User Manual

CAN Boot Loader Design on the Kinetis EA128

Introduction

This user manual is used to introduce how to use the CAN boot loader which is designed on the KEAZ128 for our user.

Contents

1 Introduction of PC software

The PC software is DemoCAN. The user interface of DemoCAN is shown in figure 1.

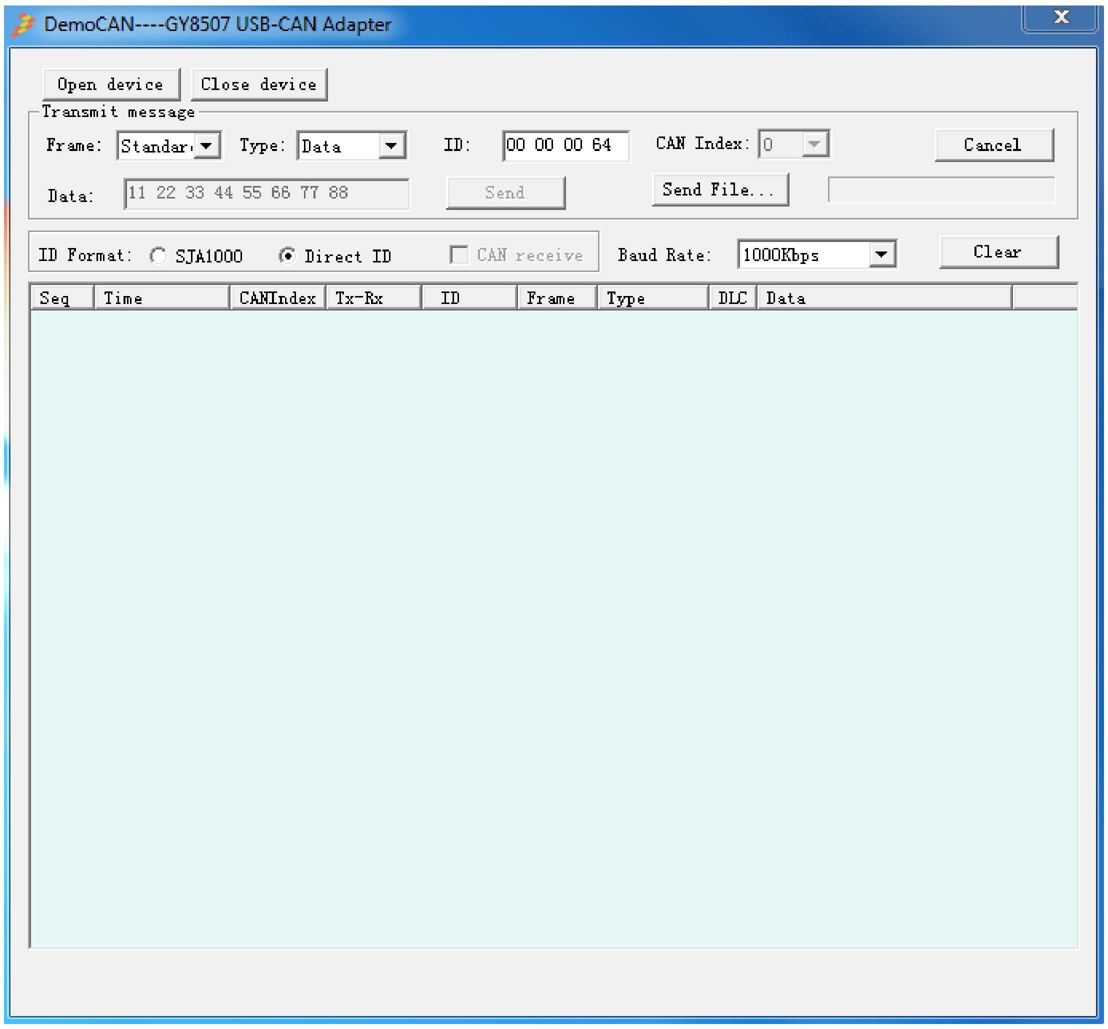


Figure 1. User interface of DemoCAN

2 Introduction of boot loader code

2.1 Software flow chart

After the power is on, the KEA will first run the boot loader to check whether a hook up will be successful. If overtime occurs, jump to the boot vector of application to initialize SP and write SCB_VTOR register with the user interrupt vector address. Then, run the application code. The software flow is shown in figure 2.

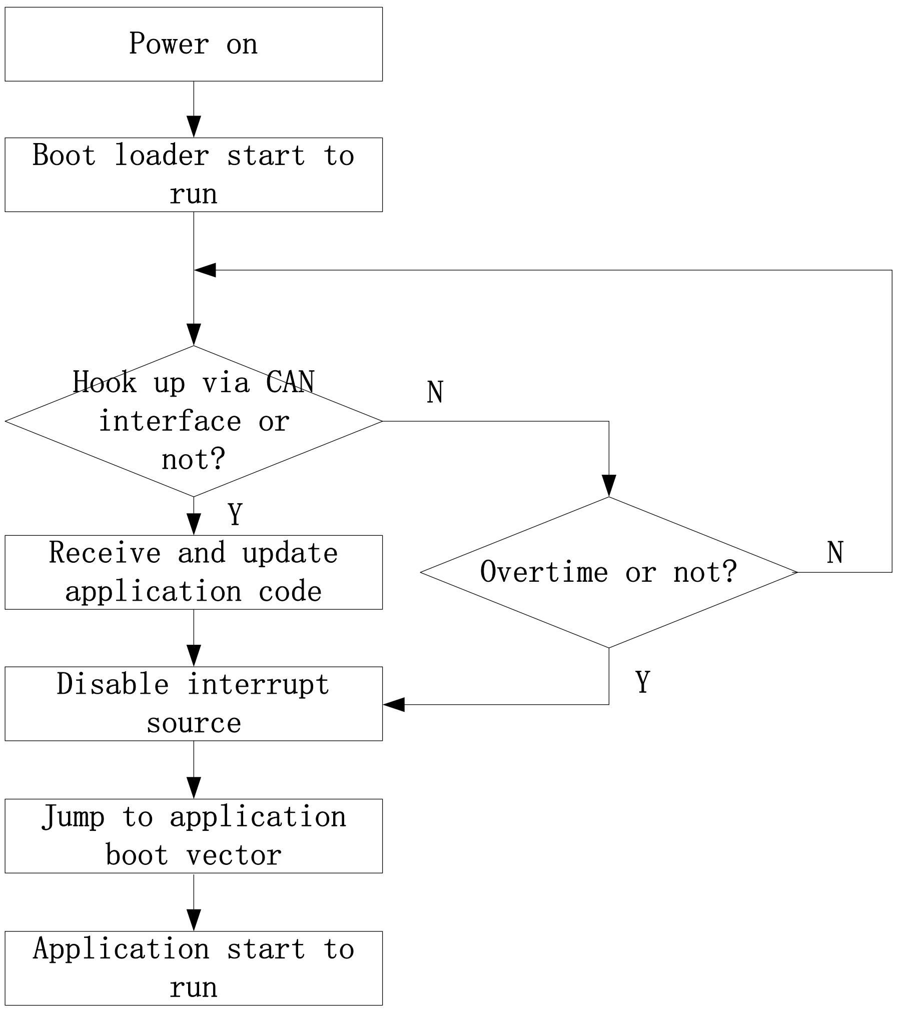


Figure 2. Software flow chart of boot loader

2.2 Boot loader communication (BLC) protocol

Using the DemoCAN.exe open the application s19 file, the PC software will send FF (*DOWN_LINK*) data until the KEAZ128 demo board feedback. The KEAZ128 sends C3 (*UP_READY*) to the PC via a high speed CAN transceiver (33901). Then, the PC send one standard frame of application s19 file after receiving a C3 (*UP_READY*) feedback. After sending the last frame of a line of s19 file, PC sends an FE (*DOWN_LINE_END*) to KEAZ128. The KEAZ128 feedback the C2 (*UP_BUSY*) to ask PC not to send CAN frame and begin to program the one line of s19 file to flash. After that, the KEAZ128 change to send C3 (*UP_READY*) to be ready.

After sending the whole s19 file, PC sends FD (*DOWN_FILE_END*) to KEAZ128 with the feedback C1 (*UP_PRGEND*), and the update is successful. The flow chart of BLC protocol based on the KEA is shown in figure 3.

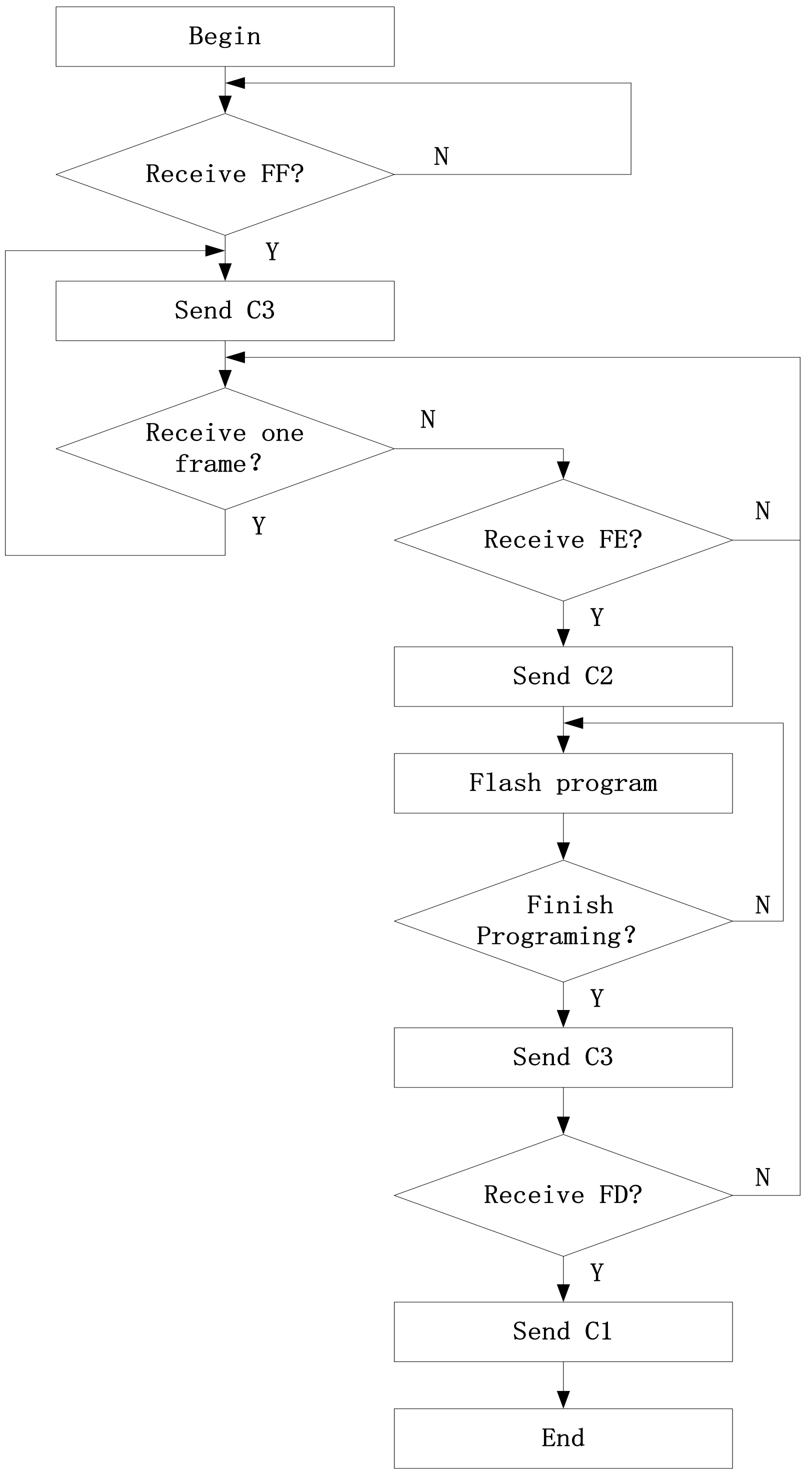


Figure 3. The flow chart of BLC protocol

FF: DOWN_LINK; FD: DOWN_LINE_END; FD: DOWN_FILE_END;

C1: UP_PRGEND; C2: UP_BUSY; C3: UP_READY

2.3 Status indicators

When executing the flash program, the LED2 blinks which means there is no error occur during the operation. If the LED3 is lighting instead of LED2 blink, there are some errors occur during the operation. The status indicators will give user a good interface to the demo.

3 Introduction of application code

For the relocation of vector table of application, we need to modify the application template to adapt to the boot loader. There are three user applications based on the KEAZ128. Two is with driver library and another built base on the default template. The first application is a lighting demo with RTC interrupt, the second one is a flash LED demo with cycle(without interrupt), and the last one is a FTM demo with RTC interrupt base on the default template of Codewarrior 10.6. The user need to create a project, build and get the .hex file from FLASH folder. Then, modify the suffix from .hex to .s19, so that, it can be identified by the PC software.

3.1 Link file

The normal project is available to define the code start address to any acceptable address. The memory is divided as below.

```
MEMORY
{
   m_interrupts     (rx) : ORIGIN = 0x00004000, LENGTH = 0xC0
   m_cfmprotrom     (rx) : ORIGIN = 0x00000400, LENGTH = 0x10
   m_text           (rx) : ORIGIN = 0x000040C0, LENGTH = 128K - 0x40C0
   m_data          (rwx) : ORIGIN = 0x1FFFF000, LENGTH = 16K
}
```

The application code will begin with 0x4000 which is the relocation vector address (RELOCATION_VERTOR_ADDR), so the boot vector should be

RELOCATION_VERTOR_ADDR plus 0x0004.

3.2 Flash configuration region

The flash configuration region is located in 0x400 to 0x40F, which is in the boot loader region. If this section is protected, it cannot be modified by application code. Otherwise, there are errors of flash programming. The application code will first read out all of contents of the sector that contain a flash configuration field at the address 0x400 to 0x40F. Then, erase this sector, modify the buffer, and then write back to this sector.

In our app_1 and app_2 demo, the flash configuration can be modified in the vector.h file as below.

```
#ifdef USE_BOOTLOADER
#else
#define CONFIG_1          0xffffffff
#define CONFIG_2          0xffffffff
#define CONFIG_3          0xffffffff
#define CONFIG_4          0xfffeffff
#endif
```

Because of boot loader protected, we need to define USE_BOOTLOADER in our application code.

```
#define USE_BOOTLOADER
```

Our app_3 demo is based on the default template of Codewarrior 10.6. If building a new MCU project by Codewarrior 10.6 by default like app_3, the flash configuration is default, so USE_BOOTLOADER is not needed to be defined as mentioned above. Only need to cover the default SKEAZ128_flash.ld file of one of the three applications demos.

4 Operation steps

Either using the application code to modify or creating a new project by default, just need to modify the link file to suit the boot loader.

4.1 Debug mode

Step one: connect each device together as shown in the figure 4 and figure 5.

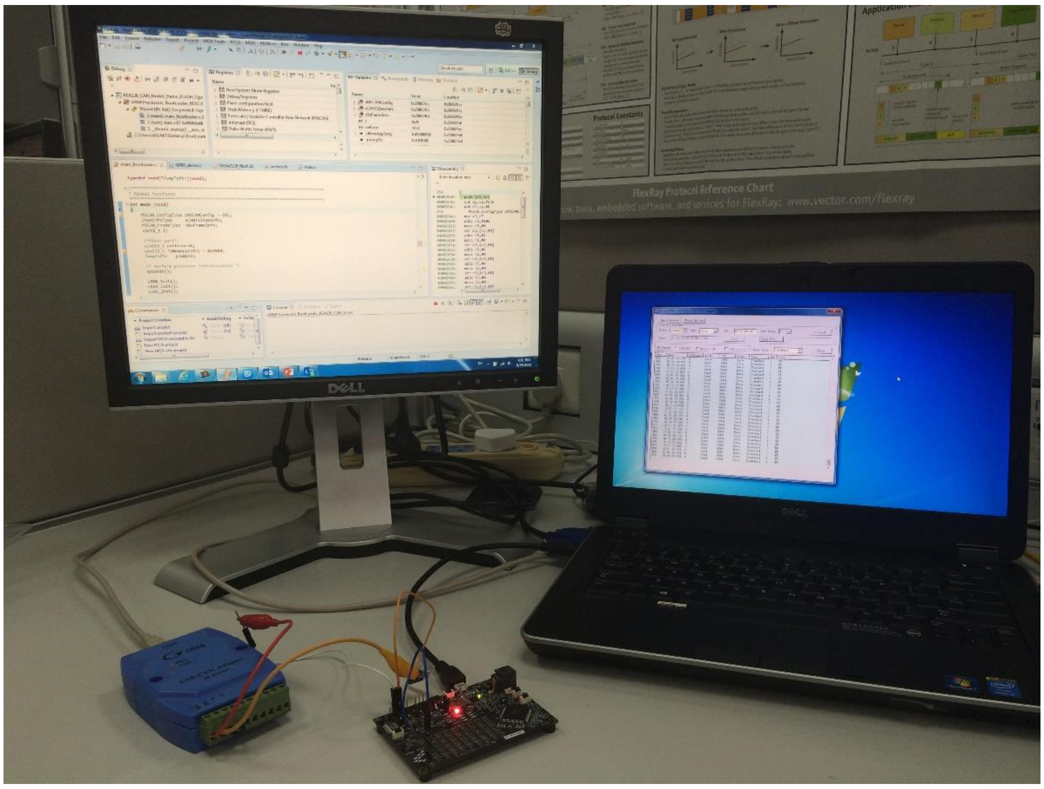


Figure 4. Photo of system

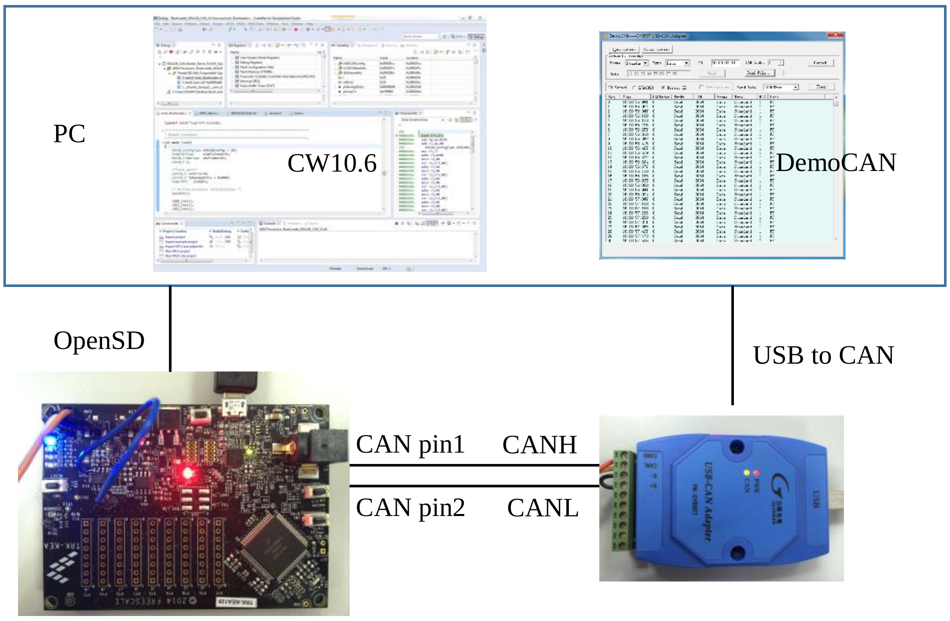
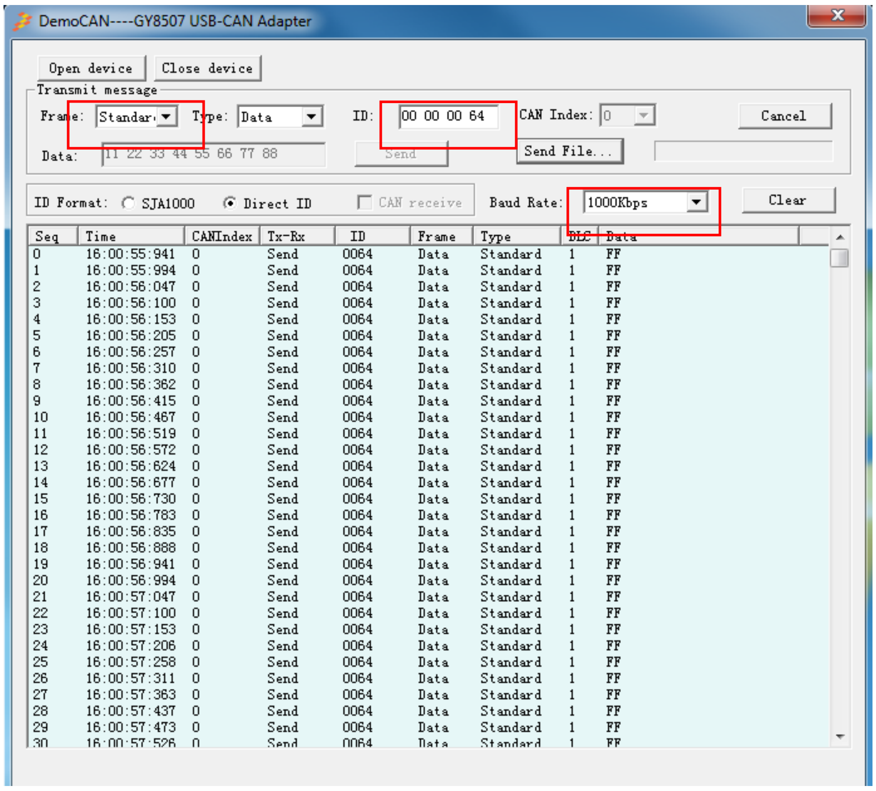


Figure 5. Schematic diagram

Step two: use one of the application template to build a project, and get the .hex file. Then, modify the suffix from .hex to .s19 which is mentioned on page six.

Step three: open the DemoCAN software and select the baud rate with 1000Kbps. Make sure the Frame is “Standard” and ID is “00 00 00 64”. Then, click the “open device” to make the software connect to the USB-CAN adapter. Click “Send File” to select an s19 file. The software begin to send “FF” to the CAN bus, as shown in figure 6 below.

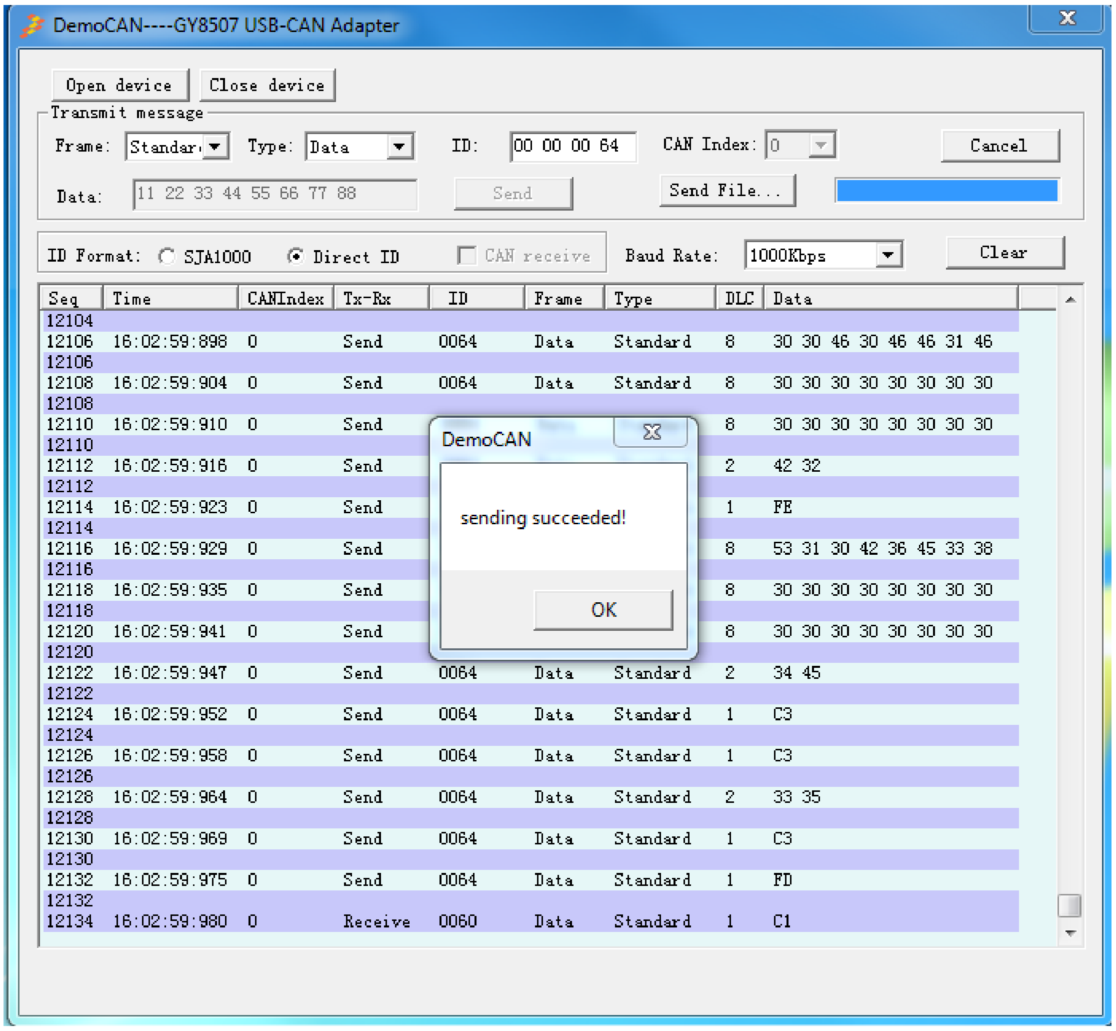


Frame: Standard Type: Data ID: 00 00 00 64 CAN Index: 0
Data: 11 22 33 44 55 66 77 88
ID Format: ( ) SJA1000 [✓ Direct ID] [ ] CAN receive Baud Rate: 1000Kbps

Seq	Time	CANIndex	Tx-Rx	ID	Frame	Type	DLC	Data
0	16:00:55:941	0	Send	0064	Data	Standard	1	FF
1	16:00:55:994	0	Send	0064	Data	Standard	1	FF
2	16:00:56:047	0	Send	0064	Data	Standard	1	FF
3	16:00:56:100	0	Send	0064	Data	Standard	1	FF
4	16:00:56:153	0	Send	0064	Data	Standard	1	FF
5	16:00:56:205	0	Send	0064	Data	Standard	1	FF
6	16:00:56:257	0	Send	0064	Data	Standard	1	FF
7	16:00:56:310	0	Send	0064	Data	Standard	1	FF
8	16:00:56:362	0	Send	0064	Data	Standard	1	FF
9	16:00:56:415	0	Send	0064	Data	Standard	1	FF
10	16:00:56:467	0	Send	0064	Data	Standard	1	FF
11	16:00:56:519	0	Send	0064	Data	Standard	1	FF
12	16:00:56:572	0	Send	0064	Data	Standard	1	FF
13	16:00:56:624	0	Send	0064	Data	Standard	1	FF
14	16:00:56:677	0	Send	0064	Data	Standard	1	FF
15	16:00:56:730	0	Send	0064	Data	Standard	1	FF
16	16:00:56:783	0	Send	0064	Data	Standard	1	FF
17	16:00:56:835	0	Send	0064	Data	Standard	1	FF
18	16:00:56:888	0	Send	0064	Data	Standard	1	FF
19	16:00:56:941	0	Send	0064	Data	Standard	1	FF
20	16:00:56:994	0	Send	0064	Data	Standard	1	FF
21	16:00:57:047	0	Send	0064	Data	Standard	1	FF
22	16:00:57:100	0	Send	0064	Data	Standard	1	FF
23	16:00:57:153	0	Send	0064	Data	Standard	1	FF
24	16:00:57:206	0	Send	0064	Data	Standard	1	FF
25	16:00:57:258	0	Send	0064	Data	Standard	1	FF
26	16:00:57:311	0	Send	0064	Data	Standard	1	FF
27	16:00:57:363	0	Send	0064	Data	Standard	1	FF
28	16:00:57:437	0	Send	0064	Data	Standard	1	FF
29	16:00:57:473	0	Send	0064	Data	Standard	1	FF
30	16:00:57:526	0	Send	0064	Data	Standard	1	FF

Figure 6. DemoCAN user interface after loading file

Step four: run the boot loader demo. The s19 file is downloaded from PC to MCU. Use the LED2 and LED3 as status indicators. LED2 blinks and the CAN frames are shown in the DemoCAN software. After finish update, a popup with “sending succeeded” feedback to user. The update is success as shown in figure 7. If there are some errors in flash program, LED3 is lighting.



Frame: Standard Type: Data ID: 00 00 00 64 CAN Index: 0

Data: 11 22 33 44 55 66 77 88

ID Format: SJA1000 [✓ Direct ID] CAN receive Baud Rate: 1000Kbps

Seq	Time	CANIndex	Tx-Rx	ID	Frame	Type	DLC	Data
12104								
12106	16:02:59:898	0	Send	0064	Data	Standard	8	30 30 46 30 46 46 31 46
12106								
12108	16:02:59:904	0	Send	0064	Data	Standard	8	30 30 30 30 30 30 30 30
12108								
12110	16:02:59:910	0	Send				8	30 30 30 30 30 30 30 30
12110								
12112	16:02:59:916	0	Send				2	42 32
12112								
12114	16:02:59:923	0	Send				1	FE
12114								
12116	16:02:59:929	0	Send				8	53 31 30 42 36 45 33 38
12116								
12118	16:02:59:935	0	Send				8	30 30 30 30 30 30 30 30
12118								
12120	16:02:59:941	0	Send				8	30 30 30 30 30 30 30 30
12120								
12122	16:02:59:947	0	Send	0064	Data	Standard	2	34 45
12122								
12124	16:02:59:952	0	Send	0064	Data	Standard	1	C3
12124								
12126	16:02:59:958	0	Send	0064	Data	Standard	1	C3
12126								
12128	16:02:59:964	0	Send	0064	Data	Standard	2	33 35
12128								
12130	16:02:59:969	0	Send	0064	Data	Standard	1	C3
12130								
12132	16:02:59:975	0	Send	0064	Data	Standard	1	FD
12132								
12134	16:02:59:980	0	Receive	0060	Data	Standard	1	C1

DemoCAN

sending succeeded!

OK

Figure 7. DemoCAN user interface after download

4.2 Power-on mode

Step one: refer to 4.1 section step one as mentioned above.

Step two: refer to 4.1 section step two as mentioned above.

Step three: refer to 4.1 section step three as mentioned above.

Step four: Power-on the KEA or reset the KEA (On our KEAZ128 demo board, press the “SW3” button to reset). The boot loader will detect whether there is a frame of “FF” in CAN bus. If detecting the “FF” during the waiting time, the MCU hook up with PC and update the application.