MCU Car Kit, Ver.5.1

Operation Test Manual (Version for RX62T)

This manual describes MCU car kit operation test procedures and supports the following boards:

- Motor drive board, Ver. 5
- Sensor board, Ver. 5

Version 1.00 [ANDTR100]
March 2014
Renesas MCU Car Rally Secretariat

Important Notice (Revision 1.2)

Copyright

- Copyright of this manual and its contents belongs to the Renesas MCU Car Rally Secretariat.
- This manual is protected under copyright law and international copyright conventions.

Prohibited Use

The user is prohibited from doing any of the following:

- Sale of the manual to a third party, or advertisement, use, marketing, or reproduction of the manual for purpose of sale
- Transfer or reauthorization to a third party of usage rights to the manual
- Modification or deletion of the contents of the manual, in whole or in part
- Translation into another language of the contents of the manual
- Use of the contents of the manual for a purpose that may pose a danger of death or injury to persons

Reprinting and Reproduction

Prior written permission from the Renesas MCU Car Rally Secretariat is required in order to reprint or reproduce this manual.

Limitation of Liability

Every effort has been made to ensure the accuracy of the information contained in this manual. However, the Renesas MCU Car Rally Secretariat assumes no responsibility for any loss or damage that may arise due to errors this manual may contain.

Other

The information contained in this manual is current as of the date of publication. The Renesas MCU Car Rally Secretariat reserves the right to make changes to the information or specifications contained in this manual without prior notice. Make sure to check the latest version of this manual before starting fabrication.

Contact Information

MCU Car Rally Secretariat, Renesas Solutions Corp.
MN Building, 2-1 Karuko-saka, Ageba-cho, Shinjuku-ku, Tokyo, 162-0824, Japan
Tel. (03) 3266-8510
E-mail: official@mcr.gr.jp

All trademarks and registered trademarks are the property of their respective owners.

Contents

1. Outline

This manual describes operation test procedures for MCU cars fabricated and assembled according to the following manuals:

• Motor Drive Board, Ver. 5, Assembly Manual
• Sensor Board, Ver. 5, Assembly Manual
• MCU Car Kit, Ver. 5.1, Body Assembly Manual

Operation testing involves the following procedures:

1. Installing the Renesas integrated development environment on a PC
Install the Renesas integrated development environment (C/C++ Compiler Package for RX Family) on the PC. For details, please refer to "Renesas Integrated Development Environment Installation Manual (Version for RX62T)".

2. Installing the sample program (operation test program)
Install the sample program for the RX62T (workspace_kit12_rx62t_eng.exe) on the PC.
This procedure is described later in this manual.

3. Writing the operation test program to the RMC-RX62T board of the MCU car
This procedure is described later in this manual.

4. Running the operation tests as described in this manual
This procedure is described later in this manual.

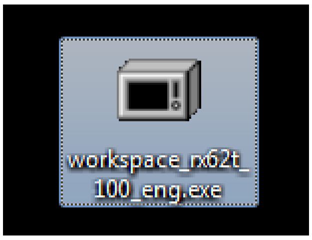
5. Adjusting the servo centre and maximum turn angle
Make the necessary adjustments by following the instructions in section 6, Adjusting the Servo Centre and Maximum Turn Angle, in *MCU Car Kit, Ver. 5.1, Program Explanation Manual—kit12_rx62t Version (Version for RX62T)*.

6. Writing the running program to the RMC-RX62T board of the MCU car
Write the kit12_rx62t.mot file of project kit12_rx62t to the RMC-RX62T board by following the instructions in *MCU Car Kit, Ver. 5.1, Program Explanation Manual—kit12_rx62t Version (Version for RX62T)*, then try running the MCU car on the course.

2. Installing the program

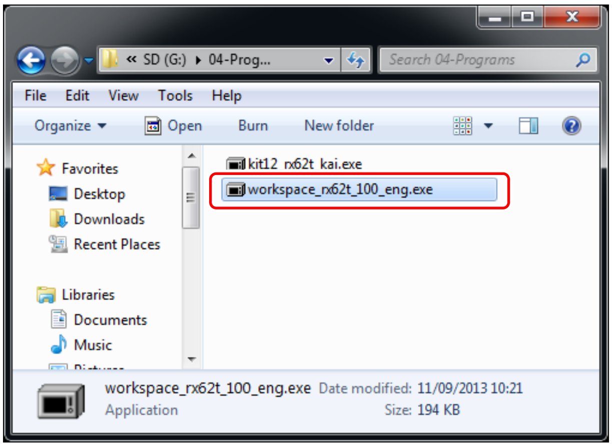
Please advance to the "2.2 Install of program" chapter, if you have CD-R for this seminar.

2.1. Download of programs

1		Get the Sample Program (workspace_rx62t_100_eng.exe) from the Renesas site. Renesas Electronics **http://www.renesas.com/company_info/carrally/** Click **Download**
2		Download **workspace_kit12_rx62t.exe**
3		Downloading is complete.

2.2. Install of program

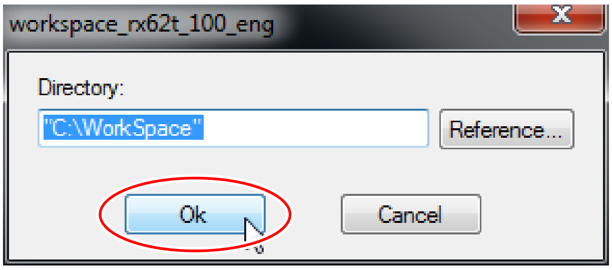
1



Run **workspace_rx62t_100_eng.exe**.

Please execute "workspace_rx62t_100_eng.exe" in the following folder, if you have CD-R for this seminar.
"CD drive:¥01-Softwares"

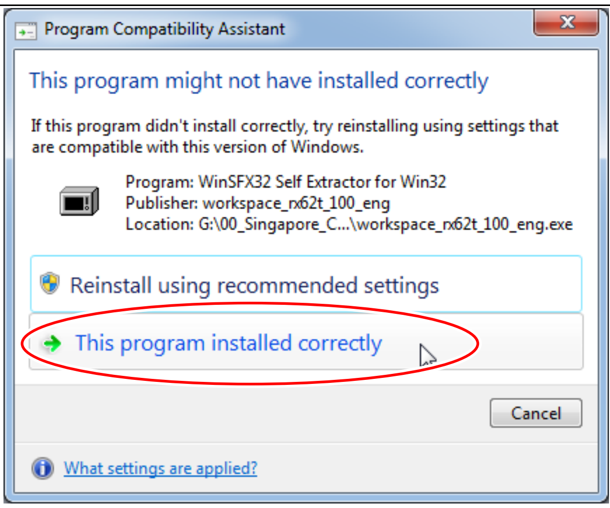
2



The installed file is in C:¥WorkSpace.

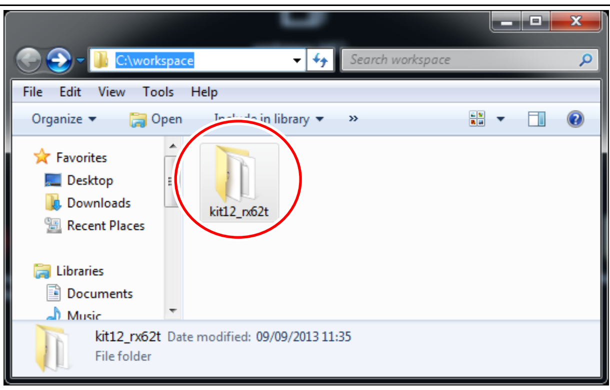
Click OK.

3



Installation has been completed.
Click This program installed correctly.

4

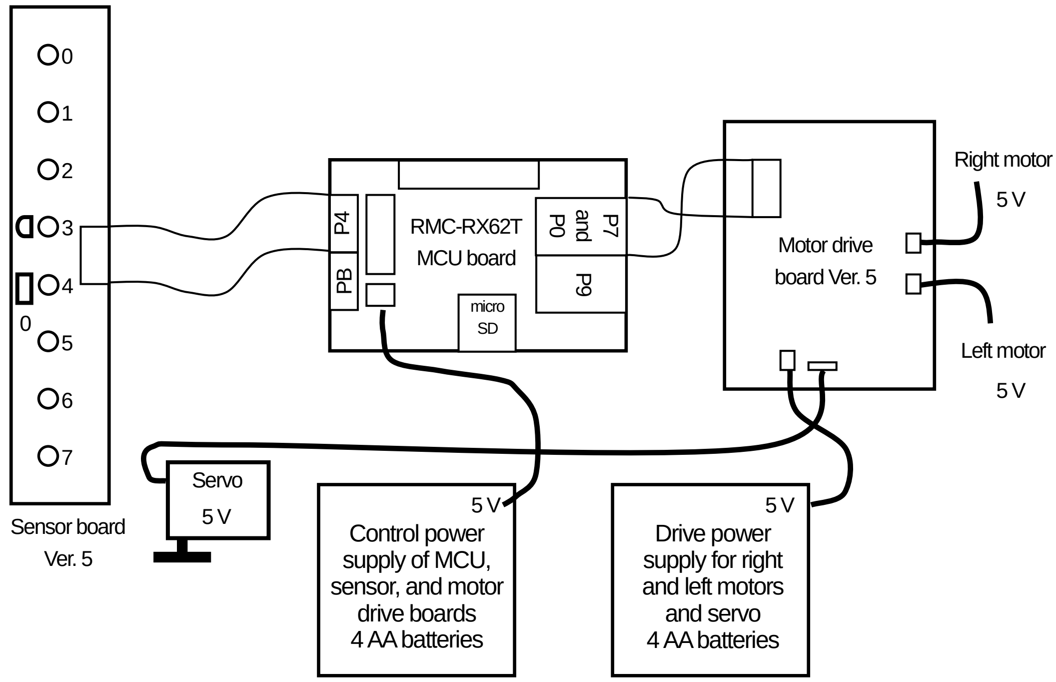


Open The "C:¥Workspace" folder

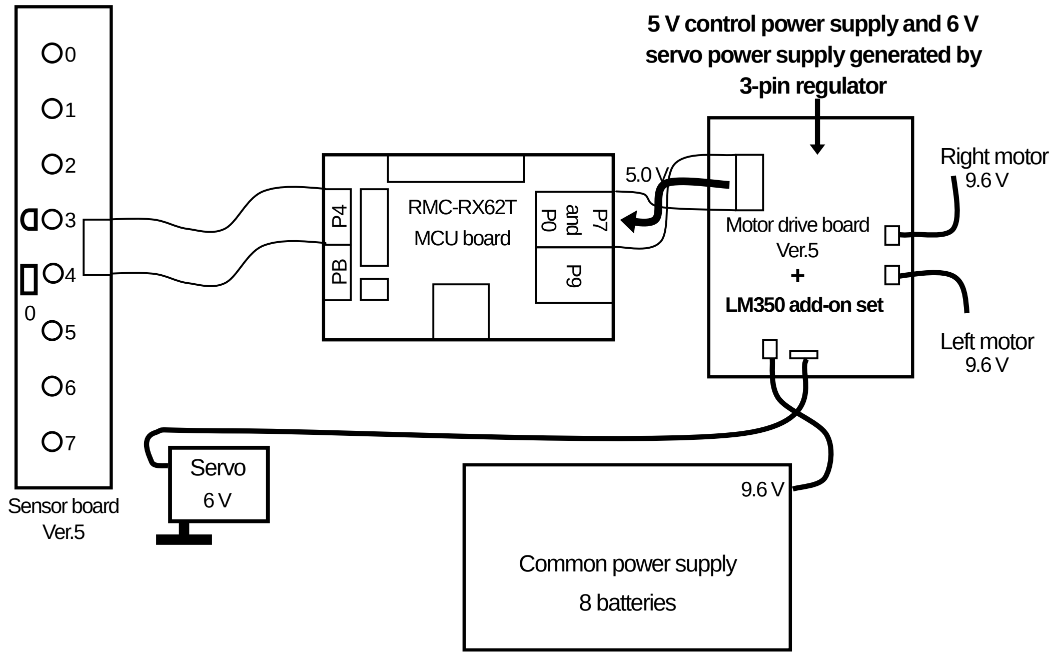
There is the operation test program at the folder "kit12_rx62t".

3. Components of MCU Car Kit

This manual describes the operation test procedures for a MCU car consisting of the components of the MCU car kit, Ver. 5.1. These components are shown below.

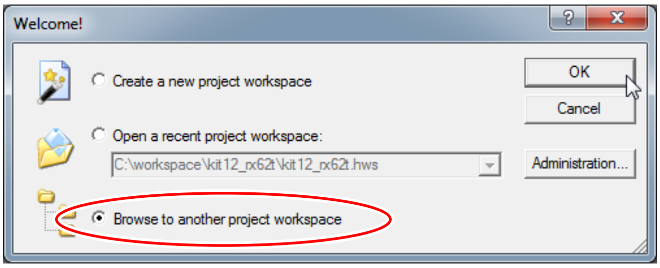
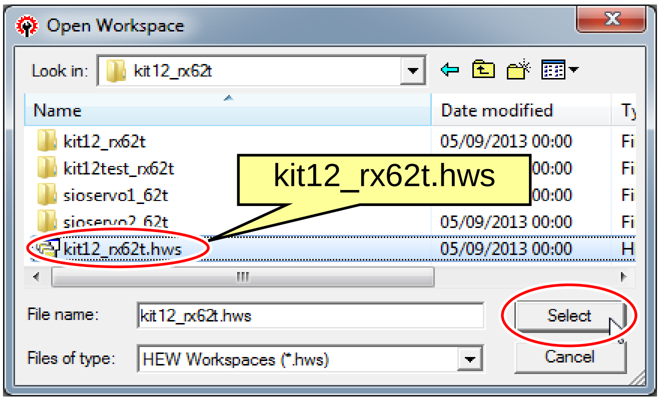
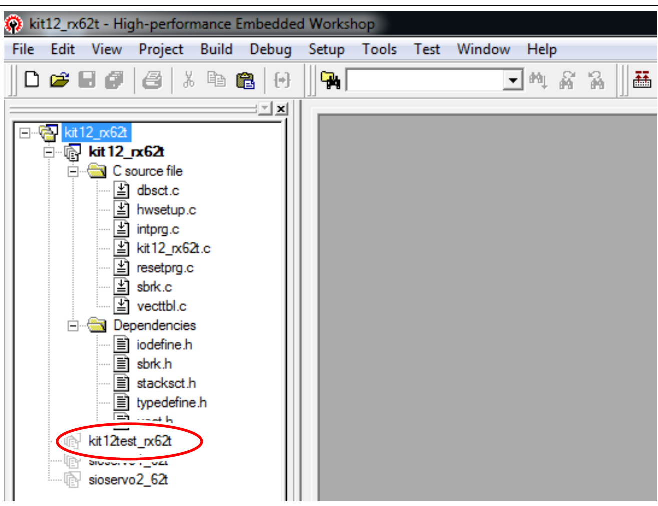


The same operation test procedures may be used for MCU cars with the LM350 add-on set installed on the motor drive board. The configuration with this addition is shown below.

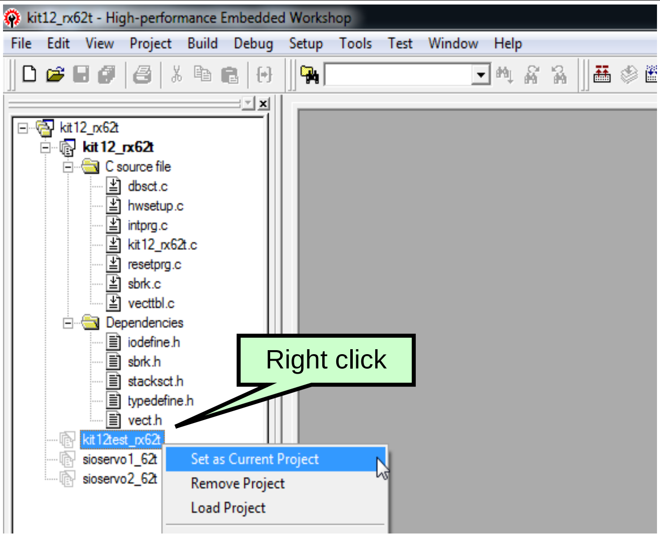
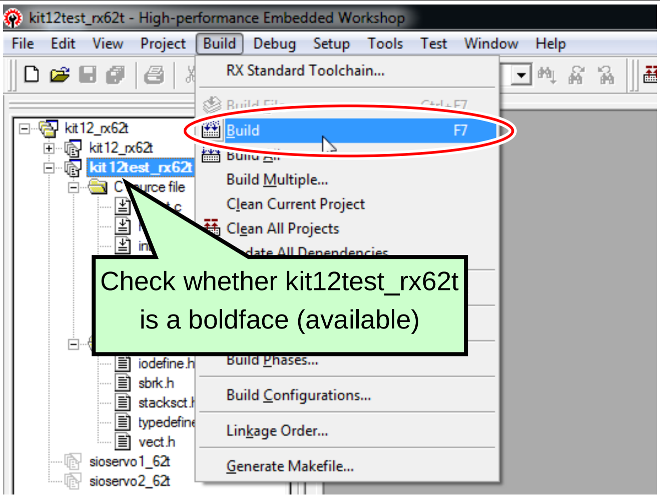
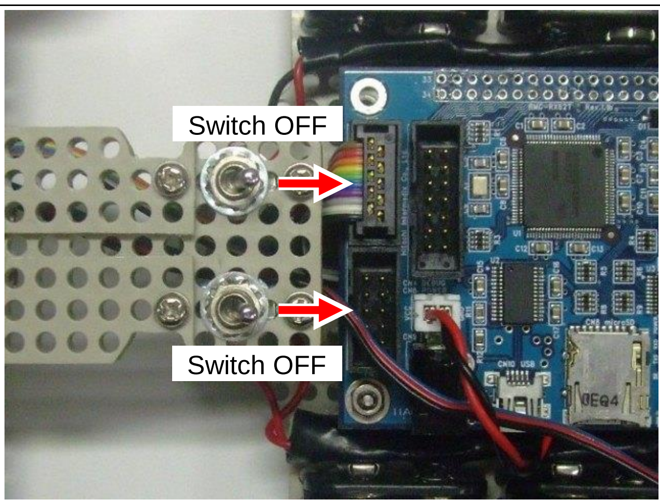


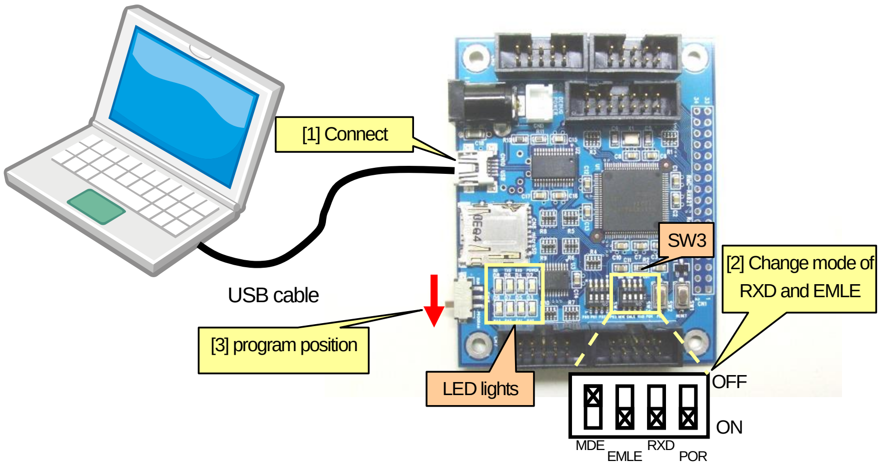
4. Writing the Operation Test Program to the MCU Board of the MCU Car

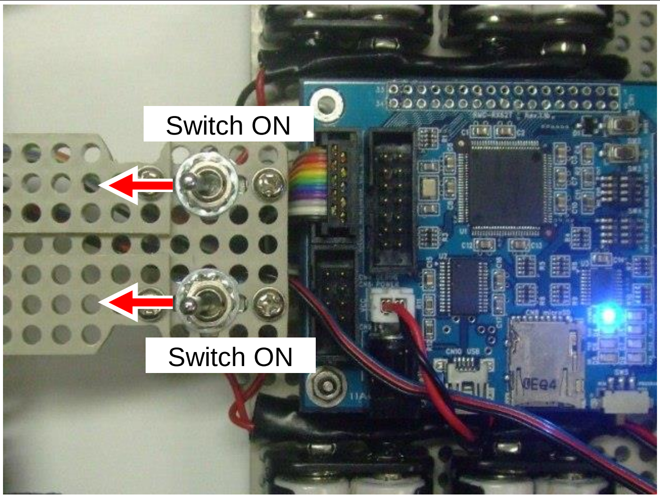
4.1. Opening the kit12_rx62t Workspace

1		Launch the High-performance Embedded Workshop.
2		Select **Browse to another project workspace** Click OK.
3		Select **kit12_rx62t.hws** from the "C:¥Workspace¥kit12_rx62t" folder
4		The kit12_rx62t workspace opens. This workspace contains four projects. ●kit12_rx62t This is the MCU car running program. ●**kit12test_rx62t** In the project described here, this program is used to confirm that the motor drive board and sensor board are operating properly. ●sioservo1_62t This is a program for adjusting the servo centre. ●sioservo2_62t This is a program for determining the servo’s maximum turn angle.

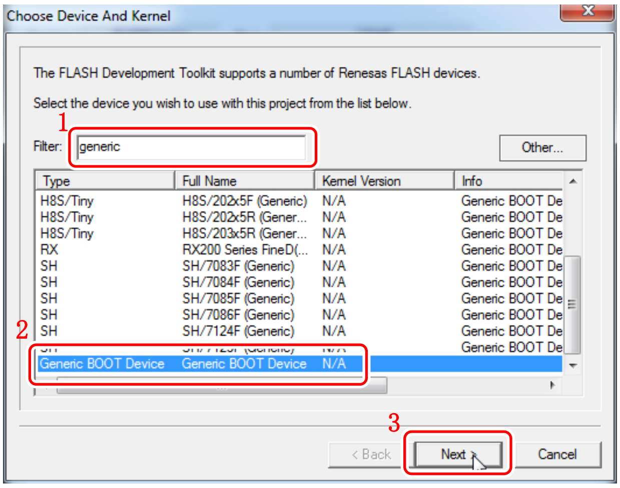
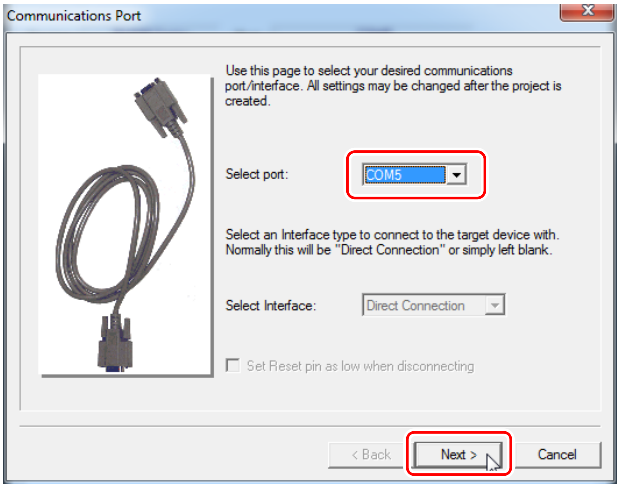
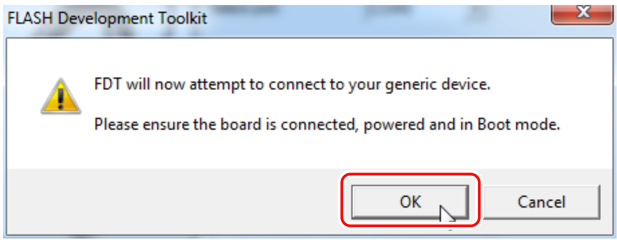
4.2. Writing the Operation Test Program to the MCU Board

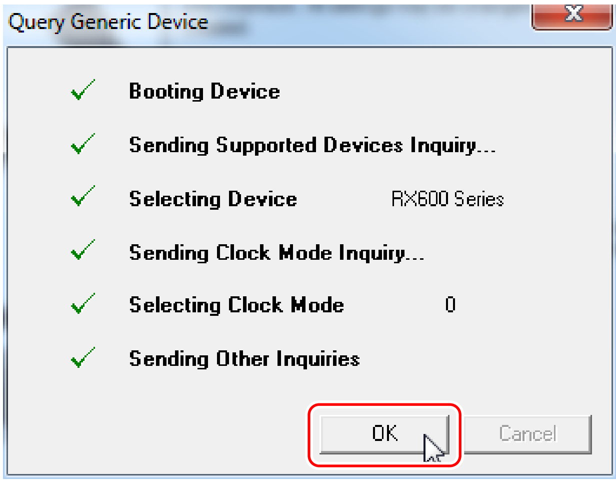
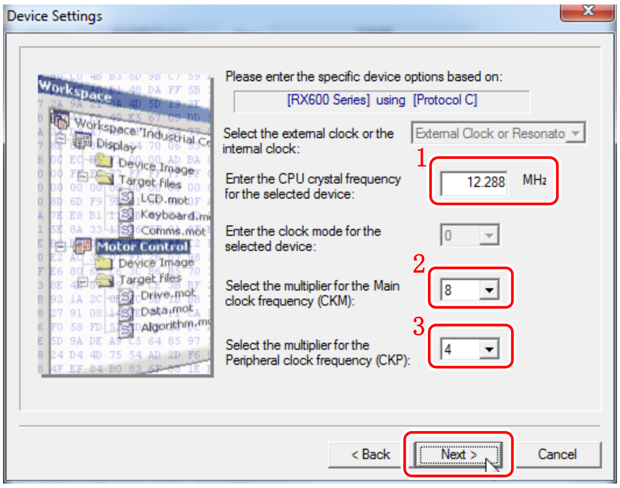
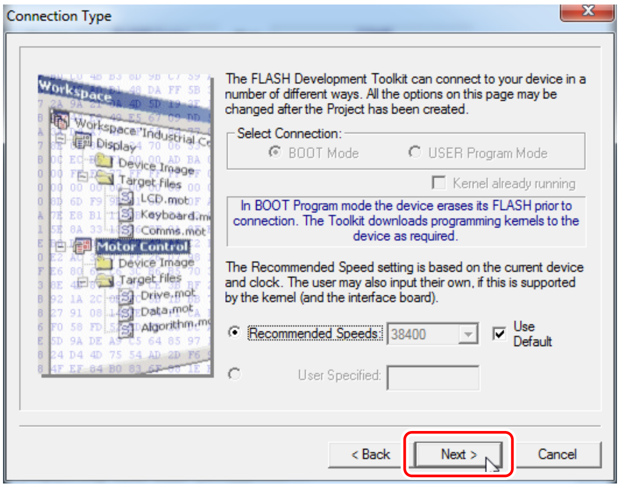
<table>
<tr><td>1</td><td>
</td><td>Set kit12test_rx62t as the current project. Right click on kit12test_rx62t and select Set as Set as Current Project from the menu. The item kit12test_rx62t is displayed in bold.</td></tr>
<tr><td>2</td><td>
</td><td>Select Build > Build menu to create a MOT file.</td></tr>
<tr><td>3</td><td>
</td><td>Move the two power switches of the MCU car to the off position.</td></tr>
</table>

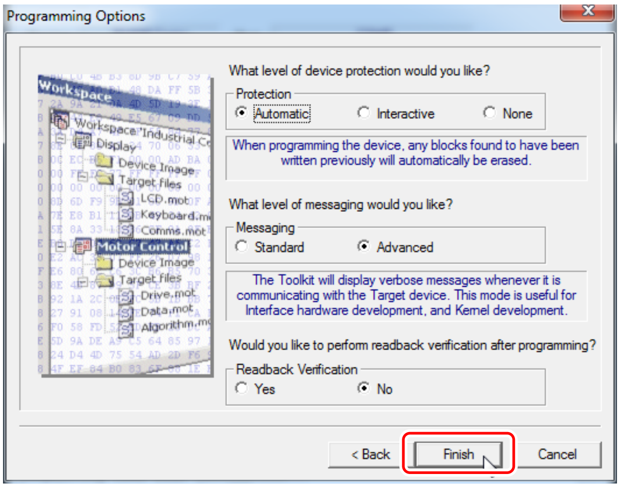
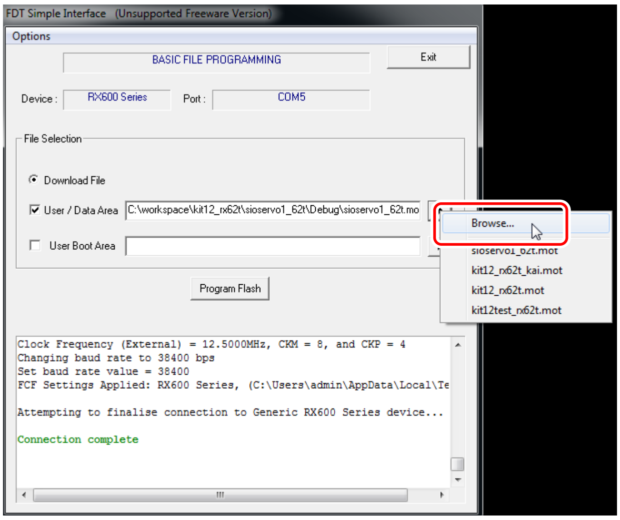
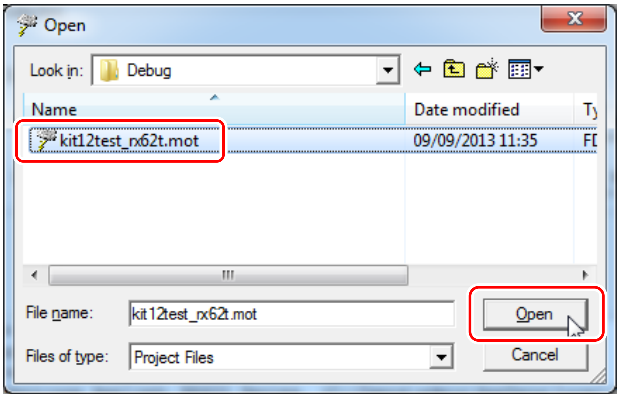
4		[1] Connect PC and MCU board by USB cable. [2] Turn on RXD and EMLE of SW3. [3] Turn SW5 to PROGRAM. SW5 must change states while the power is off.

5		Move the two power switches of the MCU car to the on position.

6		Start up **Flash Development Toolkit 4.09 Basic**. Click its shortcut icon on the desktop.

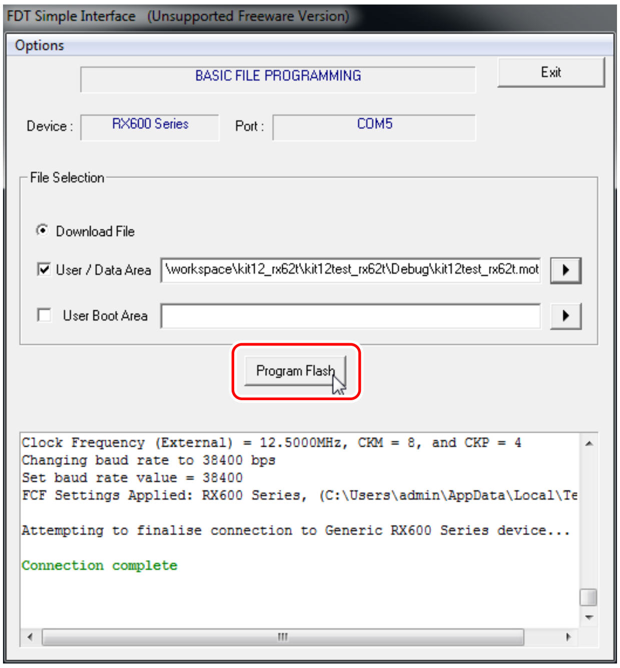
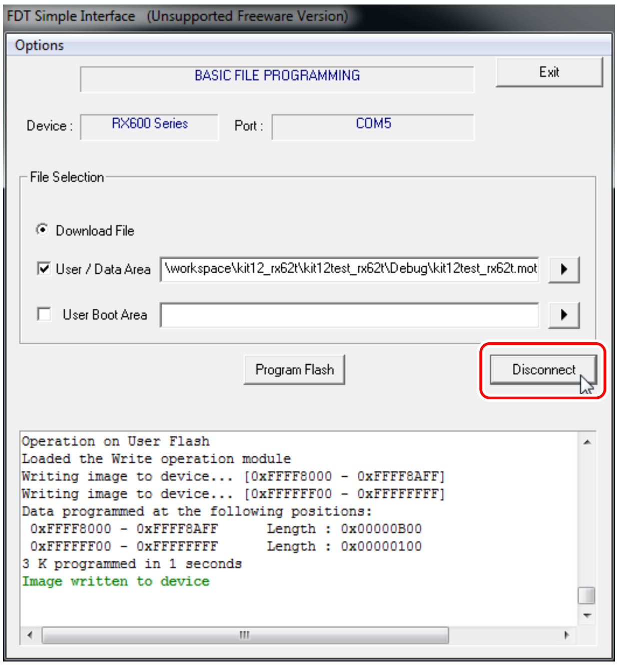
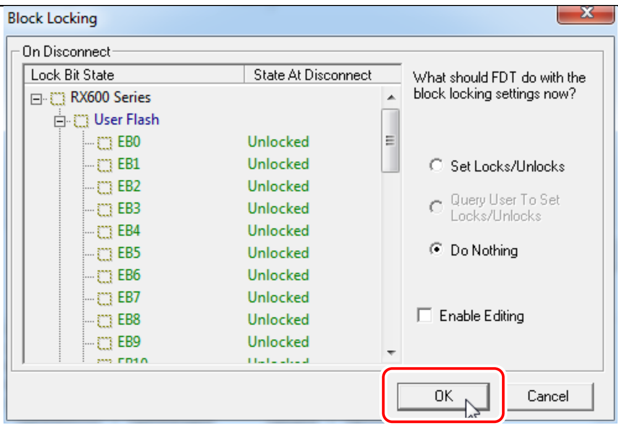
7		Write "generic" in the Filter field. Then choose **Generic BOOT Device**. Click Next.
8		Choose the COM number. Click Next.
9		Click OK.

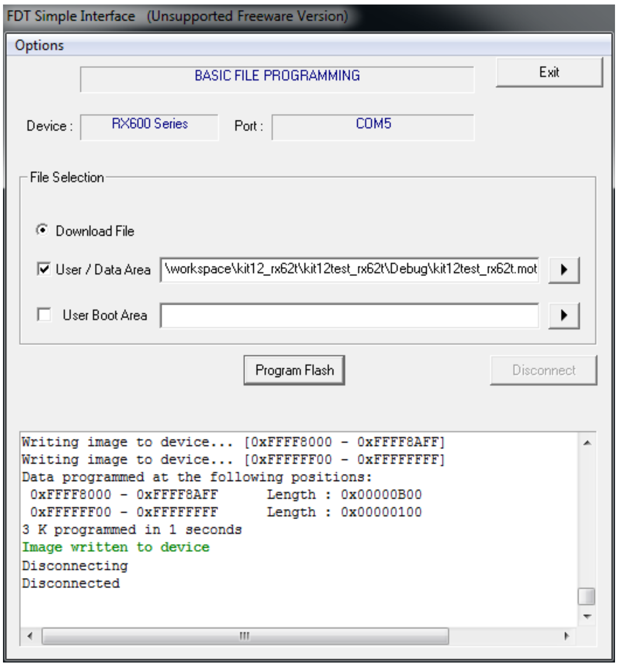
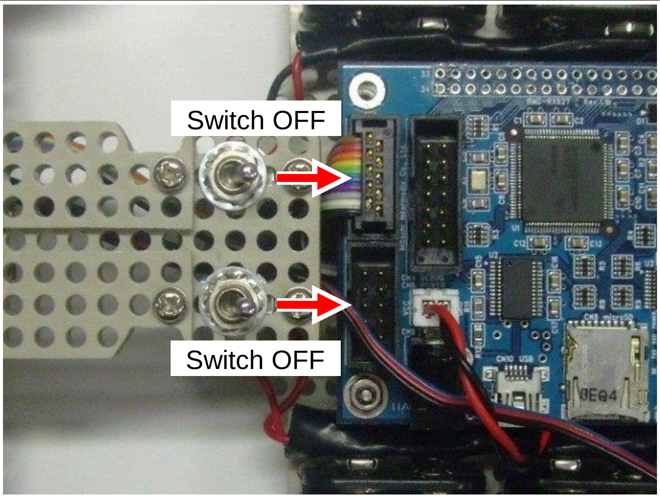
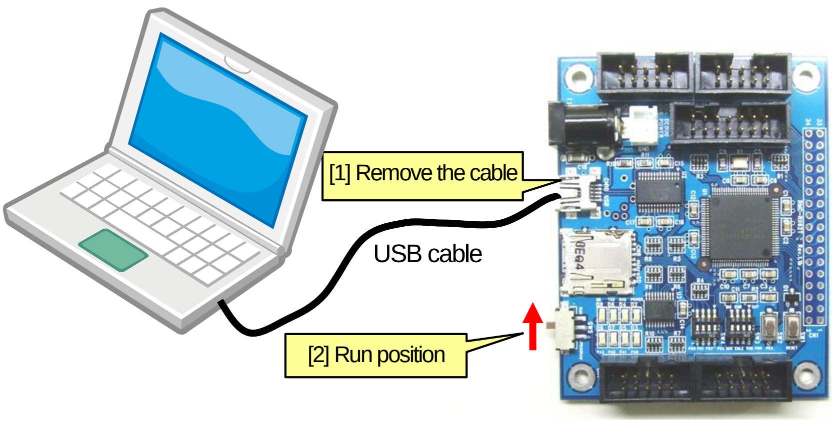
10		Click OK. If an error occurs, please try again.
11		Write 12.288 for CPU crystal frequency. Choose 8 for CKM. Choose 4 for CKP. Click Next.
12		Click Next.

<table>
<tr>
<td>13</td>
<td>
</td>
<td>Click Finish.</td>
</tr>
<tr>
<td>14</td>
<td>
</td>
<td>Check User/Data Area.
Then click the triangle on the far right and click Browse.</td>
</tr>
<tr>
<td>15</td>
<td>
</td>
<td>Open the kit12test_rx62t.mot file.

The kit12test_rx62t.mot file is found in the below folder.
"C:¥WorkSpace¥kit12_rx62t¥kit12test_rx62t¥Debug"

Click Open.</td>
</tr>
</table>

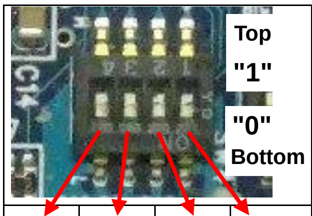
<table>
<tr><td>16</td><td>
</td><td>Click [Program Flash].
Then program writing will begin.</td></tr>
<tr><td>17</td><td>
</td><td>After programming has finished, click [Disconnect].</td></tr>
<tr><td>18</td><td>
</td><td>Click [OK].</td></tr>
</table>

19		Program writing completed.
20		Move the two power switches of the MCU car to the off position.
21		When the programming operation has completed, run the program using the following procedure. [1] The USB cable may be left connected or may be disconnected. (However, it should be disconnected if the MCU car will be operated.) [2] Turn SW5 to RUN.

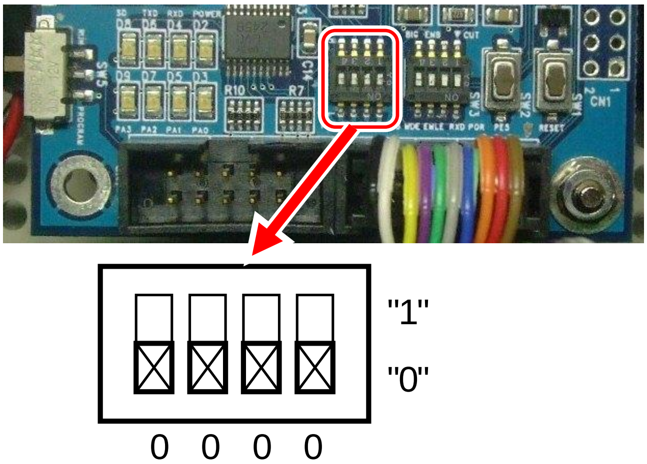
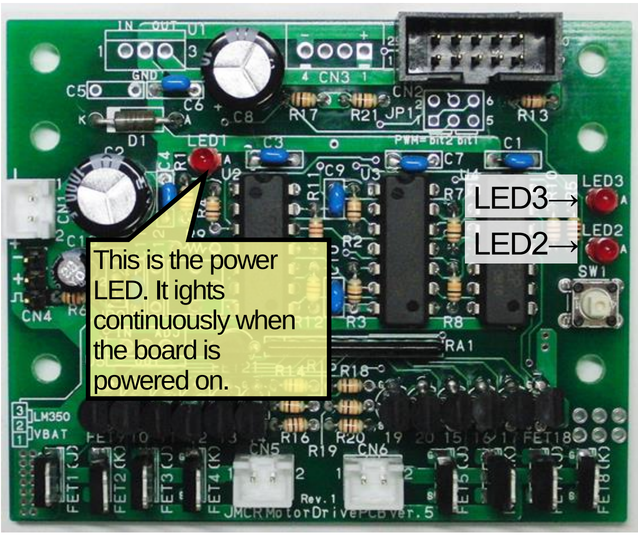
5. Operation Tests

5.1. List of Operation Tests

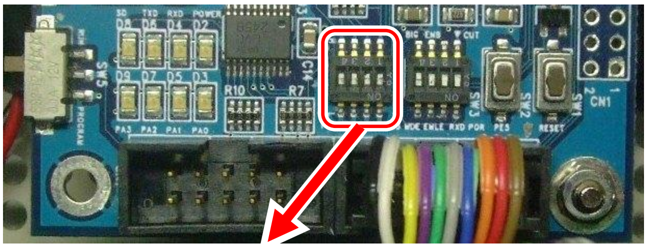
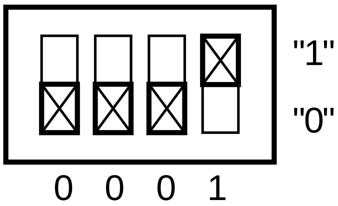
You can select and perform tests on different parts of the MCU car by changing the settings of the DIP switches on the RMC-RX62T board.

				**Description**
P1_3	**P1_2**	**P1_1**	**P1_0**	
0	0	0	0	Tests the LEDs. The LEDs light alternately at 0.5-second intervals.
0	0	0	1	Tests the pushbutton. LED3 lights when the button is in the off position and LED2 lights when it is in the on position.
0	0	1	0	Performs the servo operation test. The servo cycles through the 0°, 30° right, and 30° left positions repeatedly.
0	0	1	1	Does nothing.
0	1	0	0	Performs the right motor operation test. The motor switches between forward and brake operation repeatedly.
0	1	0	1	Performs the right motor operation test. The motor switches between reverse and brake operation repeatedly.
0	1	1	0	Performs the left motor operation test. The motor switches between forward and brake operation repeatedly.
0	1	1	1	Performs the left motor operation test. The motor switches between reverse and brake operation repeatedly.
1	0	0	0	Performs the sensor board bit1 and bit0 operation test. The states of sensor bits 1 and 0 are output to LED2 and LED3. ※bit0 is a start bar detection sensor and a combined use.
1	0	0	1	Performs the sensor board bit3 and bit2 operation test. The states of sensor bits 3 and 2 are output to LED2 and LED3.
1	0	1	0	Performs the sensor board bit5 and bit4 operation test. The states of sensor bits5 and 4 are output to LED2 and LED3.
1	0	1	1	Performs the sensor board bit7 and bit6 operation test. The states of sensor bits 7and 6 are output to LED2 and LED3.
1	1	0	0	Confirms the MCU car’s ability to run straight forward. The car advances at PWM 50% and then stops after 2 seconds.
1	1	0	1	Confirms the MCU car’s ability to run straight forward. The car advances at PWM 50% and then stops after 5seconds.
1	1	1	0	Confirms the MCU car’s ability to run straight forward. The car advances at PWM 100 % and then stops after 2 seconds.
1	1	1	1	Confirms the MCU car’s ability to run straight forward. The car advances at PWM 100 % and then stops after 5 seconds.

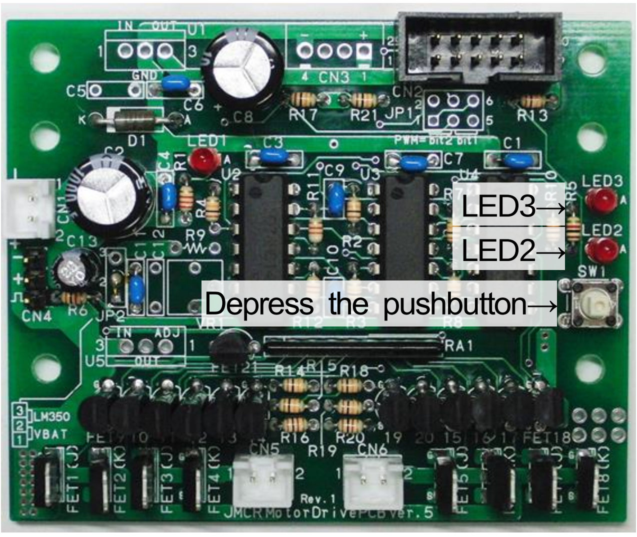
5.2. LED Operation Test

1		This test checks the operation of the LEDs on the motor drive board. With the DIP switches set to 0000, move the two power switches of the MCU car to the on position.
2		LED2 and LED3 on the motor drive board light alternately at 0.5-second intervals. After the operation test has finished, slide the two power switches to the off position. If LED2 or LED3 does not light, possible causes include a fault in the flat cable connecting the RMC-RX62T board and motor drive board, solder bridging (shorting), and incorrect LED mounting orientation. Identify the problem visually or using a tester and correct it.

5.3. Pushbutton Operation Test

<table>
<tr>
<td>1</td>
<td>



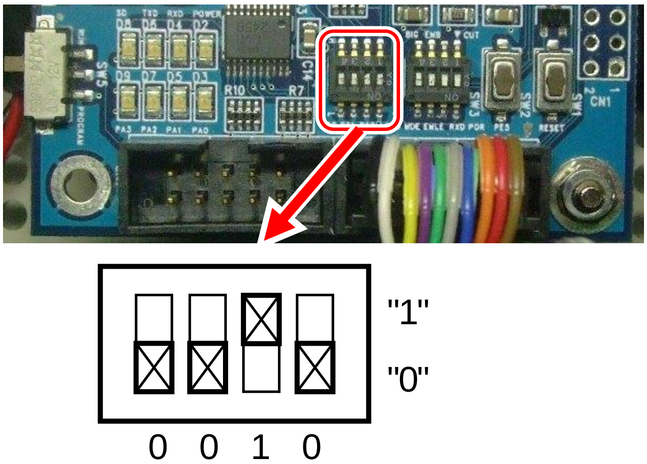
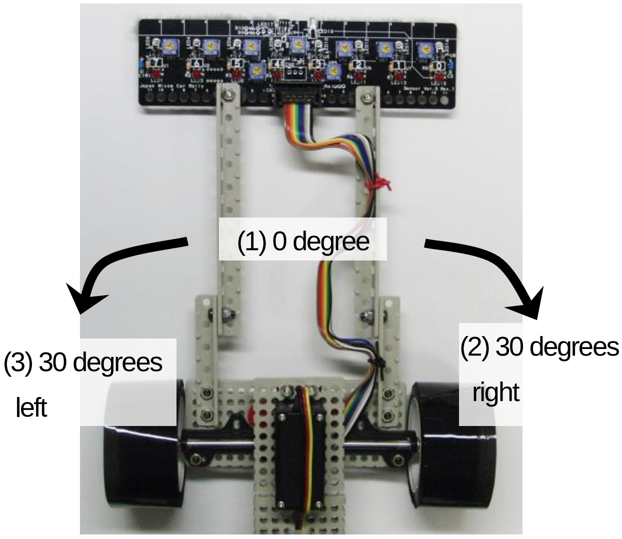
</td>
<td>This test checks the response of the pushbutton on the motor drive board.
With the DIP switches set to 0001, move the two power switches of the MCU car to the on position.</td>
</tr>
</table>

<table>
<tr>
<td>2</td>
<td>


</td>
<td>LED3 lights when the pushbutton on the motor drive board is not pressed and LED2 lights when it is pressed. After the operation test has finished, slide the two power switches to the off position.

If only LED3 lights, possible causes include a soldering fault in the circuit leading to the switch. If LED2 is lit constantly, possible causes include solder bridging. Identify the problem visually or using a tester and correct it.</td>
</tr>
</table>

5.4. Servo Operation Test

<table>
<tr>
<td>1</td>
<td>
</td>
<td>This test checks the operation of the servo connected to the motor drive board. The area around the MCU car should be kept free of obstacles during this test because the front portion of the car moves back and forth. With the DIP switches set to 0010, move the two power switches of the MCU car to the on position.</td>
</tr>
<tr>
<td>2</td>
<td>
</td>
<td>The servo repeatedly cycles through the 0°, 30° right, and 30° left positions at one-second Intervals. After the operation test has finished, slide the two power switches to the off position.

If the servo does not operate, possible causes include a soldering fault in the circuit leading to the servo and incorrect mounting orientation of the servo connector. Also check whether the power LED on the motor drive board is lit. Identify the problem visually or using a tester and correct it.</td>
</tr>
<tr>
<td>3</td>
<td>

```
470
471  /************************************************************
472  /* Servo steering operation
473  /* Arguments:    servo operation angle: -90 to 90
474  /*               -90: 90-degree turn to left, 0: straight,
475  /*                90: 90-degree turn to right
476  /************************************************************
477  void handle( int angle )
478  {
479      /* When the servo move from left to right in reverse, re
480      MTU3.TGRD = SERVO_CENTER - angle * HANDLE_STEP;
481  }
482
483  /***********************            ***********************
484  /* end of file
485  /***********************            ***********************
```

Change − to ＋

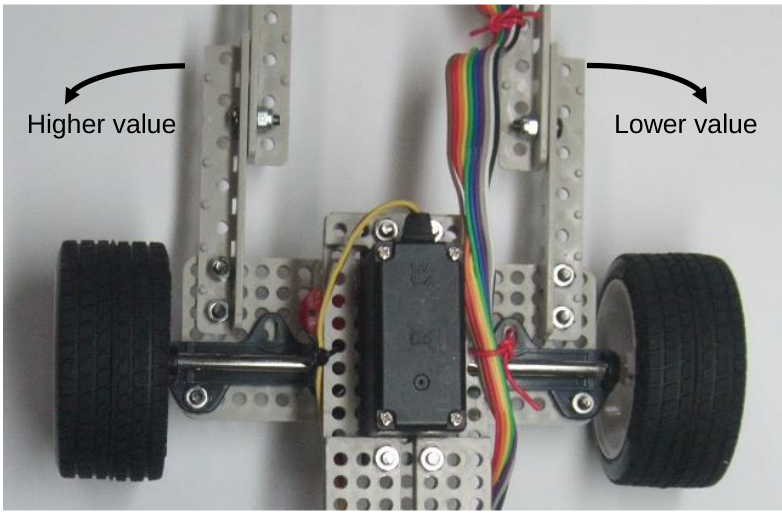
</td>
<td>If the servo supplied with the kit is replaced with a different servo, and the resulting sequence is 0° → 30° left → 30° right, the servo is a model that reverses the right and left turn operations. If this is the case, change the − (minus sign) to a＋(plus sign) in line 480 of the handle function in kit12test_38a.c. This will reserve right and left, resulting in the correct sequence of 0° → 30° right→30° left.</td>
</tr>
</table>

Note: Servo centre adjustment

For the MCU car to operate properly, the servo angle must be 0° at initial power-on. In most cases, however, the angle is not exactly 0° when power is turned on. Since each servo has its own centre value (the value at which the wheels are oriented straight ahead), the value must be adjusted individually for each MCU car. To adjust, change the value of SERVO_CENTER in kit12test_rx62t.c from 2300 as necessary to achieve the true centre position. A difference of 13 is equivalent to about one degree of change in the position.

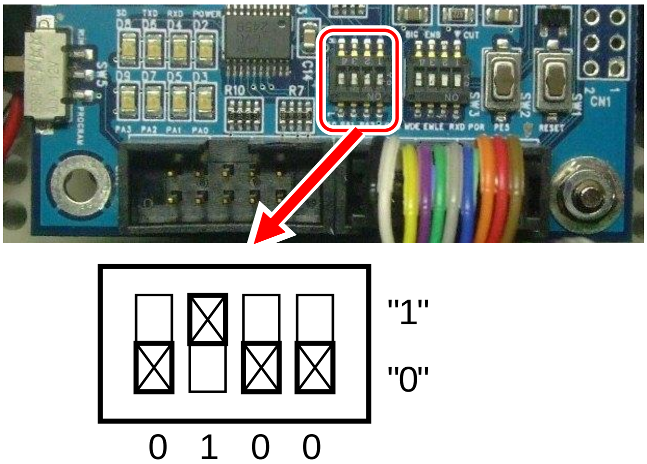
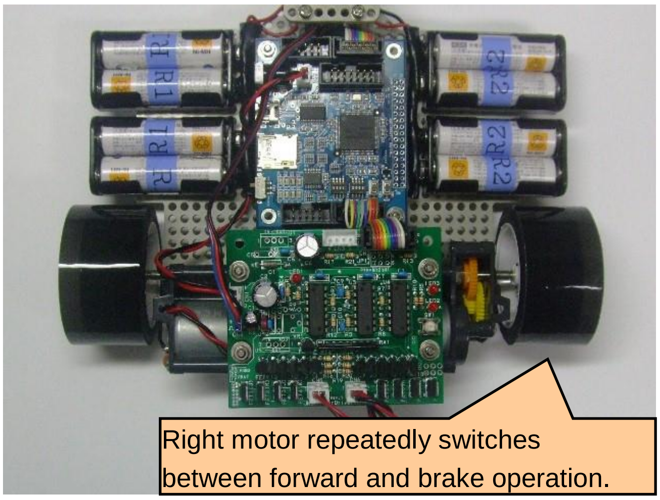
```
54 : #define      SERVO_CENTER   2300   /*  Servo centre value      */
```

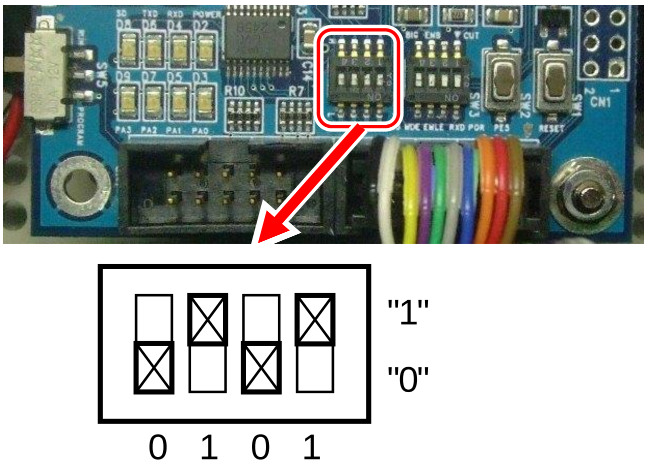
Increasing the value turns the wheels to the left (moving forward), and reducing the value turns the wheels to the right.



The instructions in section 7, Adjusting the Servo Centre and Maximum Turn Angle, in *MCU Car Kit, Ver. 5.1, Program Explanation Manual—kit12_rx62t Version (Version for RX62T)* are useful when adjusting the servo centre value.

5.5. Right Motor Operation Test

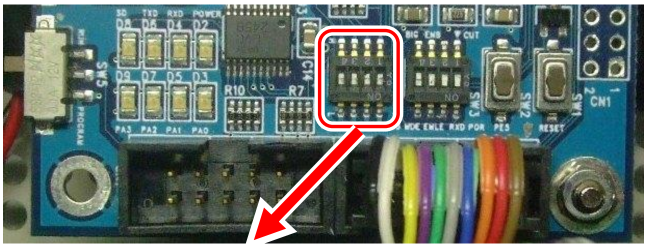
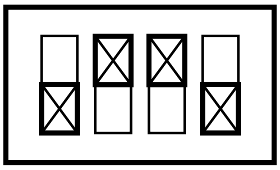
<table>
<tr>
<td>1</td>
<td>
</td>
<td>This test checks the forward, reverse, and brake operation of the right motor. Since the motor operates (and the wheels move), lift the MCU car so that the tires are not making contact with the floor when running this test. With the DIP switches set to 0100, move the two power switches of the MCU car to the on position.</td>
</tr>
<tr>
<td>2</td>
<td>
</td>
<td>The right motor repeatedly switches between forward and brake operation at one-second intervals. After the operation test has finished, slide the two power switches to the off position.

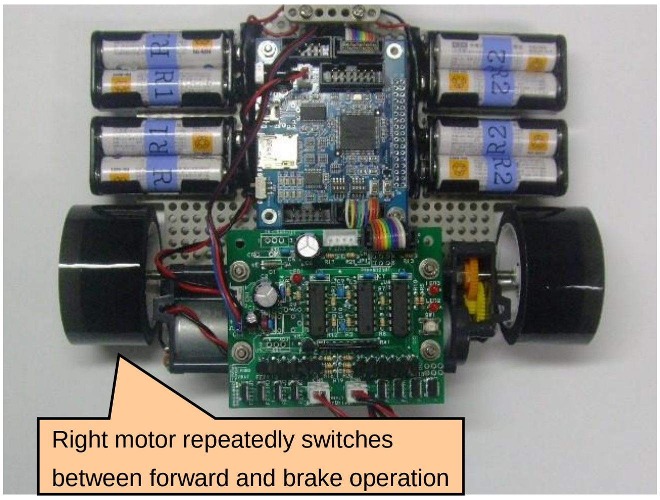
If the right motor does not operate in the forward direction, possible causes include a soldering fault in the right motor control circuit. If the right motor turns constantly (does not switch to brake operation), possible causes include solder bridging. Identify the problem visually or using a tester and correct it. If the wheel turns in the wrong direction, the motor cable connection is reversed.
If the wheel does not turn, possible causes include solder bridging. Switch the positions of pins 1 and 2 of the connector.</td>
</tr>
<tr>
<td>3</td>
<td>
</td>
<td>With the DIP switches set to 0101, move the two power switches of the MCU car to the on position.

The right motor repeatedly switches between reverse and brake operation at one-second intervals. After the operation test has finished, slide the two power switches to the off position.

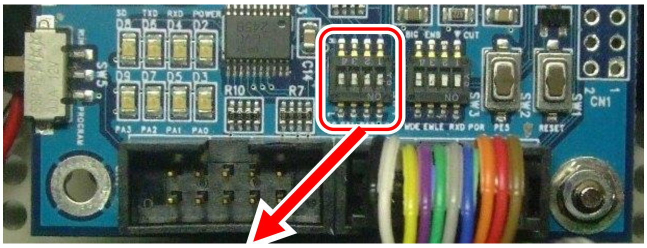
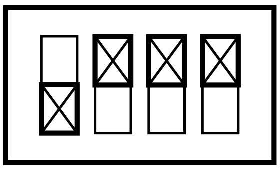
If the right motor does not operate in the reverse direction, possible causes include a soldering fault or a short. Identify the problem visually or using a tester and correct it.</td>
</tr>
</table>

5.6. Left Motor Operation Test

<table>
<tr><td>1</td><td>

</td><td>This test checks the forward, reverse, and brake operation of the left motor. Since the motor operates (and the wheels move), lift the MCU car so that the tires are not making contact with the floor when running this test. With the DIP switches of the MCU car to the on position.</td></tr>
</table>

<table>
<tr><td>2</td><td>
</td><td>The left motor repeatedly switches between forward and brake operation at one-second intervals. After the operation test has finished, slide the two power switches to the off position.

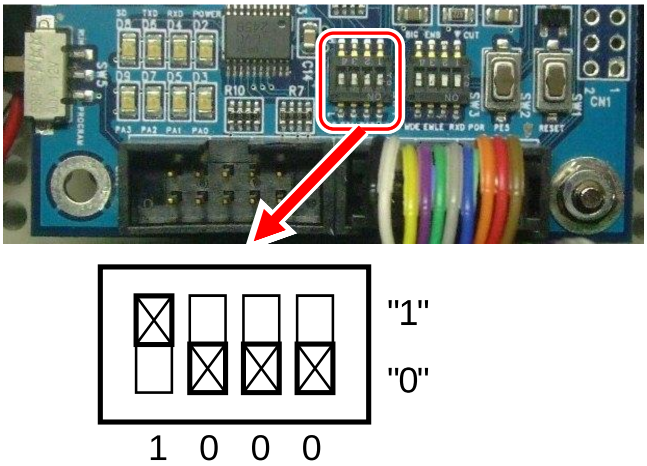
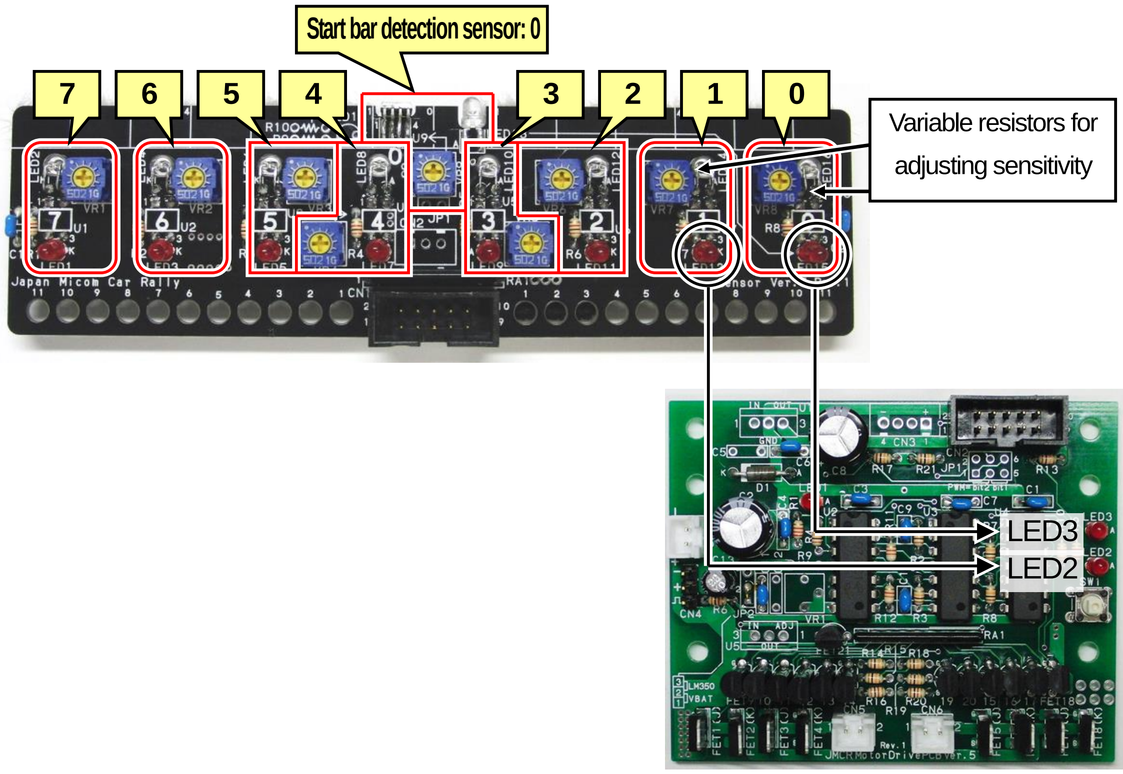
If the left motor does not operate in the forward direction, possible causes include a soldering fault in the left motor control circuit. If the left motor turns constantly (does not switch to brake operation), possible causes include solder bridging. Identify the problem visually or using a tester and correct it.
If the wheel turns in the wrong direction, the motor cable connection is reversed. If the wheel does not turn, possible causes include solder bridging. Switch the positions of pins 1 and 2 of the connector.</td></tr>
</table>

<table>
<tr><td>3</td><td>

</td><td>With the DIP switches set to 0111, move the two power switches of the MCU car to the on position.

The left motor repeatedly switches between reverse and brake operation at one-second intervals. After the operation test has finished, slide the two power switches to the off position.

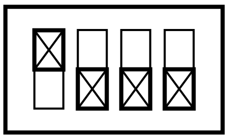
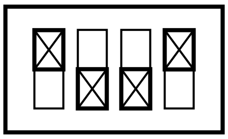
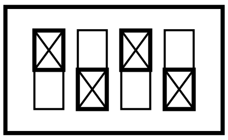
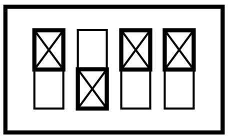
If the left motor does not operate in the reverse direction, possible causes include a soldering fault or a short. Identify the problem visually or using a tester and correct it.</td></tr>
</table>

5.7. Sensor Board Operation Test

<table>
<tr><td>1</td><td>
</td><td>With the DIP switches set to 1000, move the two power switches of the MCU car to the on position.</td></tr>
<tr><td rowspan="2">2</td><td colspan="2">
</td></tr>
<tr><td colspan="2">The states of sensors 0 and 1 on the sensor board, shown in the photo above, are output to the two LEDs on the motor drive board. The LEDs on the sensor board also light, so this test allows you to confirm that they respond identically. The sensitivity of the sensors can be adjusted by using the variable resistors.
If the LEDs on the sensor board do not light, possible causes include faulty soldering on the sensor board, solder bridging, and incorrect part mounting orientation. If the LEDs on the sensor board light but the LEDs on the motor drive board do not, there may be a problem related to the connector. Identify the problem visually or using a tester and correct it.

Note: The start bar detections sensor is output to LED (LED15) of sensor 0. Test the sensor when the DIP switch is 1000 (this time).</td></tr>
</table>

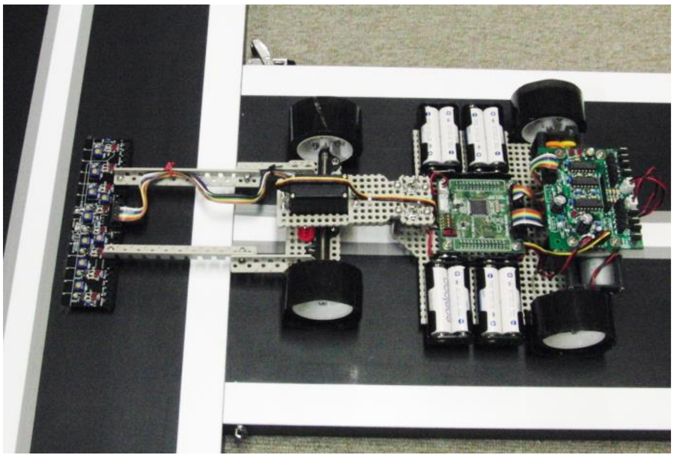
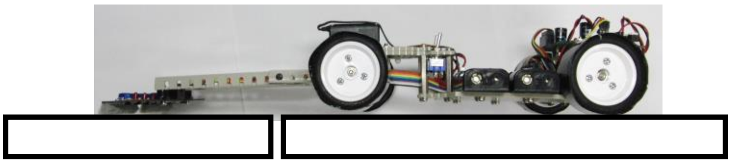
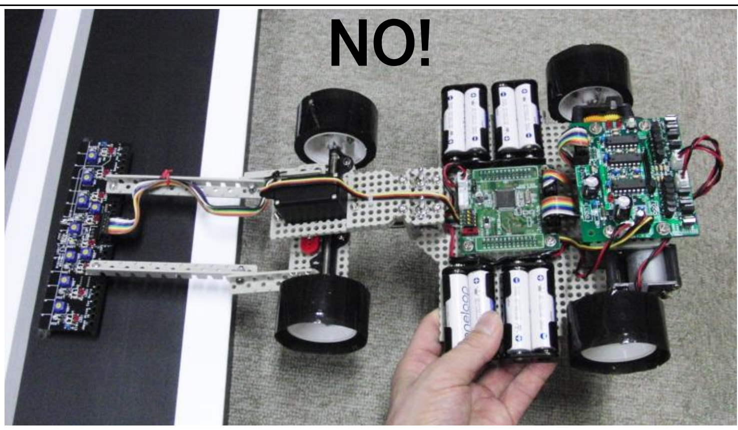
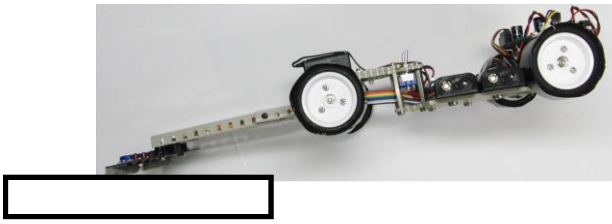
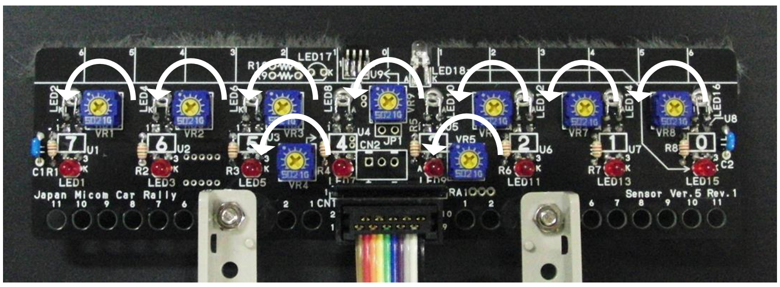
3

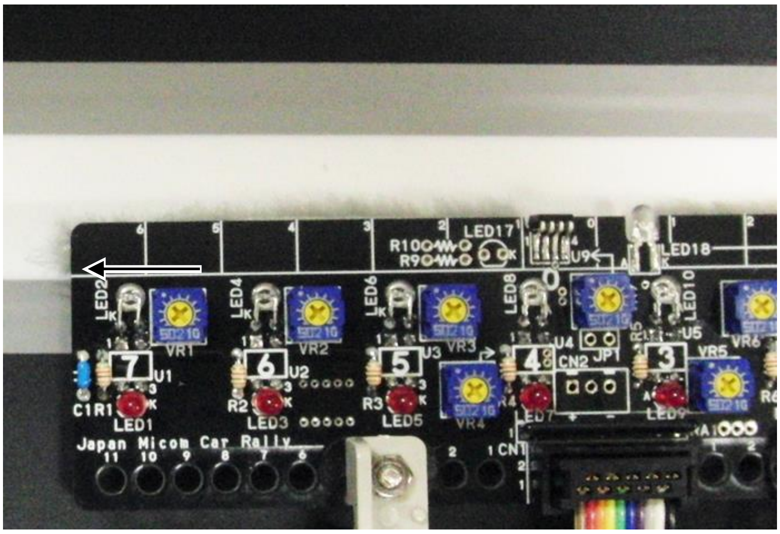
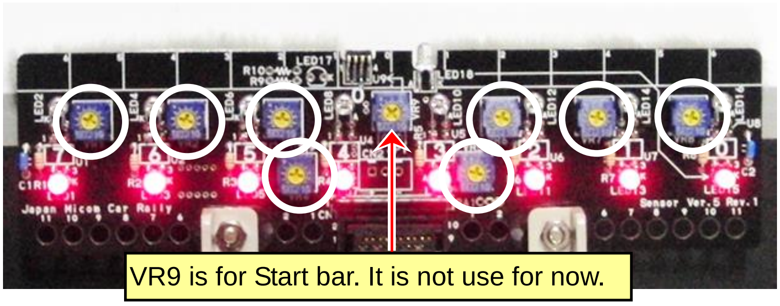
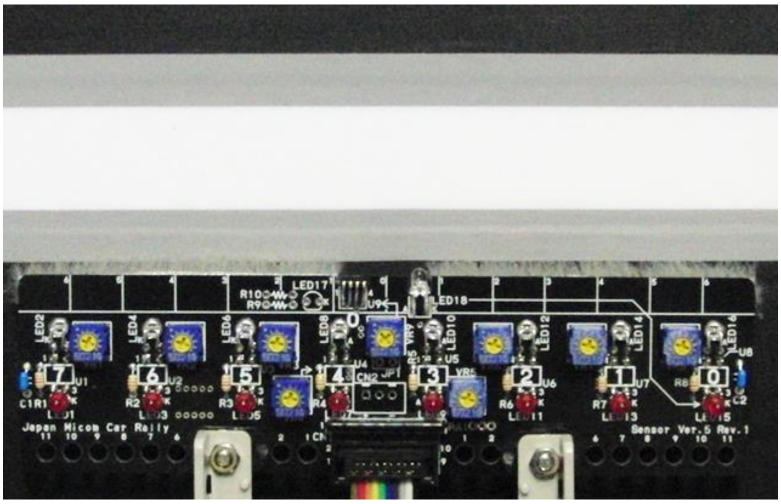
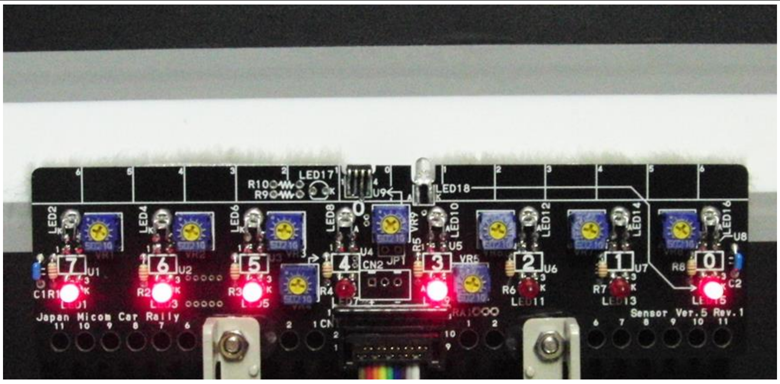
DIP switches	Sensor number output to LED2 on motor drive board	Sensor number output to LED3 on motor drive board
 "1" "0" 1 0 0 0	1	0 It lights even if there is the reaction of the start bar detection sensor
 "1" "0" 1 0 0 1	3	2
 "1" "0" 1 0 1 0	5	4
 "1" "0" 1 0 1 1	7	6

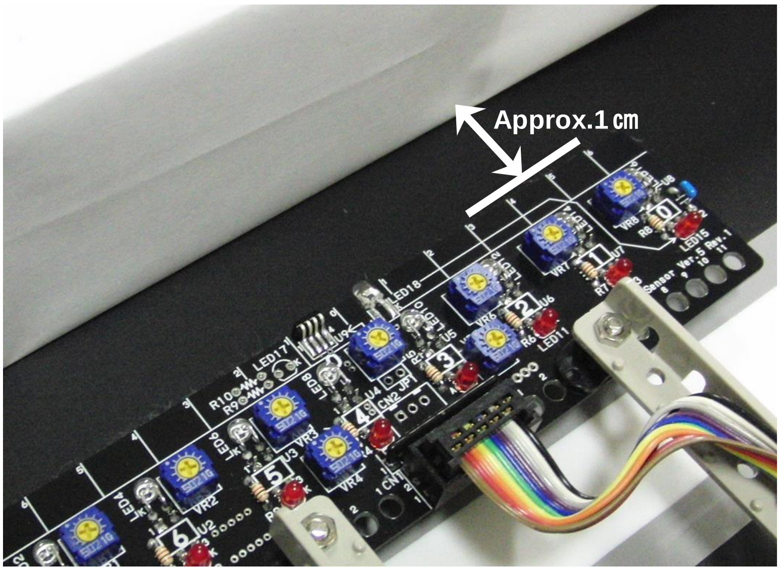
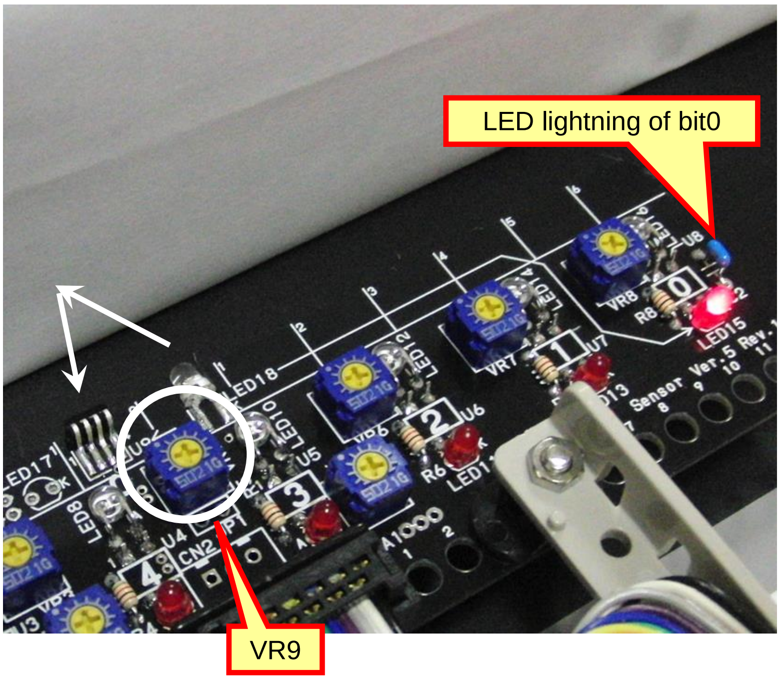
In like manner, you can check the operation of all eight sensors on the sensor board, 7 to 0, by switching the settings of the DIP switches as shown in the table at left.

※Sensor 0 and the start bar detection sensor should both cause the rightmost red LED to light.

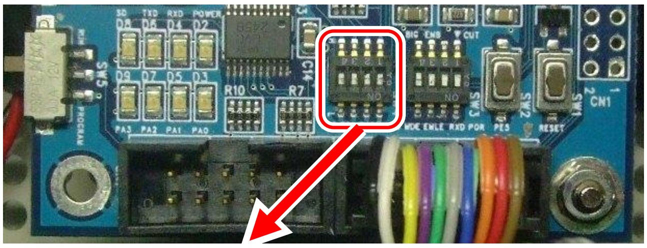
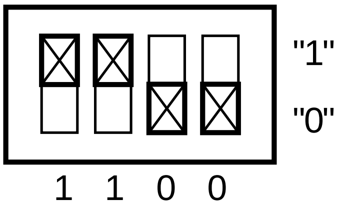
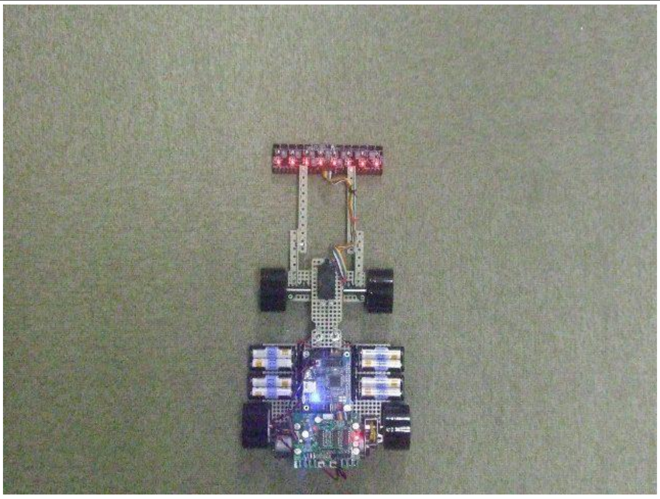
■Adjustment method of the sensor

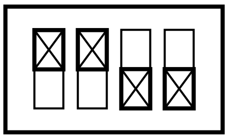
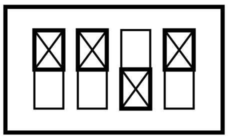
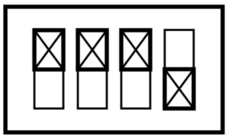
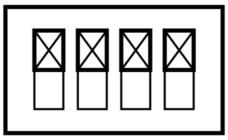
1	 ◆Lateral view 	Place the sensor board parallel to the grey line of the centre of the track as shown in the photo. Place the MCU car body then on the same level as the track and make it the same as a running state.
2	 Note: View from side 	Thus, we cannot adjust it properly even if we try to adjust the sensor while holding it in your hand because a sensor and the interval with the track do not become constant. **The MCU car should be put on the same level as the track.**
3		Turn all nine volume all the way counter clockwise.

4		Set the horizontal line of the board on the line which is switching of white and grey of the track. **Look at it from straight above and set it.**
5		Turn eight volumes clockwise and turn on LED. **Turn it slowly one by one, and stop turning at the moment when LED was turned on.** Adjust it to react to grey by this adjustment. Adjust the MCU car kit to react with white and grey.
6		Lower the sensors a little. All the lights disappear.
7		Bring a sensor close to the grey part **parallel slowly** again. Increase the sensitivity of the LED which is not turned on (clockwise direction). When it is turned on earlier than other LED, decrease the sensitivity (counter clockwise direction). **Adjust it many times so that all LEDs turn on at almost the same time.**

8		Next, adjust a sensor detecting a start bar. **Stand a piece of white board or paper about 1cm away** from the edge of the sensor board. This acts as a substitute for the start bar. **At this time confirm that under the rightmost course detection sensor is a black surface and that LED 15 is not lit.**
9		Turn VR9 of 〇clockwise slowly and adjust it so that **Because LED15 is the most right course detection sensor and combined use, please adjust it so that right under this sensor is black.** Check that the LED turns off as the white board or paper is moved away from the sensor board. The adjustment is complete if the LED disappears.

5.8. Straight Forward Test

1	 	This test checks whether the MCU car can run straight. Place the MCU car on a long flat surface, such as the floor of a hallway. (The MCU car is not placed on the track for this test.) Set the DIP switches to 1100.
2		Move the two power switches of the MCU car to the off position. Two seconds after power-on, the MCU car advances straight forward at a PWM value of 50% for two seconds. By placing the MCU car on a long flat surface, such as the floor of a hallway, this test allows you to confirm that that the car can run straight. If it turns to the right or left, adjust the SERVO_CENTER value as described in **5.4, Servo Operation Test,** so that the car runs straight. The ability to run in a straight line is extremely important when the MCU car is operating at high speed, so make sure to perform this test and adjustment. In the end, you should fine adjust the SERVO_CENTER value down to increments of 1.

<table>
<tr>
<td>3</td>
<td>
<table>
<tr><th>DIP switches</th><th>PWM value</th><th>Duration until stop</th></tr>
<tr><td>
"1"
"0"
1 1 0 0</td><td>50%</td><td>2 seconds</td></tr>
<tr><td>
"1"
"0"
1 1 0 1</td><td>50%</td><td>5 seconds</td></tr>
<tr><td>
"1"
"0"
1 1 1 0</td><td>100%</td><td>2 seconds</td></tr>
<tr><td>
"1"
"0"
1 1 1 1</td><td>100%</td><td>5 seconds</td></tr>
</table>
</td>
<td>The straight forward test has four patterns covering different PWM values and running durations until stop. A larger PWM value, or a longer duration until stop, causes the MCU car to run a longer distance. Choose the test pattern that best matches the amount of space available, select a long, flat test location such as a hallway, and run the straight forward test to confirm the MCU car can run straight.

Note:
The instructions in section 7, Adjusting the Servo centre and Maximum Turn Angle, in MCU Car Kit, Ver. 5.1, Program Explanation Manual-t12_rx62t Version (Version for RX62T) are useful when adjusting the servo centre value.</td>
</tr>
</table>

5.9. Completion of Operation Tests

Once all the functions of the MCU car are operating correctly, write the running program to the MCU and try it out on the track. Before doing that, however, you’ll want to make final adjustments.

- Follow the instructions in section 7, Adjusting the Servo Centre and Maximum Turn Angle, in *MCU Car Kit, Ver. 5.1, Program Explanation Manual—kit12_rx62t Version (Version for X62T)* to adjust the servo centre value.
- Refer to *MCU Car Kit, Ver. 5.1, Program Explanation Manual—kit12_rx62t Version (Version for RX62T)* and write the project kit12_rx62t program in workspace kit12_rx62t to the RMC-RX62T board. Then try running the MCU car on the track.

6. Program Source Code

6.1. Program Code Listing of kit12test_rx62t.c

```
/****************************************************************************/
/*  Supported Microcontroller:RX62T                                         */
/*  File:                   kit12test_rx62t.c                               */
/*  File Contents:          Operation Test Program(RX62T version)           */
/*  Version number:         Ver.1.00                                        */
/*  Date:                   2013.09.05                                      */
/*  Copyright:              Renesas Micom Car Rally Secretariat             */
/****************************************************************************/
/*
This program supports the following boards:
* RMC-RX62T board
* Sensor board Ver. 5
* Motor drive board Ver. 5
*/

/*
operation check MCU car kit sensor board Ver.5 and Motor drive board
Ver.5

change content of operation check by dip switch of MCU board
   DipSW
bit3 2 1 0
   0 0 0 0 check LED           LED on alternately intervals of 0.5 seconds
   0 0 0 1 check push switch   OFF：LED0 ON  ON：LED1ON
   0 0 1 0 check servo         repeat 0° →right30° →left30°
   0 0 1 1 no operation
   0 1 0 0 check right motor   repeat forward  → brake
   0 1 0 1                     repeat backward → brake
   0 1 1 0 check left motor    repeat forward  → brake
   0 1 1 1                     repeat backward → brake

   1 0 0 0 check sensor        output sensor bit1,0 to LED1,0
   1 0 0 1                     output sensor bit3,2 to LED1,0
   1 0 1 0                     output sensor bit5,4 to LED1,0
   1 0 1 1                     output sensor bit7,6 to LED1,0

   1 1 0 0 check straightness  PWM forward at 50%  stop agter 2 seconds
   1 1 0 1 check straightness  PWM forward at 50%  stop after 5 seconds
   1 1 1 0 check straightness  PWM forward at 100% stop after 2 seconds
   1 1 1 1 check straightness  PW  forward at 100% stop after 5 seconds
*/

/*======================================*/
/* Include                              */
/*======================================*/
#include "iodefine.h"

/*======================================*/
/* Symbol definitions                   */
/*======================================*/

/* Constant settings */
#define PWM_CYCLE       24575           /* Motor PWM period (16ms)      */
#define SERVO_CENTER    2300            /* Servo center value           */
#define HANDLE_STEP     13              /* 1 degree value               */

/*======================================*/
/* Prototype declarations               */
/*======================================*/
void init(void);
unsigned char sensor_inp( unsigned char mask );
unsigned char dipsw_get( void );
unsigned char buttonsw_get( void );
unsigned char pushsw_get( void );
void led_out_m( unsigned char led );
void led_out( unsigned char led );
void motor( int accele_l, int accele_r );
void handle( int angle );

/*======================================*/
/* Global variable declarations         */
/*======================================*/
unsigned long   cnt0;
```

```
unsigned long   cnt1;
int             pattern;

/************************************************************************/
/* Main program                                                         */
/************************************************************************/
void main(void)
{
    unsigned char   now_sw;             /* memorize dip switch now      */
    unsigned char   before_sw;          /* memorize dip switch last time*/
    unsigned char   c;                  /* for work                     */

    /* Initialize MCU functions */
    init();

    /* Initialize micom car state */
    handle( 0 );
    motor( 0, 0 );
    led_out( 0x0 );

    /* variable initialization */
    before_sw = dipsw_get();
    cnt1 = 0;

    while( 1 ) {
    /* read dip switch */
    now_sw = dipsw_get();

    /* comparing with switch at last time */
    if( before_sw != now_sw ) {
        /* mismatch →  update value at last time, clear timer value */
        before_sw = now_sw;
        cnt1 = 0;
    }

    /* choose operation check mode by value of dip switch */
    switch( now_sw ) {

        /* check LED LED on alternately intervals of 0.5 seconds */
        case 0:
            if( cnt1 < 500 ) {
                led_out( 0x1 );
            } else if( cnt1 < 1000 ) {
                led_out( 0x2 );
            } else {
                cnt1 = 0;
            }
            break;

        /* check push switch   OFF：LED0 ON  ON：LED1ON */
        case 1:
            led_out( pushsw_get() + 1 );
            break;

        /* check servo   repeat 0° →right30° →left30°  */
        case 2:
            if( cnt1 < 1000 ) {
                handle( 0 );
            } else if( cnt1 < 2000 ) {
                handle( 30 );
            } else if( cnt1 < 3000 ) {
                handle( -30 );
            } else {
                cnt1 = 0;
            }
            break;

        /* not do anything */
        case 3:
            break;

        /* check right motor   repeat forward  → brake */
        case 4:
            if( cnt1 < 1000 ) {
                motor( 0, 100 );
            } else if( cnt1 < 2000 ) {
                motor( 0, 0 );
            } else {
                cnt1 = 0;
            }
```

```
            break;

        /* check right motor   repeat backward  → brake */
        case 5:
            if( cnt1 < 1000 ) {
                motor( 0, -100 );
            } else if( cnt1 < 2000 ) {
                motor( 0, 0 );
            } else {
                cnt1 = 0;
            }
            break;

        /* check left motor    repeat forward  → brake */
        case 6:
            if( cnt1 < 1000 ) {
                motor( 100, 0 );
            } else if( cnt1 < 2000 ) {
                motor( 0, 0 );
            } else {
                cnt1 = 0;
            }
            break;

        /* check left motor    repeat backward  → brake */
        case 7:
            if( cnt1 < 1000 ) {
                motor( -100, 0 );
            } else if( cnt1 < 2000 ) {
                motor( 0, 0 );
            } else {
                cnt1 = 0;
            }
            break;

        /* check sensor   output sensor bit1,0 to LED1,0 */
        case 8:
            c = sensor_inp( 0x03 );
            led_out( c );
            break;

        /* check sensor   output sensor bit3,2 to LED1,0 */
        case 9:
            c = sensor_inp( 0x0c );
            c = c >> 2;
            led_out( c );
            break;

        /* check sensor   output sensor bit5,4 to LED1,0 */
        case 10:
            c = sensor_inp( 0x30 );
            c = c >> 4;
            led_out( c );
            break;

        /* check sensor   output sensor bit7,6 to LED1,0 */
        case 11:
            c = sensor_inp( 0xc0 );
            c = c >> 6;
            led_out( c );
            break;

        /* check straightness  PWM  forward at 50%  stop agter 2 seconds */
        case 12:
            if( cnt1 < 2000 ) {
                motor( 0, 0 );
            } else if( cnt1 < 4000 ) {
                motor( 50, 50 );
            } else {
                motor( 0, 0 );
            }
            break;

        /* check straightness  PWM  forward at 50%  stop agter 5 seconds */
        case 13:
            if( cnt1 < 2000 ) {
                motor( 0, 0 );
            } else if( cnt1 < 7000 ) {
                motor( 50, 50 );
            } else {
```

```
                motor( 0, 0 );
            }
            break;

        /* check straightness  PWM  forward at 100%  stop agter 2 seconds */
        case 14:
            if( cnt1 < 2000 ) {
                motor( 0, 0 );
            } else if( cnt1 < 4000 ) {
                motor( 100, 100 );
            } else {
                motor( 0, 0 );
            }
            break;

        /* check straightness  PWM  forward at 100%  stop agter 5 seconds */
        case 15:
            if( cnt1 < 2000 ) {
                motor( 0, 0 );
            } else if( cnt1 < 7000 ) {
                motor( 100, 100 );
            } else {
                motor( 0, 0 );
            }
            break;

        /* if none */
        default:
            break;
    }
    }
}

/************************************************************************/
/* RX62T Initialization                                                 */
/************************************************************************/
void init(void)
{
    // System Clock
    SYSTEM.SCKCR.BIT.ICK = 0;               //12.288*8=98.304MHz
    SYSTEM.SCKCR.BIT.PCK = 1;               //12.288*4=49.152MHz

    // Port I/O Settings
    PORT1.DDR.BYTE = 0x03;                  //P10:LED2 in motor drive board

    PORT2.DR.BYTE  = 0x08;
    PORT2.DDR.BYTE = 0x1b;                  //P24:SDCARD_CLK(o)
                                            //P23:SDCARD_DI(o)
                                            //P22:SDCARD_DO(i)
                                            //CN:P21-P20
    PORT3.DR.BYTE  = 0x01;
    PORT3.DDR.BYTE = 0x0f;                  //CN:P33-P31
                                            //P30:SDCARD_CS(o)
    //PORT4:input                           //sensor input
    //PORT5:input
    //PORT6:input

    PORT7.DDR.BYTE = 0x7e;                  //P76:LED3 in motor drive board
                                            //P75:forward reverse signal(right motor)
                                            //P74:forward reverse signal(left motor)
                                            //P73:PWM(right motor)
                                            //P72:PWM(left motor)
                                            //P71:PWM(servo motor)
                                            //P70:Push-button in motor drive board
    PORT8.DDR.BYTE = 0x07;                  //CN:P82-P80
    PORT9.DDR.BYTE = 0x7f;                  //CN:P96-P90
    PORTA.DR.BYTE  = 0x0f;                  //CN:PA5-PA4
                                            //PA3:LED3(o)
                                            //PA2:LED2(o)
                                            //PA1:LED1(o)
                                            //PA0:LED0(o)
    PORTA.DDR.BYTE = 0x3f;                  //CN:PA5-PA0
    PORTB.DDR.BYTE = 0xff;                  //CN:PB7-PB0
    PORTD.DDR.BYTE = 0x0f;                  //PD7:TRST#(i)
                                            //PD5:TDI(i)
                                            //PD4:TCK(i)
                                            //PD3:TDO(o)
                                            //CN:PD2-PD0
    PORTE.DDR.BYTE = 0x1b;                  //PE5:SW(i)
                                            //CN:PE4-PE0
```

```

    // Compare match timer
    MSTP_CMT0 = 0;                          //CMT Release module stop state
    MSTP_CMT2 = 0;                          //CMT Release module stop state

    ICU.IPR[0x04].BYTE  = 0x0f;             //CMT0_CMI0 Priority of interrupts
    ICU.IER[0x03].BIT.IEN4 = 1;             //CMT0_CMI0 Permission for interrupt
    CMT.CMSTR0.WORD     = 0x0000;           //CMT0,CMT1 Stop counting
    CMT0.CMCR.WORD      = 0x00C3;           //PCLK/512
    CMT0.CMCNT          = 0;
    CMT0.CMCOR          = 96;               //1ms/(1/(49.152MHz/512))
    CMT.CMSTR0.WORD     = 0x0003;           //CMT0,CMT1 Start counting

    // MTU3_3 MTU3_4 PWM mode synchronized by RESET
    MSTP_MTU            = 0;                //Release module stop state
    MTU.TSTRA.BYTE      = 0x00;             //MTU Stop counting

    MTU3.TCR.BYTE   = 0x23;                 //ILCK/64(651.04ns)
    MTU3.TCNT = MTU4.TCNT = 0;              //MTU3,MTU4TCNT clear
    MTU3.TGRA = MTU3.TGRC = PWM_CYCLE;      //cycle(16ms)
    MTU3.TGRB = MTU3.TGRD = SERVO_CENTER;   //PWM(servo motor)
    MTU4.TGRA = MTU4.TGRC = 0;              //PWM(left motor)
    MTU4.TGRB = MTU4.TGRD = 0;              //PWM(right motor)
    MTU.TOCR1A.BYTE = 0x40;                 //Selection of output level
    MTU3.TMDR1.BYTE = 0x38;                 //TGRC,TGRD buffer function
                                            //PWM mode synchronized by RESET
    MTU4.TMDR1.BYTE = 0x00;                 //Set 0 to exclude MTU3 effects
    MTU.TOERA.BYTE  = 0xc7;                 //MTU3TGRB,MTU4TGRA,MTU4TGRB permission for output

    MTU.TSTRA.BYTE  = 0x40;                 //MTU0,MTU3 count function
}

/************************************************************************/
/* Interrupt                                                            */
/************************************************************************/
#pragma interrupt Excep_CMT0_CMI0(vect=28)
void Excep_CMT0_CMI0(void)
{
    cnt0++;
    cnt1++;
}

/************************************************************************/
/* Sensor state detection                                               */
/* Arguments:       masked values                                       */
/* Return values:   sensor value                                        */
/************************************************************************/
unsigned char sensor_inp( unsigned char mask )
{
    unsigned char sensor;

    sensor  = ~PORT4.PORT.BYTE;

    sensor &= mask;

    return sensor;
}

/************************************************************************/
/* DIP switch value read                                                */
/* Return values: Switch value, 0 to 15                                 */
/************************************************************************/
unsigned char dipsw_get( void )
{
    unsigned char sw,d0,d1,d2,d3;

    d0 = ( PORT6.PORT.BIT.B3 & 0x01 );  /* P63~P60 read                 */
    d1 = ( PORT6.PORT.BIT.B2 & 0x01 ) << 1;
    d2 = ( PORT6.PORT.BIT.B1 & 0x01 ) << 2;
    d3 = ( PORT6.PORT.BIT.B0 & 0x01 ) << 3;
    sw = d0 | d1 | d2 | d3;

    return  sw;
}

/************************************************************************/
/* Push-button in MCU board value read                                  */
/* Return values: Switch value, ON: 1, OFF: 0                           */
/************************************************************************/
unsigned char buttonsw_get( void )
```

```
{
    unsigned char sw;

    sw = ~PORTE.PORT.BIT.B5 & 0x01;     /* Read ports with switches    */

    return  sw;
}

/************************************************************************/
/* Push-button in motor drive board value read                          */
/* Return values: Switch value, ON: 1, OFF: 0                           */
/************************************************************************/
unsigned char pushsw_get( void )
{
    unsigned char sw;

    sw  = ~PORT7.PORT.BIT.B0 & 0x01;    /* Read ports with switches    */

    return  sw;
}

/************************************************************************/
/* LED control in MCU board                                             */
/* Arguments: Switch value, LED0: bit 0, LED1: bit 1. 0: dark, 1: lit  */
/*                                                                      */
/************************************************************************/
void led_out_m( unsigned char led )
{
    led = ~led;
    PORTA.DR.BYTE = led & 0x0f;
}

/************************************************************************/
/* LED control in motor drive board                                     */
/* Arguments: Switch value, LED0: bit 0, LED1: bit 1. 0: dark, 1: lit  */
/* Example: 0x3 -> LED1: ON, LED0: ON, 0x2 -> LED1: ON, LED0: OFF      */
/************************************************************************/
void led_out( unsigned char led )
{
    led = ~led;
    PORT7.DR.BIT.B6 = led & 0x01;
    PORT1.DR.BIT.B0 = ( led >> 1 ) & 0x01;
}

/************************************************************************/
/* Motor speed control                                                  */
/* Arguments:   Left motor: -100 to 100, Right motor: -100 to 100      */
/*        Here, 0 is stopped, 100 is forward, and -100 is reverse.     */
/* Return value:    None                                                */
/************************************************************************/
void motor( int accele_l, int accele_r )
{
    int    sw_data;

    sw_data = dipsw_get() + 5;
    accele_l = accele_l * sw_data / 20;
    accele_r = accele_r * sw_data / 20;

    /* Left Motor Control */
    if( accele_l >= 0 ) {
        PORT7.DR.BYTE &= 0xef;
        MTU4.TGRC = (long)( PWM_CYCLE - 1 ) * accele_l / 100;
    } else {
        PORT7.DR.BYTE |= 0x10;
        MTU4.TGRC = (long)( PWM_CYCLE - 1 ) * ( -accele_l ) / 100;
    }

    /* Right Motor Control */
    if( accele_r >= 0 ) {
        PORT7.DR.BYTE &= 0xdf;
        MTU4.TGRD = (long)( PWM_CYCLE - 1 ) * accele_r / 100;
    } else {
        PORT7.DR.BYTE |= 0x20;
        MTU4.TGRD = (long)( PWM_CYCLE - 1 ) * ( -accele_r ) / 100;
    }
}

/************************************************************************/
/* Servo steering operation                                             */
/* Arguments:   servo operation angle: -90 to 90                        */
```

```
/*              -90: 90-degree turn to left, 0: straight,              */
/*               90: 90-degree turn to right                           */
/***********************************************************************/
void handle( int angle )
{
    /* When the servo move from left to right in reverse, replace "-" with "+". */
    MTU3.TGRD = SERVO_CENTER - angle * HANDLE_STEP;
}

/***********************************************************************/
/* end of file                                                         */
/***********************************************************************/
```