Software Guide for NS-RX231

<INFRARED TRANSMIT>

Contents

1 Load Project

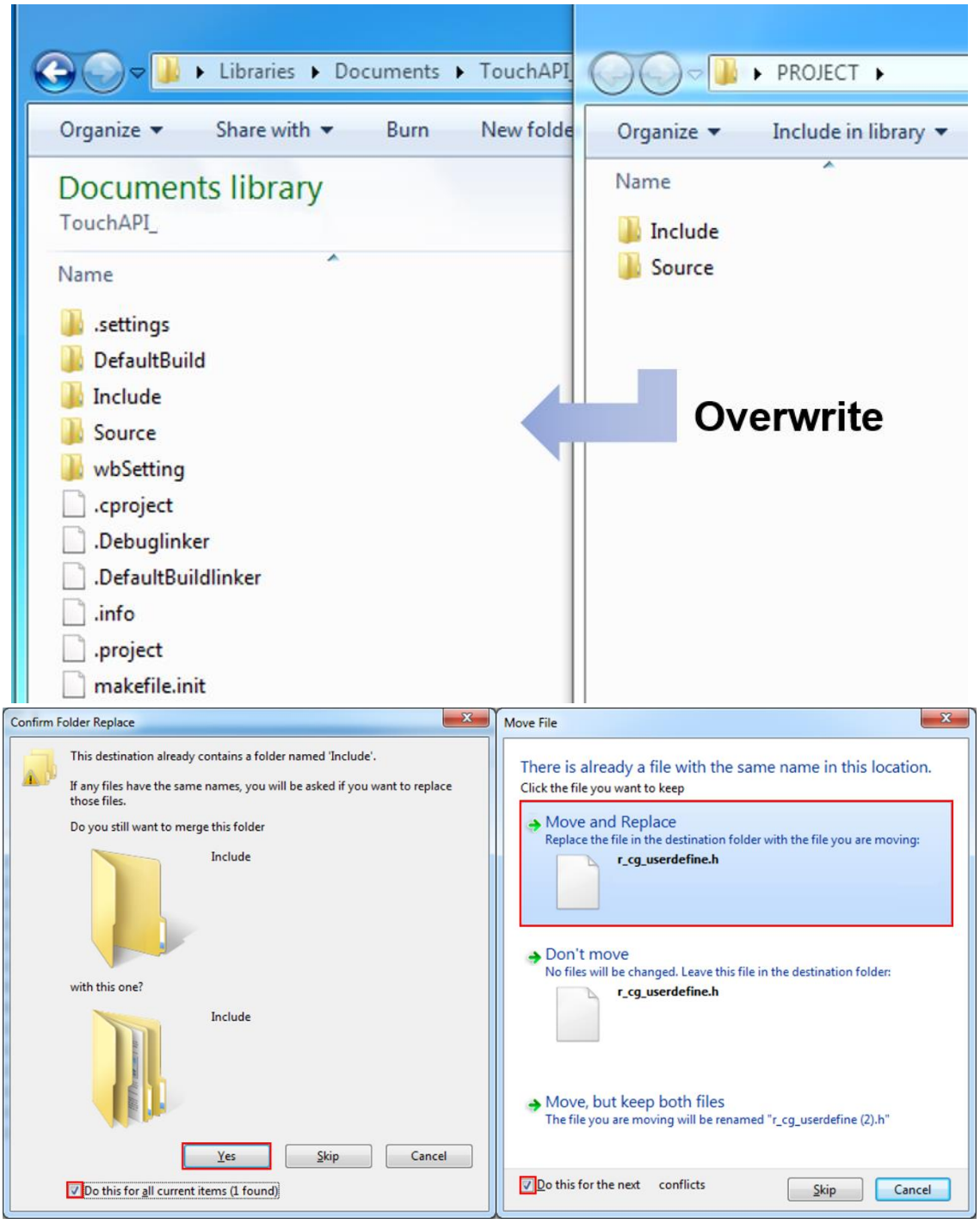

Figure 1-1 Overwrite the source file

Overwrite the attached source file with the project created by the Workbench6 First step guide wizard and run it in e2studio.

2 Summary

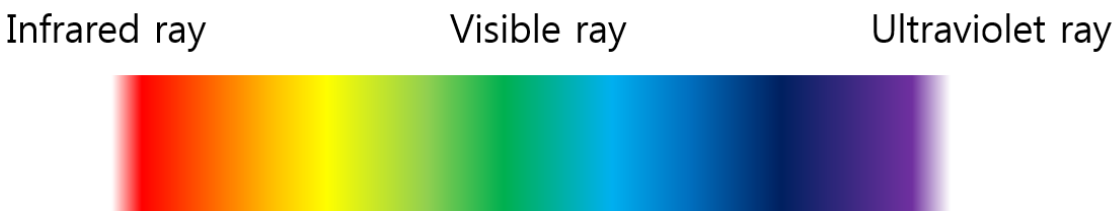

Figure 2-1 Electromagnetic Wave Wavelength

An infrared ray is an electromagnetic wave having a longer wavelength than visible light and is located outside of the wavelength expressing red, so it is called infrared.

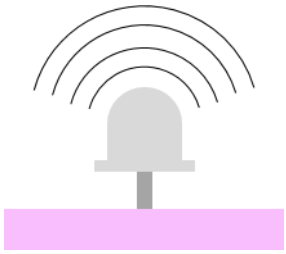

Figure 2-2 InfraRed Emitting Device IR LED

The difference of IR LEDs emit light to current flows like ordinary LEDs is that IR LEDs emit infrared rays which are not visible to the human eye You can use it to implement wireless communication. A good example of using infrared as communication method is an infrared remote controller.

Infrared communication is slow because of its characteristics and is not suitable for high capacity data communication but suitable for small capacity data.

Carrier frequency	30 kHz	TSOP4130
	33 kHz	TSOP4133
	36 kHz	TSOP4136
	38 kHz	TSOP4138
	40 kHz	TSOP4140
	56 kHz	TSOP4156

Figure 2-3 The carrier frequency of the infrared receiver

In case of TSOP4138, which is an IR receiving device attached to NS-RX231, it has the characteristic that it catches the signal of 38kHz, so the timer repeats interruption in every 13.15us and make wave to make 38kHz.

3 Project

3.1 Communication Structure

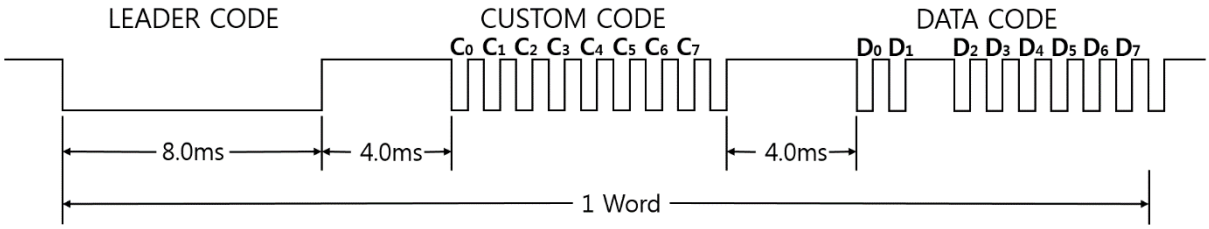

Figure 3-1 Communication Protocol

As an example, here is shown a project created with simple signal format.

The configuration consists of a reader code to indicate the start of communication, a custom code to distinguish the remote controller after that, and a data code to transmit the data.

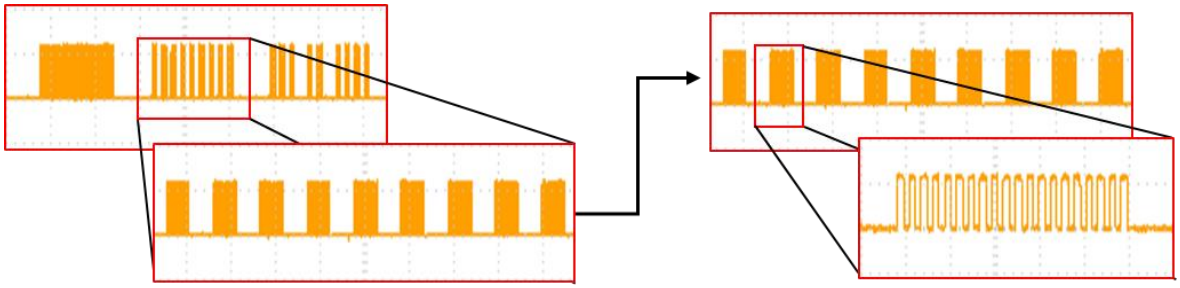

Figure 3-2 Infrared transmission waveform

And the distinction of one bit between 0 and 1 is as follows

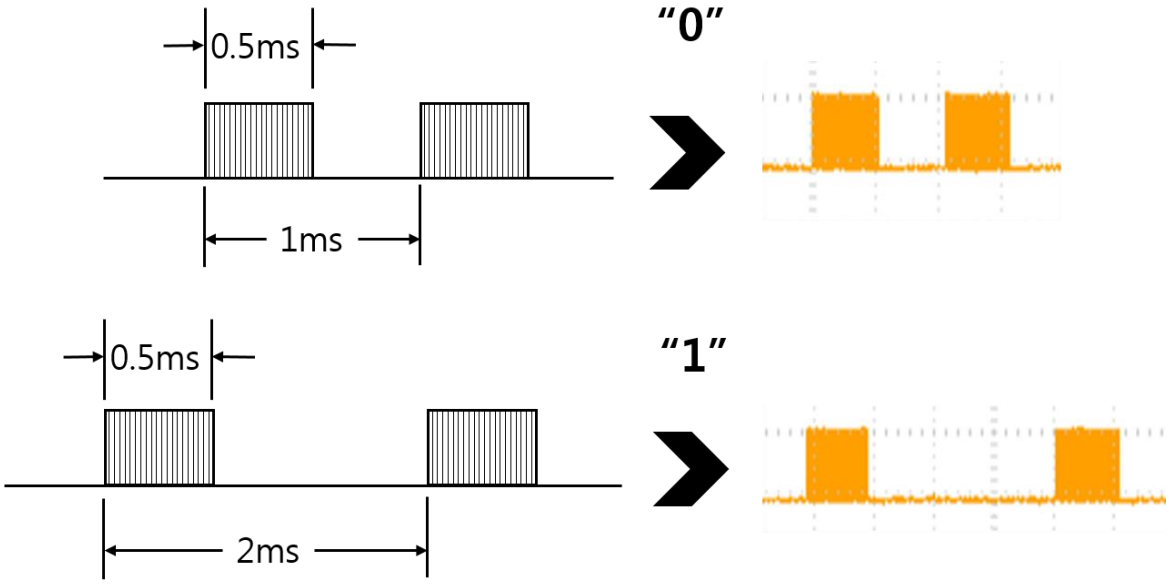

Figure 3-3 Distinction between 0 and 1 in receiving infrared communication

3.2 Source Code

r_main.c

```
if (_0_SUCCESS == R_Set_Cap_Touch_Result_Create( method ))
{
    ts_result = R_Get_Cap_Touch_Result( method );
    if(1 == ts_result.button[1] >> 0)
    {
        PORTB.PODR.BYTE = 0x02;
        sendArr = IR_send(232,40);
    }
    else if(1 == ts_result.button[1] >> 6)
    {
        PORTB.PODR.BYTE = 0x08;
        sendArr = IR_send(232,24);
    }
    else if(1 == ts_result.button[1] >> 7)
    {
        PORTB.PODR.BYTE = 0x20;
        sendArr = IR_send(232,168);
    }
}
```

r_cg_tmr_user.c

```
if(sendArr[37] >= 1){
    sendArr[cnt]--;
    if(flg == 0){
        PORTA.PODR.BIT.B4 ^= 1;
    }

    if(sendArr[cnt] < 1){
        flg ^= 1;
        cnt++;
        if(cnt == 38){
            cnt = 0;
        }
    }
}
```

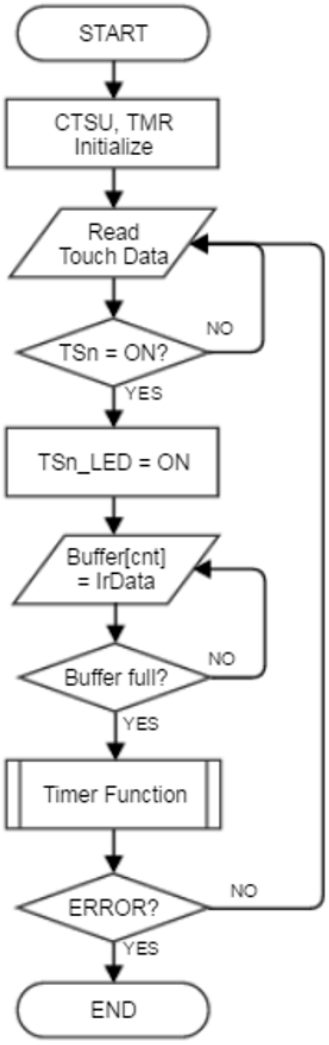

Figure 3-4 Source code and flowchart

The timer is activated in every 13us to process the conditional expression. (One cycle of 38kHz is 26.3us)
It works when the last element of the sendArr array is 1 or more. If flg is 0, it generates a waveform. If it waits for the time specified by the elements of the array without creating a waveform

An example of operation is as follows. In order to create a waveform of 8 ms, referring to the 0th element of the sendArr array, it is set to 616 and it is specified that the interrupt processing is repeated 616 times (13 us * 616 = 8008 us). Since the variable flag is 0, inverting the output to the port for each timer interrupt and subtract the 0th element of the array one by one. When it repeats 616 times and becomes 0, it inverts the value of flg and adds +1 the value of variable cnt to refer to the next element of the array.

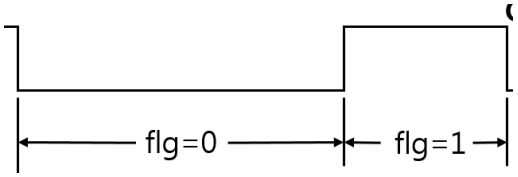

Figure 3-5 Change of flag value

4 Debugging

Figure 4-1 Connect NS-RX231 to power adapter and E1 debugger.

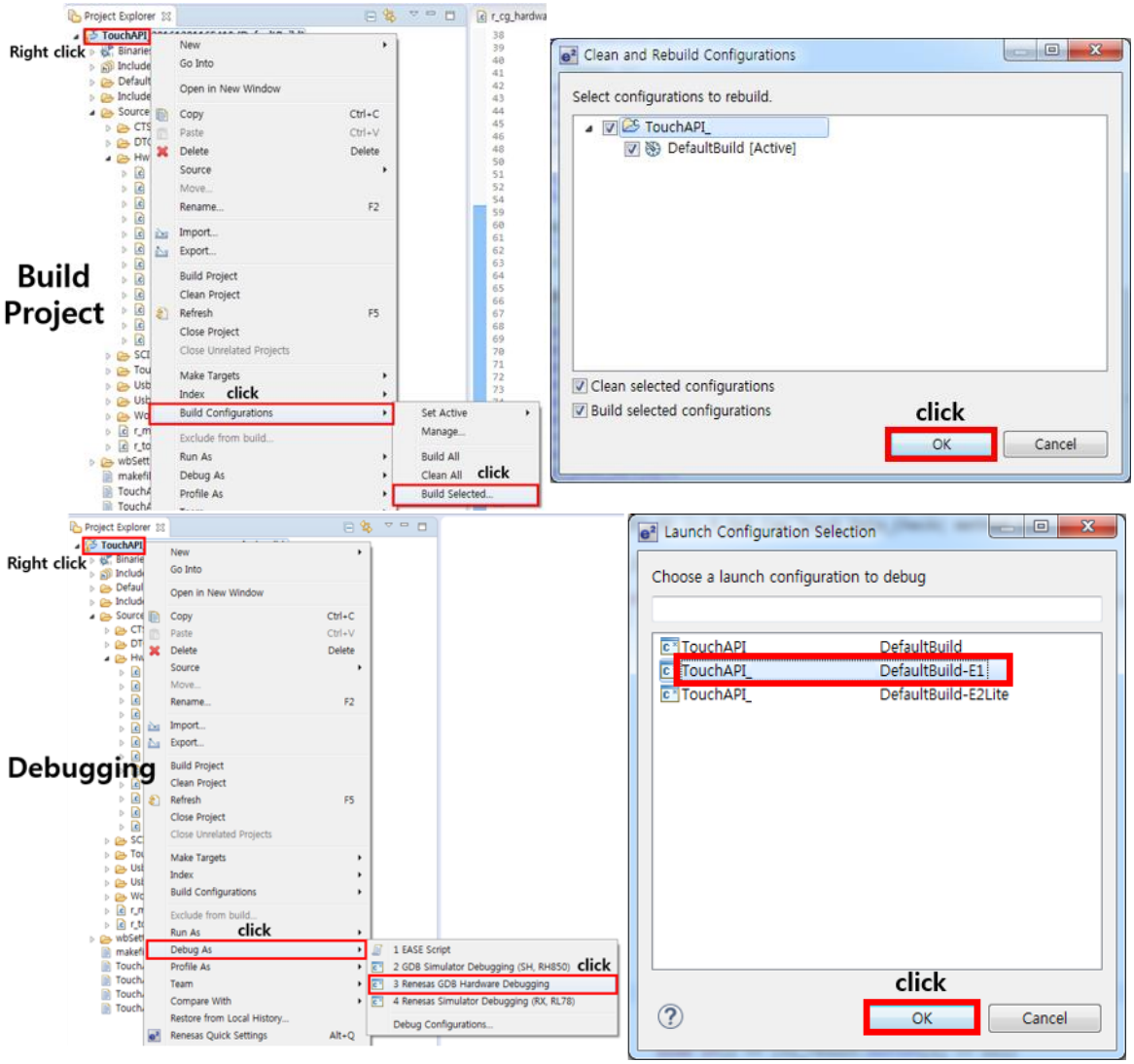

Figure 4-2 Project Build and Debugging

5 Execution

Uploading the above source code to the board, touch TS16.

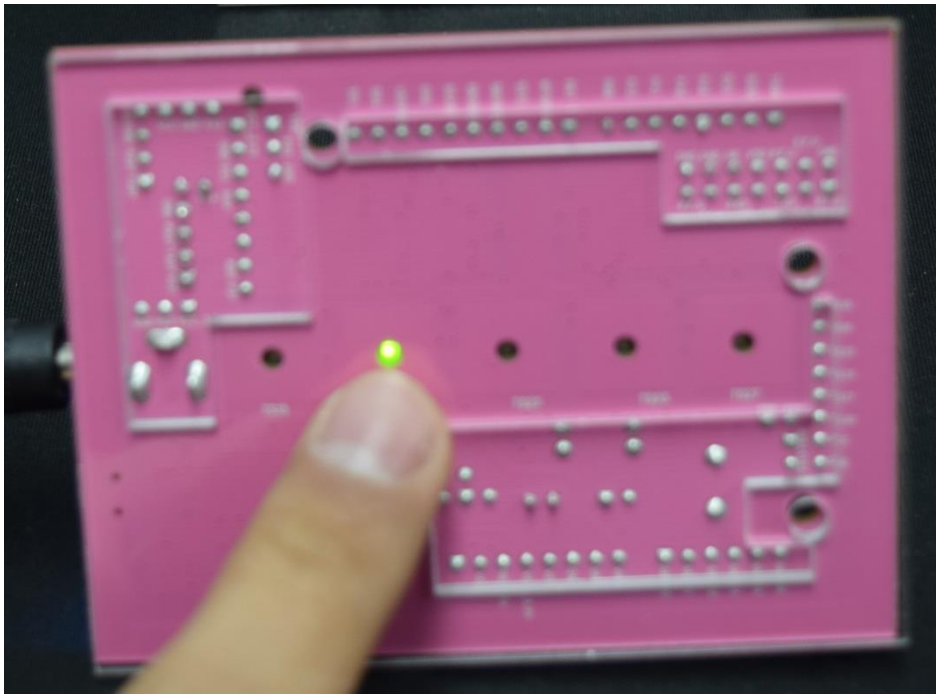

Figure 5-1 Touch the TS16

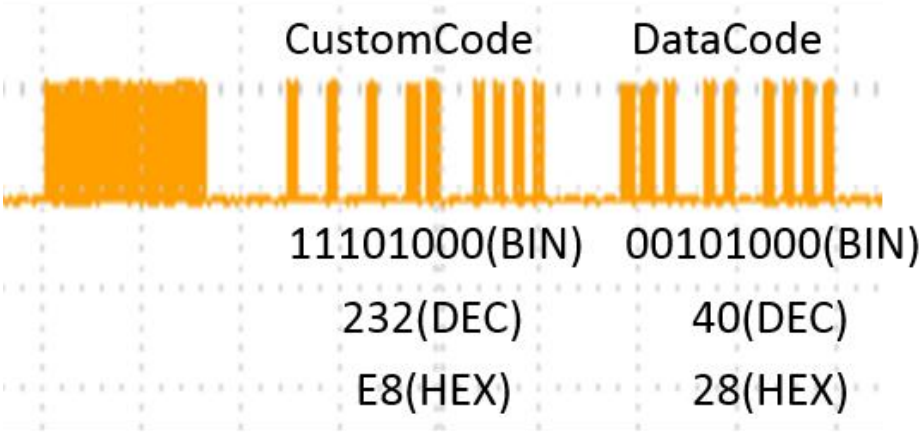

Figure 5-2 Waveform Observed on the Oscilloscope

A waveform in the figure above can be observed on the oscilloscope.

6 Schematic

The figure below is a circuit diagram of the infrared remote control transmitter.

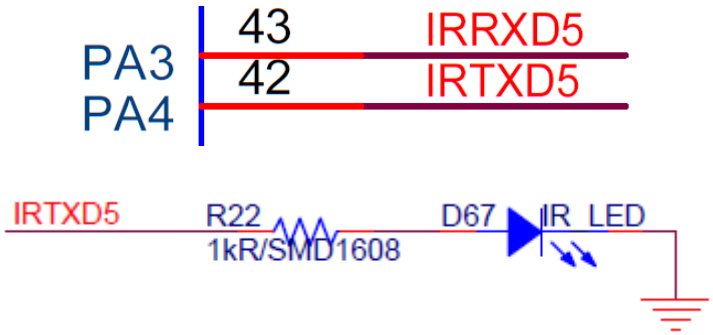

Figure 6-1 Infrared transmitter circuit of NS-RX231

7 Added

To change the length of the reader code, or to send the data code to reduce the error of the data code after sending the inverted data code or to make another protocol you have created, change the IR_send function part in the r_detectIR.c source folder.

ReturnArr [38] is the array to return from the function, and 38 specifies the size of the array.

The contents of the array are as follows.

Reader Code 8ms is returnArr[0] = 616
Reader Code 4ms is returnArr[1] = 308

Custom Code C0 returnArr[2] = 38
Custom Code C0 returnArr[3] = if value as 1 then 116, else then 38

…

Custom code C7 returnArr[17] = if value as 1 then 116, else then 38

Short waveform indicating the end = 38
waiting time 4ms = 308

…

One bit that distinguish between 1 and 0 is a pair of the first argument for outputting short waveform (0.5 ms) and the second argument (0.5 ms or 1.5 ms) that divide 1 and 0. As a total of 8 bits are 1 byte, 16 arguments are required. And as the ending code announcing the end of a byte is also a pair of 2 arguments, a total of 18 array elements is needed to add a byte of data code.

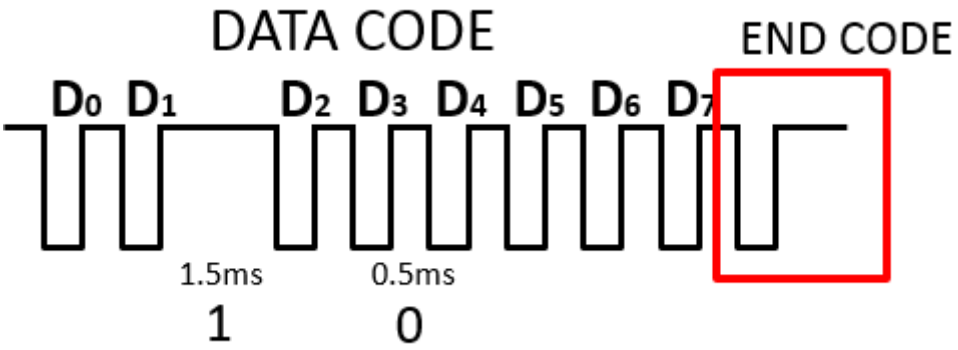

figure 7-1 1-byte data code example

※ Each array element value to be added must be an even number.