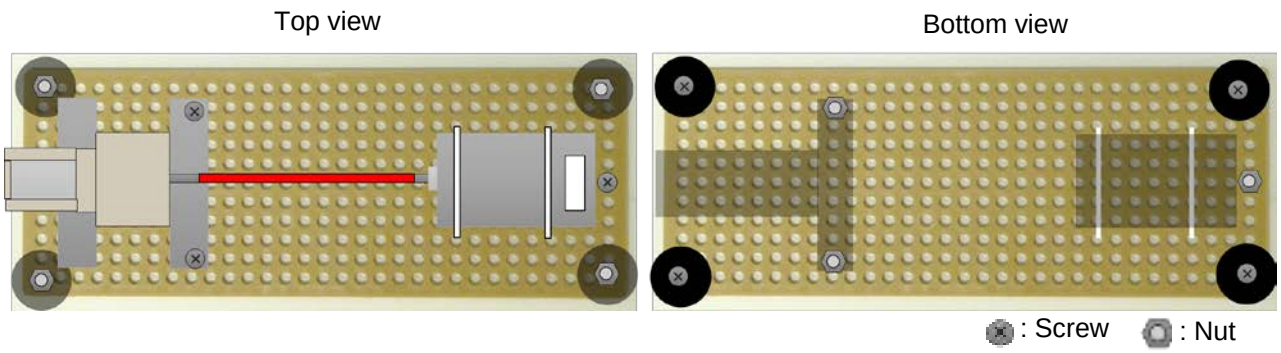


Figure 2. Simple motor bench - Parts location

4. USB-serial converter cable

USB-serial converter cable C232HM-EDHSL-0, manufactured by FTDI (Future Technology Devices International), is used for communication between RX66T and the PC.

i. How to connect the cable

- Solder the pin header (36x2) to the CNC through hole on the RX66T CPU board.
- Referring to Table 2, connect the C232HM-EDHSL-0 pins to the CNC pin header. Connect pins 27 and 28 of the CNC pin header with short block jumper.

Table 2. USB-Serial converter cable pin connection list

C232HM-EDHSL-0 pin assign		**CNC pin header pin assign**	
Function	**Wire color**	**Pin No.**	**Function**
FSCLK	Yellow	29	SCK8
FSDI	Orange	15	TXD8
FSDO	Green	35	RXD8
FSCTS	Brown	33	IRQ1
-	-	28 - 27 (Jumper short)	P31 -> CTS8#
GND	Black	31	GND

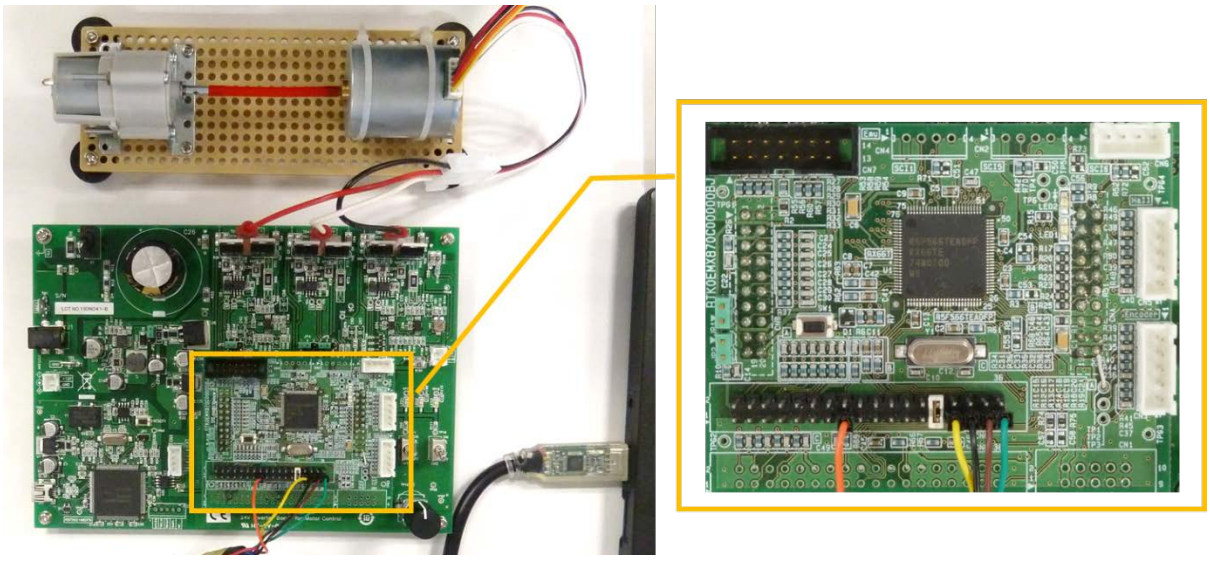


Figure 3. USB-Serial converter cable connection diagram

ii. Device Setting

Note that the FT232HL operating mode must be changed before attaching the USB serial converter cable. Please carry out the following steps to change the operating mode.

① Download the **FT_Prog** software from the FTDI site and install.
http://www.ftdichip.com/Support/Utilities.htm#FT_Prog

② Download and install **.NET Framework 4.0** which is required for running FT Prog.
https://msdn.microsoft.com/ja-jp/vstudio/ff687189.aspx

③ After the above programs are downloaded and installed, execute FT_PROG.exe. Figure 4. shows the initial startup screen after executing FT_PROG.exe. Click the magnifying glass icon (circled in red in the figure) to view a list of connected FTDI devices. Connect the USB serial conversion cable here, then press the magnifying glass icon (circled in red). The connected devices will then be displayed on the screen.

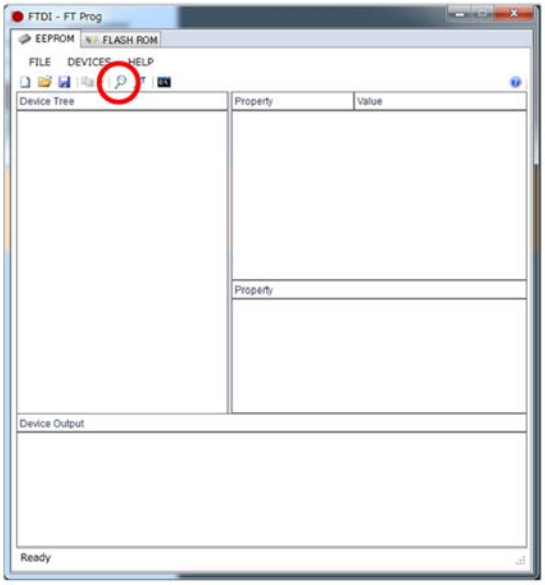


Figure 4. Initial Startup Screen

④ Make the following changes to operating modes as shown in Figure 5: [Hardware]→OPTO isolate, [Driver]→D2XX.

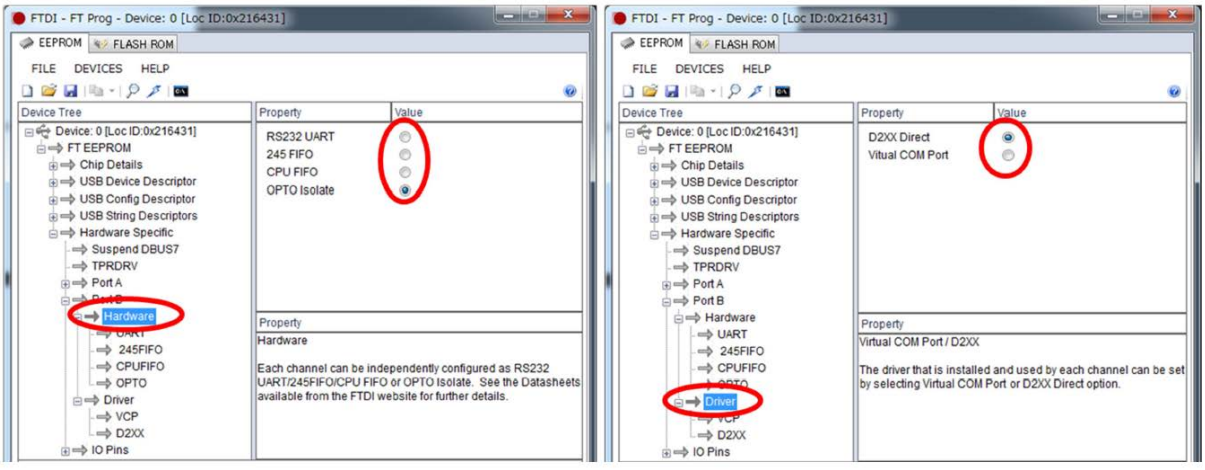


Figure 5. “Hardware” Driver Setting Screen

⑤ Click the lightning icon (circled in red in Figure 6) to open the EEPROM program dialog

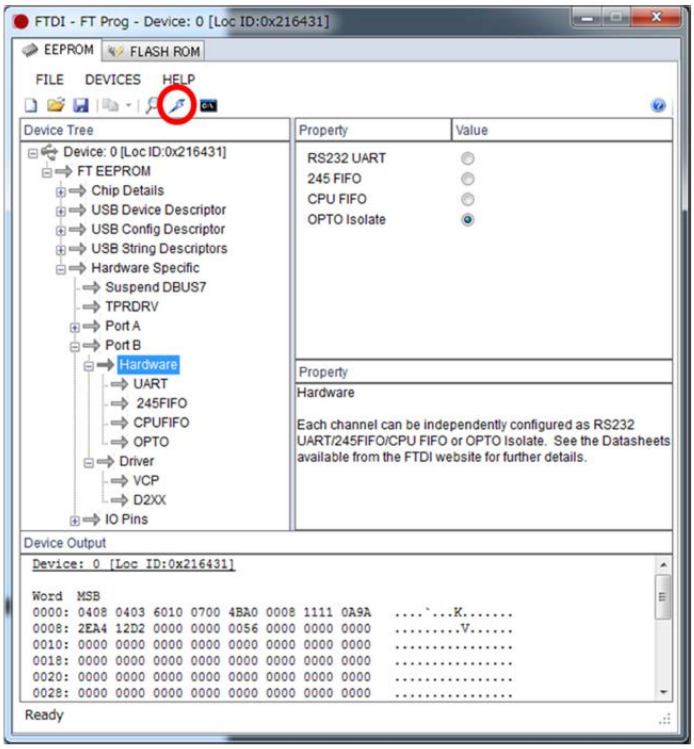


Figure 6. Setting Screen

⑥ Press Program (circled in red in Figure 7.) to implement all of the changes. After a few seconds, the settings will be programmed to the EEPROM on the board.

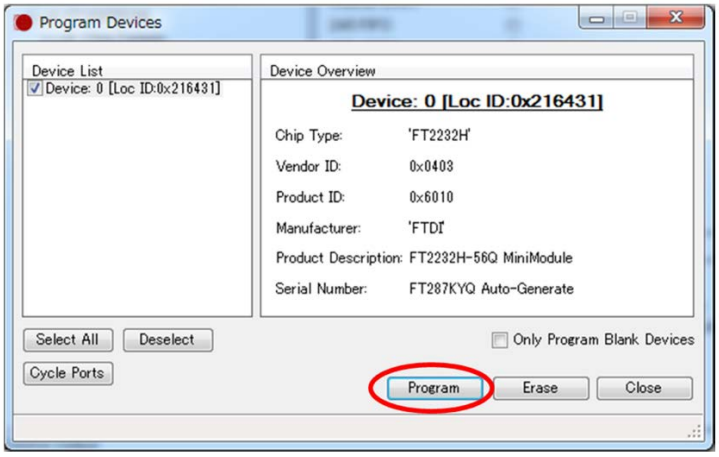


Figure 7. FTDI Device Program Screen

⑦ After programming is completed, detach the USB-serial conversion cable from the PC. Once the cable is re-attached, the system will operate based on the new settings.

5. Write ROM file to the MCU

This section explains how to write a ROM file to the MCU using E2 Lite. Refer to Reference Document [4] “RX66T CPU Card User’s Manual” for details on the position of the emulator connector for the E2 Lite connection.

① e2studio

For instructions on how to import a sample project into e2 studio using E2Lite, refer to Reference Document [8] “e2 studio Integrated Development Environment User’s Manual – Getting Started Guide.

② Renesas Flash Programmer

For instructions on how to write a mot file with the Renesas Flash Programmer, refer to Reference Document [9] “Renesas Flash Programmer V3.05 Flash Memory Programming Software User’s Manual.”

Make sure to specify DefaultBuild/”Samplesoft name”.mot of the sample software as in the path of the program file to be written to the MCU.

6. Operations

Refer to Reference Document [3] for explanations on how to supply the power, control the motor, and turn off the power.

Instructions on how to control the motor speed with VR1 are provided below. The letters and numbers correspond to indications in the figure.

- Rotation speed
  - ⓪：Stop
  - MIN：1000rpm
  - MAX：2000rpm

Note: Please operate the VR1 gently. Operating it too suddenly will cause the overcurrent protection to start up.

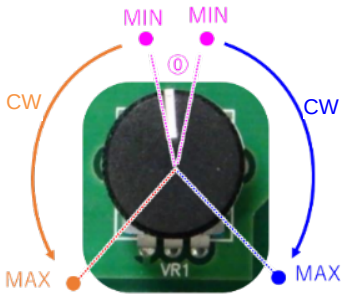


Figure 8. Control of motor rotation speed

The following shows the normal and abnormal states. This sample software defines the shaft deviation as the abnormal state. You can reproduce the abnormal state by pressing the universal plate to bend the air tube a little. Please lightly press the universal plate with finger like a Figure 10 because a strong pressing force may result in motor stop. As a guide, it is reach 0.3A when the motor rotation speed is 1000rpm.

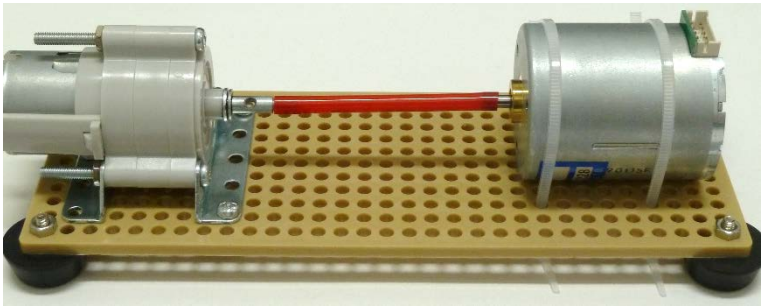


Figure 9. Normal State

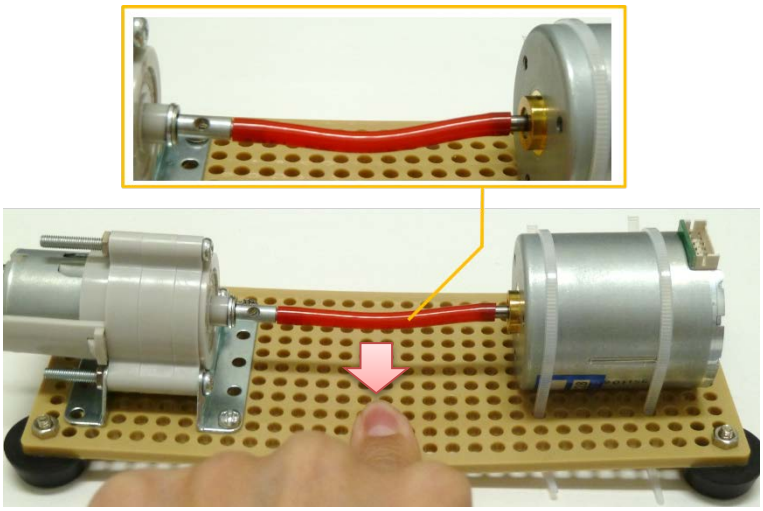


Figure 10. Abnormal State

7. Reference information

The following describes the reference information of simple motor bench evaluation. If you can't get an expected AI inference result, load of simple motor bench may not be appropriate. Please adjust it by referring to the peak current value in the following table. The peak current value and AI inference result are average value of 10 seconds that is calculated from wave monitor tool log.

Motor Rotation Speed (*)	Motor Load	Motor bench A		Motor bench B		Motor bench C	
		Peak Current (A)	AI Inference Result (%)	Peak Current (A)	AI Inference Result (%)	Peak Current (A)	AI Inference Result (%)
1000rpm	Normal	0.121	0	0.130	15	0.113	13
	Abnormal	0.306	84	0.407	100	0.285	93
1500rpm	Normal	0.124	1	0.141	0	0.124	0
	Abnormal	0.318	98	0.537	100	0.327	88
2000rpm	Normal	0.135	5	0.156	6	0.131	1
	Abnormal	0.285	95	0.531	96	0.290	96

* : Adjusting the motor rotation speed displayed in wave monitor tool..

Appendix2. MCU Software: detailed information

1. Memory Usage

Table 1. Memory Usage

Item	Total Size	Description
RAM	40.02KB	―
ROM	41.87KB	Romdata section : 11.36 KB Program section : 30.51 KB

2. CPU Load

Table 2. CPU Load

Item	Processing Time	CPU Load Factor
50 [us] period interrupt (carrier interrupt) process	12.5µs	25.00%
500 [us] period interrupt process	2.4µs	0.48%
FFT process (fExecuteFFT()) (*)	2128.7µs	0.95%
e-AI inference process (AI_Inference()) (*)	316.7µs	0.14%

* : Except interrupt processing time

3. Smart Configurator Settings

i. Clock settings

Table 3 lists the clock settings for the sample software.

Table 3. Clock Settings

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=40MHz)
SCKCR(ICLK[3:0])	x1 (ICLK=160MHz)
SCKCR(PCLKA[3:0])	x1/2 (PCLKA=80MHz)
SCKCR(PCLKB[3:0])	x1/4 (PCLKB=40MHz)
SCKCR(PCLKC[3:0])	x1 (PCLKC=160MHz)
SCKCR(PCLKD[3:0])	x1/4 (PCLKD=40MHz)
SCKCR(BCK[3:0])	x1/4 (BCLK=40MHz)

Note: Default values are used for settings not listed here.

ii. CMT1

CMT1 creates the motor's 3 shunt current A/D sampling frequency (2kHz).

Table 4 shows the settings for CMT1.

Table 4. CMT1 Settings

Item	Description
Clock setting	PCLK/8
Interval time	500 μ s
Enable compare match interrupt (CMI1)	Select ON
Interrupt priority	11

Note: Default values are used for settings not listed here.

iii. SCI8

Use SCI8 to carry out serial communication between RX66T and the PC. FTDI's USB-serial converter cable C232HM-EDHSL-0 serves as the serial communication interface, and transfers are carried out in the MPSSE(*1) mode. To enable this, the SCI8 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 5 lists the settings for SCI8.

Table 5. SCI8 Settings

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	5,000,000
Bit modulation function	Enabled
SCK8 pin function	Clock output
Hardware flow control	CTS8#
Send data processing	Processed by DMAC
Receive data processing	Processed in interrupt service routine
TXI8 priority	12
RXI8 priority	8
Receive error interrupt enable (ERI8)	Enabled
TEI8, ERI8 priority (group BL1)	Level 15 (Notes2)
Callback function setting	Send complete, receive complete, receive error
Pins used	TXD8: PA4/TXD8 (pin 37) RXD8: PD1/RXD8 (pin 24) SCK8: PA3/SCK8 (pin 38) CTS8: P30/CTS8# (pin 63)

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_SCI8_user.c (output by smart configurator)
Target function: R_Config_SCI8_Create_UserInit function (created by smart configurator)

```
void R_Config_SCI8_Create_UserInit(void)
{
    /* Start user code for user init. Do not edit comment generated here */
    /* Set 9th bit to "0" */
    SCI8.TDRHL.BYTE.TDRH = 0xFE;
    /* End user code. Do not edit comment generated here */
}
```

iv. DMAC0

Table 6 lists the settings for DMAC0.

Table 6. DMAC0 Settings

Item	Setting Value
Activation factor	Set to SCI8 (TXI8)
Activation factor flag control	At the start of transfer, clear the interrupt flag that became the activation factor
Transfer mode	Normal mode
Transfer data size	8 bits
No. of transfers	1 (set in application at time of transfer)
Transfer origin address	0x00000000 (set in application at time of transfer)
Transfer origin address update method	Increments
Transfer destination address	0x00000000 (set separately in user initial setting section)
Transfer destination address update method	Fixed address
Interrupt setting (DMAC0I)	Enable transfer complete interrupt
Interrupt priority	8

Note: Default values are used for settings not listed here.

The following code is added to set the DMAC0 transfer address to the SCI8 send register.
Target file: Config_DMAC0_user.c (output by smart configurator)
Target function: R_ConfigCMAC0_Create_UserInit function (created by smart configurator)

```
void R_Config_DMAC0_Create_UserInit(void)
{
    /* Start user code for user init. Do not edit comment generated here */
    DMAC0.DMDAR = (void *)&SCI8.TDRHL.BYTE.TDRL;
    /* End user code. Do not edit comment generated here */
}
```

4. Functions

Table 7 lists the functions used in main processing. The functions in bold lettering are new additions or have been updated.

Table 7. Functions Used in Main Processing

Function Name	Description
clrpsw_i()	Disables interrupts
R_MTR_InitHardware()	Initializes motor peripheral functions
R_MTR_InitBoardUi()	Initializes user interface
software_init()	Initializes variables 変
R_MTR_InitControl()	Initializes three-phase vector motor control
ics2_init()	Initializes for use of " Renesas Motor Workbench" tools
R_MTR_ExecEvent()	Executes FOC control events
R_Systeminit()	Initializes sensor peripheral functions
setpsw_i()	Enables interrupts
R_MTR_ChargeCapacitor()	Waits for bus voltage stabilization
MotionSensorInit()	**Initializes AI inference processing**
board_ui()	**Motor control processing**
ics_ui()	Processing for use of " Renesas Motor Workbench"
MotionSensorMonitor()	**AI inference processing**
R_MTR_ClearWdt()	Clears WDT

Table 8 lists the functions used for motor control processing.

Table 8. Functions used for Motor Control Processing

Function Name	Description
R_MTR_GetStatus()	Acquires motor status
get_sw1()	Acquires SW1 position
get_vr1()	Acquires VR1 position
R_MTR_SetSpeed()	Sets rotation speed command value
get_sw2()	Acquires SW2 state

Table 9 lists the functions used for AI inference processing.

Table 9. Functions used for AI Inference Processing

Function Name	Description
fGetAdData()	Acquires A/D conversion value
fSetADSendData()	Creates A/D conversion result send data
fCalcFrequencySpectrum()	Executes FFT operation, acquires frequency spectrum
fExecuteFFT()	Executes FFT operation
fInitFft()	Initializes FFT operation
fGetMotorData()	Acquires rotation speed and peak current
fSendFtdi()	Sends data to PC
fPcCommandRecv()	Analyzes PC command
fPcGetData()	Acquires PC command
fUpadteMovingAverage()	Updates AI inference value moving average data
fGetMovingAverage()	Acquires AI inference value moving average data

5. Constants

Table 10 lists the constants used for AI inference processing. The blue area indicates constants used for the FFT operation used by the DSP library.

Table 10. Constants (MotionSensor.h)

MotionSensor.h		
Constant Name	Setting Value	Description
DEF_PCLKB	40000000.0	PCLKB setting value (Unit: MHz)
DEF_CMT_CLK8	DEF_PCLKB / 8.0	CMT1 operating clock
DEF_DefaultSamplingFrequency	2000	Default sampling frequency value (Unit: Hz)
DEF_MinSamplingFrequency	1000	Minimum sampling frequency (Unit: Hz)
DEF_MaxSamplingFrequency	8000	Maximum sampling frequency (Unit: Hz)
DEF_DefaultSamplingCount	512	Default sampling count
DEF_SamplingCount128	128	Sampling count (128)
DEF_SamplingCount256	256	Sampling count (256)
DEF_SamplingCount512	512	Sampling count (512)
DEF_SamplingCount1024	1024	Sampling count (1024)
DEF_MaxSamplingCount	DEF_SamplingCount512	Maximum sampling count
DEF_MinSamplingCount	DEF_SamplingCount128	Minimum sampling count
DEF_DefaultOverlapSampling	64	Default overlap size
DEF_MinOverlapSampling	16	Minimum overlap size
DEF_MaxOverlapSampling	128	Maximum overlap size
DEF_FFT_NUM_BITREV	240	No. of elements in bit reverse table
DEF_SamplingRingBuffer	1024	Sampling ring buffer size
DEF_HeaderSize	7	Send data header byte size
DEF_SendAdMonitor	1	“AD converted value” send command
DEF_SendFrequencyMonitor	2	“Frequency spectrum” send command “
DEF_StartMonitor	1	“Monitor start” receive command
DEF_StopMonitor	2	“Monitor stop” receive command
DEF_SetUpSampling	3	“Sampling count setup” receive command
DEF_PcRecvBuffSize	128	Receive buffer size from PC
DEF_PcRecvTimeOut	100	Receive timeout time from PC (Unit: 10msec)
DEF_MovingAverageMinCount	1	AI inference moving average minimum count
DEF_MovingAverageMaxCount	100	AI inference moving average maximum count
DEF_DefaultMovingAverageMaxCount	20	Default AI inference moving average count
DEF_ShuntCurrentSampling	1	Current mode

6. Variables

Table 11 lists the global variables used for AI inference processing. The blue area indicates variables used for the FFT API functions in the DSP library.

Table 11. Global Variables (MotionSensor.c)

MotionSensor.c		
Type Name	**Variable Name**	**Description**
ST_SamplingRingBuffer	gv_SamplingRingBuffer	Ring buffer for A/D sampling
uint16_t	gv_SamplingCounter	A/D sampling counter
int16_t	gv_FrameSamplingEnd	Acquisition end position notification for 1 frame of A/D sampling buffer
int16_t	gv_AD0Bffer[(DEF_MaxSamplingCount * 2) - DEF_MinOverlapSampling]	A/D conversion result (U phase) buffer ｱ
int16_t	gv_AD1Bffer[(DEF_MaxSamplingCount * 2) - DEF_MinOverlapSampling]	A/D conversion result (V phase) buffer
int16_t	gv_AD2Bffer[(DEF_MaxSamplingCount * 2) - DEF_MinOverlapSampling]	A/D conversion result (W phase) buffer
int16_t	gv_RspiSendStatus	FTDI send status
int32_t	gs_fft_VecCplxMagi32[DEF_MaxSamplingCount/2]	Complex magnitude value storage buffer
cplxi32_t	gs_fft_buf[(DEF_MaxSamplingCount/2)]	Storage area for result of "R_DSP_FFT_i16ci32" function operation (sampling count/2)
vector_t	gs_fft_time = {DEF_MaxSamplingCount, NULL}	Argument for "R_DSP_FFT_i16ci32" function (displays input buffer)
vector_t	gs_fft_freq = {(DEF_MaxSamplingCount/2), gs_fft_buf}	Argument for "R_DSP_FFT_i16ci32" function (displays output buffer)
vector_t	gs_fft_mag = {(DEF_MaxSamplingCount/2), gs_fft_VecCplxMagi32}	Argument for "R_DSP_VecCplxMag_ci32i32" function (displays output buffer)
r_dsp_fft_t	gs_fft_handle = {DEF_MaxSamplingCount, 0};	FFT function handle
int16_t	gs_fft_twiddles[(DEF_MaxSamplingCount+(DEF_MaxSamplingCount/2))]	Rotation factor buffer
Int32_t	gs_fft_bitrev[DEF_FFT_NUM_BITREV]	Bit reverse buffer
float	gv_DecibelsBelowFullScale	A/D converter full scale value (unit: dB) Working buffer for frequency conversion calculation
float	gv_SquareSamplingHalf	Working buffer for frequency conversion calculation of (sampling count/2)^2 ｱ
int16_t	gv_MonitorStatus	Measurement start/stop status
ST_SamplingMonitor	gv_SamplingMonitor	Monitor send data (stores data such as frequency spectrum)
uint8_t	gv_RecvData	Receive data (data to confirm receive is valid at initialization)
ST_PcRecvRingBuffer	gv_PcRecvRingBuffer	Ring buffer for receive from PC
uint16_t	gv_PcRecvStatus	Status of receive from PC

uint8_t	gv_PcCommand	Receive command from PC
uint32_t	gv_PcRecvTimer	Monitor timer for receive from PC
ST_SamplingConditions	gv_SamplingConditions	Sampling conditions
uint16_t	gv_SamplingSize	Sampling buffer size
uint16_t	gv_Buffer2to1Overlap	Stores sampling buffer overlap sample count
uint16_t	gv_Frame1End	Stores end position of even frames
uint16_t	gv_Frame2End	Stores end position of odd frames
ST_MovingAverage	gv_MovingAverage	Working buffer for AI inference moving average calculation

The following lists structures used in the e-AI sample code that is run in this sample software. The items highlighted in gray are structures or member names that will be eliminated in the future.

```
/* A/D sampling buffer structures */
typedef struct {
  int16_t             m_Ad0Value; /* A/D conversion result (U phase) buffer */
  int16_t             m_Ad1Value; /* A/D conversion result (V phase) buffer */
  int16_t             m_Ad2Value; /* A/D conversion result (W phase) buffer */
} ST_SamplingRingData;

/* A/D ring buffer structures */
typedef struct {
  uint16_t              m_Write; /* Ring buffer write position */
  uint16_t              m_Read;   /* Ring buffer read position */
  ST_SamplingRingData       m_SamplingRingData[DEF_SamplingRingBuffer]; /* Ring
buffer */
} ST_SamplingRingBuffer;

/* Information structures (frame length 128 samples) */
typedef struct {
  float                 m_FrequencySpectrum[3][DEF_SamplingCount128/2]; /*
Stores frequency spectrum */
  float                 m_RotationSpeed;  /* Stores rotation speed */
  float                 m_PeekCurrent;    /* Stores peak current value */
  int8_t                m_AiResult;       /* Stores AI inference value (%)*/
  float                 m_AiMovingAverage;/* Stores AI inference value
(%)moving average  */
} ST_Sampling128;

/* Information structures (frame length 256 samples) */
typedef struct {
  float                 m_FrequencySpectrum[3][DEF_SamplingCount256/2];
  float                 m_RotationSpeed;
  float                 m_PeekCurrent;
  int8_t                m_AiResult;
  float                 m_AiMovingAverage;
} ST_Sampling256;

Information structures (frame length 512 samples)
typedef struct {
  float                 m_FrequencySpectrum[3][DEF_SamplingCount512/2];
  float                 m_RotationSpeed;
  float                 m_PeekCurrent;
  int8_t                m_AiResult;
  float                 m_AiMovingAverage;
} ST_Sampling512;

Information structures (frame length 1024 samples)
typedef struct {
  float                 m_FrequencySpectrum[3][DEF_SamplingCount1024/2];
  float                 m_RotationSpeed;
  float                 m_PeekCurrent;
  int8_t                m_AiResult;
  float                 m_AiMovingAverage;
} ST_Sampling1024;
```

```
/* Serial communication buffer structures and unions */
typedef struct {
   uint8_t              m_Header[DEF_HeaderSize]; /* Stores communication
header */
   uint8_t              m_Cmd; /* Communication control command, 01: A/D
conversion value, 02: FFT processed value */
   union {
      ST_SamplingRingData  m_SamplingData; /* Entity of 3 shunt current buffer
structure */
      ST_Sampling128    m_Sampling128; /* Entity of data structures */
      ST_Sampling256    m_Sampling256;
      ST_Sampling512    m_Sampling512;
      ST_Sampling1024   m_Sampling1024;
   } UN_Sampling;
} ST_SamplingMonitor;

/* Serial communication ring buffer structure */
typedef struct {
  uint16_t              m_Write; /* Ring buffer write position */
  uint16_t              m_Read; /* Ring buffer read position */
  uint8_t            m_RingData[DEF_PcRecvBuffSize]; /* Ring buffer */
} ST_PcRecvRingBuffer;

/* Moving average operation buffer structure */
typedef struct {
  uint16_t              m_MaxCount; /* Acquired data count */
  uint16_t              m_BufferFull;  /* Buffer full flag */
  uint16_t              m_Count; /* Moving average length */
  uint32_t              m_Sum;/* Moving average sum buffer */
  uint16_t              m_AiInference[DEF_MovingAverageMaxCount]; /* AI
inference value buffer */
} ST_MovingAverage;

/* Sampling conditions structure  */
typedef struct {
  uint16_t              m_SamplingFrequency; /* Sampling frequency */
  uint16_t              m_SamplingCount;  /* Sampling count */
  uint16_t              m_SamplingOverLap;   /* Overlap size */
  uint16_t              m_MovingAverageMaxCount;   /* Moving average maximum
count  */
  uint16_t              m_SamplingMode;   /* Smampling mode */
  uint32_t              m_CheckSum;    /* Checksum */
} ST_SamplingConditions;
```

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	June. 20, 2019	-	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/.

© 2019 Renesas Electronics Corporation. All rights reserved.