Wiegand Interface with N76E003

Example Code Introduction for 8-bit NuMicro® Family

Information

Application	This sample code is based on the N76E003 timer and the GPIO to send and receive Wiegand protocol package.
BSP Version	N76E003_BSP_Keil_C51_V1.0.6
Hardware	NT-N76E003 V2.1

The information described in this document is the exclusive intellectual property of Nuvoton Technology Corporation and shall not be reproduced without permission from Nuvoton.

Nuvoton is providing this document only for reference purposes of NuMicro microcontroller based system design. Nuvoton assumes no responsibility for errors or omissions.

All data and specifications are subject to change without notice.

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

1 Function Description

1.1 Introduction

The Wiegand protocol, also commonly called the Wiegand interface, developed by Motorola, is a wiring standard found on card readers and access control systems, and has become the most common de faction communication method for such systems. The most common format is the standard 26-bit packet. This sample code is based on the N76E003's Timer and GPIO to send and receive 26-bit Wiegand format data.

1.2 Wiegand Interface

The Wiegand interface is a communication protocol consisting of 3 lines of Data0, Data1, Data return. Data0 and Data1 maintain a high level of +5V when there is no data output. If the output is 0, D0 is pulled low for a while; if the output is 1, D1 is pulled low.

1.3 Wiegand Protocol

26-bit Wiegand Format The composition of the open de facto industry standard 26 bit Wiegand format contains 8 bits for the facility code field and 16 bits for the ID number field. Mathematically these 8 facility code bits allow for a total of just 256 (0 to 255) facility codes, while the 16 ID number bits allow for a total of only 65,536 (0 to 65,535) individual ID's within each facility code. Due to the mathematical limitations of the 26-bit Wiegand format, code duplication might occur.

A summary the 26-bit Wiegand format:

Bit Number	Purpose
Bit 1	Even parity over bits 2 to 13
Bit [2:9]	Facility code (0 to 255); Bit 2 is MSB
Bit [10:25]	ID Number (0 to 65,535); Bit 10 is MSB
Bit [26]	Odd parity over bits 14 to 25

The format is represented as following:

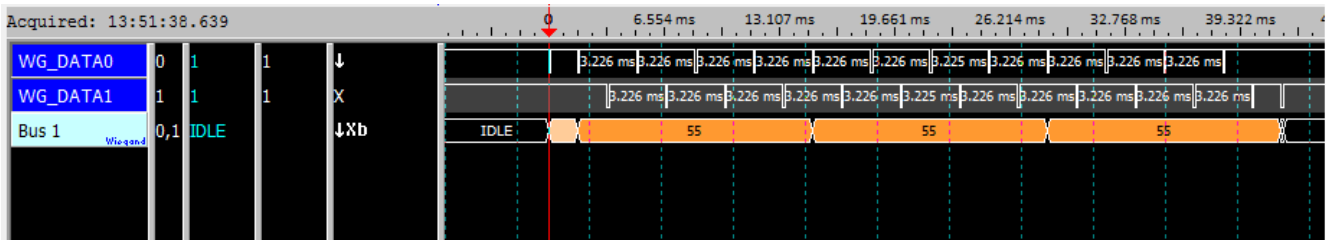
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
P	A	A	A	A	A	A	A	A	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	P
	Even parity Field								Odd Parity Field																
	Facility Code								Card Number																

Public ID code (PID): Each Card with a different Public ID code. The PID is easily found in the output of the reader

Hidden ID code (HID): HID is a very important number that exists not only in the card but also in the card reader. If the HID in the card is different from the HID in the card reader, then the card will not work properly on this card reader.

1.4 Wiegand Demo Result

After send 0x55, 0x55, 0x55 following signal will be show on Weigand Bus.



2 Code Description

The main loop：

```
/*---------------------------------------------------------------------------*/
/* Global variables                                                          */
/*---------------------------------------------------------------------------*/
unsigned char out[3] = {0x55, 0x55, 0x55};

/*---------------------------------------------------------------------------*/
/* Functions                                                                 */
/*---------------------------------------------------------------------------*/
void main(void)
{
    //Keep wiegand bus on high state.
    P05 = 1;
    P06 = 1;
    //initial all gpio on quasi mode
    Set_All_GPIO_Quasi_Mode;
    delay_1500us();
    //send wiegand package
    send_wiegand26(out);
    while(1);
}
```

Send 26 bit data detail ：

```
void send_wiegand26(unsigned char *str)
{
    unsigned char data i;
    unsigned  char check_temp;
    unsigned char data check;
    bit  even;
    bit  odd;
    unsigned char data wiegand[3];

    wiegand[0] = *str;
    wiegand[1] = *(str + 1);
    wiegand[2] = *(str + 2);
    check = 0;
    check_temp = wiegand[0];
    //even data check for data 2-9
    for(i = 0; i < 8; i++)
    {
        if((check_temp & 0x80) == 0X80)
        {
            check++;
        }
        check_temp = check_temp << 1;
    }
    if(check % 2 == 0)
        even = 0;
    else
        even = 1;
    check = 0;
    //odd data check for data 10-25
    check_temp = wiegand[1];
    for(i = 0; i < 8; i++)
    {
```

```
        if(check_temp & 0x80 == 0X80)
        {
            check++;
        }
        check_temp = check_temp << 1;
    }
    check_temp = wiegand[2];
    for(i = 0; i < 8; i++)
    {
        if(check_temp & 0x80 == 0X80)
        {
            check++;
        }
        check_temp = check_temp << 1;
    }
    if(check % 2 == 0)
        odd = 1;
    else
        odd = 0;
    //send header ep
    if(even)
    {
        WG_send_bit_1();
    }
    else
    {
        WG_send_bit_0();
    }
    //send data
    for(i = 0; i < 24; i++)
    {

        if((wiegand[0]) & 0x80)
        {
            WG_send_bit_1();
        }
        else
        {
            WG_send_bit_0();
        }
        (*(long*)&wiegand[0]) <<= 1;
    }
    //send op
    if(odd)
    {
        WG_send_bit_1();
    }
    else
    {
        WG_send_bit_0();
    }
}           …
           GPIO_CLR_INT_FLAG(PE, BIT0);
       }
```

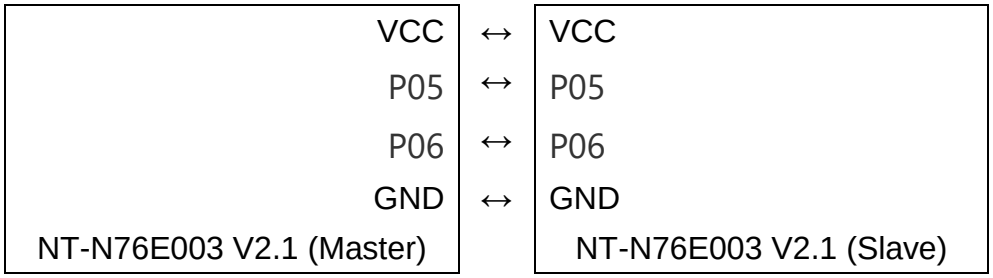
3 Software and Hardware Environment

Software Environment

- BSP Version
  - N76E003_BSP_Keil_C51_V1.0.6
- IDE Version
  - Keil uVersion 4.6 and PK51 Development Kit V9.52

Hardware Environment

- Circuit component
  - NT-N76E003 V2.1 * 2
- Diagram



4 Directory Information

- N76E003 Wiegand_V1.0
  - Common: N76E003 common function library.
  - Include: N76E003 code header.
  - Sample_Code
    - ExampleCode： Source file of example code
      - Wiegand_Master: Example code source file of wiegand master.
      - Wiegand_Slave: Example code source file of wiegand slave.
  - Startup
    - N76E003: startup code.

5 How to Execute Example Code

1. The project can be run in Keil uVision4 PK51 Development Kit V9.52 and above version.
2. Enter the Sample_Code\ExampleCode folder according to the Directory Information. Double-click Wiegand_Master and Wiegand_Slave folder the Wiegand_Master.uvproj and Wiegand_Slave.uvproj project file
3. Enter Keil compiler mode
   a. Build
   b. Download
   c. Enter / Exit Debug Mode
4. When in under Debug mode
   a. Free run

6 Revision History

Date	Revision	Description
Sep. 24, 2019	1.00	1. Initially issued.

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.

Please note that all data and specifications are subject to change without notice.
All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.