Freescale Semiconductor, Inc.
Application Note

Document Number: AN4874
Rev. 0, 03/2014

KE06 CAN Boot Loader Design

by: Jonson Chen

1 Overview

There are many applications or products need to upgrade firmware in field to fix some bugs or sometimes to improve the performance. Most of these applications and products do not use the dedicated debug interface, but only use the communication interfaces, such as UART, USB, I^2C, and so on. In this case, a serial boot loader is required to perform firware upgrade via one of the communication interfaces without debugger or dedicated program tools.

This application note provides the guidelines to design boot loader on KE06 MCU with CAN interface.

Contents

1 Overview 1
2 Introduction 2
3 Software architecture 2
3.1 Convert board 2
3.2 Target board 4
4 Memory allocation 9
5 Conclusion 10
6 References 10
7 Acronyms and abbreviations 10
8 Revision history 11

2 Introduction

Boot loader is a built-in firmware, which is implemented to program the application code to flash through the communication interface. This application note describes the procedure to use FRDM-KE06Z board to convert the UART data from PC terminal to CAN bus. In addition, it explains the procedure to communicate with the target board, FRDM-KE06Z, to implement the updates of target application code.

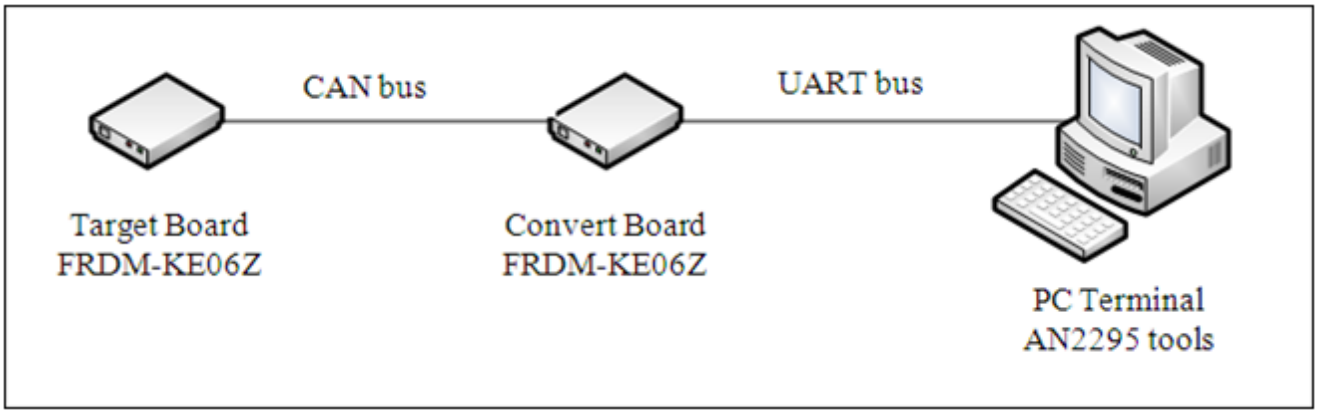

Figure 1. Top-level view

The boot loader is using the features of AN2295SW_Rev1 software tool, which is widely used in all the Kinetis devices to update application code through an UART interface.

The convert board uses freedom board, FRDM-KE06Z, to convert the UART bus to CAN bus and to repackage the data transfer to the target board. The target board will program the application code to flash.

The CAN boot loader sample code can directly run on the FRDM-KE06Z board, and it will be downloaded to the target board, “Bridge_UARTToCAN” to Convert board, and project “RTC_demo” is for generating S19 file, which can be downloaded using PC software.

3 Software architecture

Win_hc08sprg.exe software decodes S19 file and communicate with convert board through FC protocol.

3.1 Convert board

The PC cannot communicate with the target board via CAN directly, for this, we need a convert board to transfer the UART signal (PC end) into CAN signal (target board). Therefore, the convert board communicates with PC terminal through FC protocol. The convert board repackages data frame with data length and checksum to receive or transmit data package with target board using CAN bus.

Table 1. Package content

Data length	Original data frame	Checksum

Then it reads data from the target board. The first data received determines if the target board is ready or not. If it is ready (command |0x80), then it indicates correct acknowledge is received to the receiver, for example, the command send to the target board is 0x03 and the received ACK should be 0x03|0x80.

At first, it will send FC_CMD_HOOK (0x02) to the target board and then read status from the target board to check if it works in boot loader mode or user code mode. If the received state is FC_CMD_HOOK|0x80, then it will send 0xFC to start hook with PC terminal, otherwise, it will always check state of target board till receiving FC_CMD_HOOK|0x80. The software flowchart of convert board is as follows:

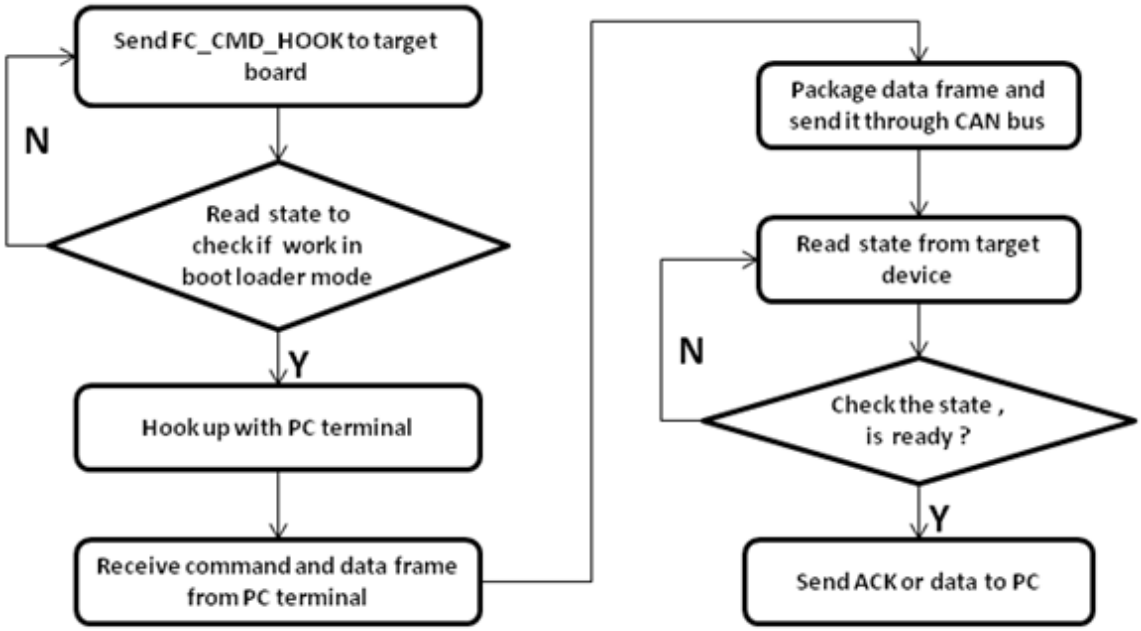

Figure 2. Software flowchart of convert board

The convert board functions as a bridge between PC terminal and target board, using that the S19 file can be downloaded to the target board from PC.

3.2 Target board

The target board contains built-in boot loader code. After startup, the target board first checks the work mode of the boot loader code, that is, whether it is in boot mode or user code mode. There are various methods to perform this check. For example, check the level of an external GPIO, if the GPIO pin is low, then it will enter into boot mode to run boot loader and if the GPIO pin is high, then it will enter user code mode to run application code.

Because of the limited GPIO resources in MCU, the GPIO pins might not be available for the boot loader detecting mode. In this case, a different method is introduced to determine work mode through hook up command. If overtime occurs and hook up fails, then the GPIO enter into user mode, and if it succeeds, then the GPIO enter into boot loader mode.

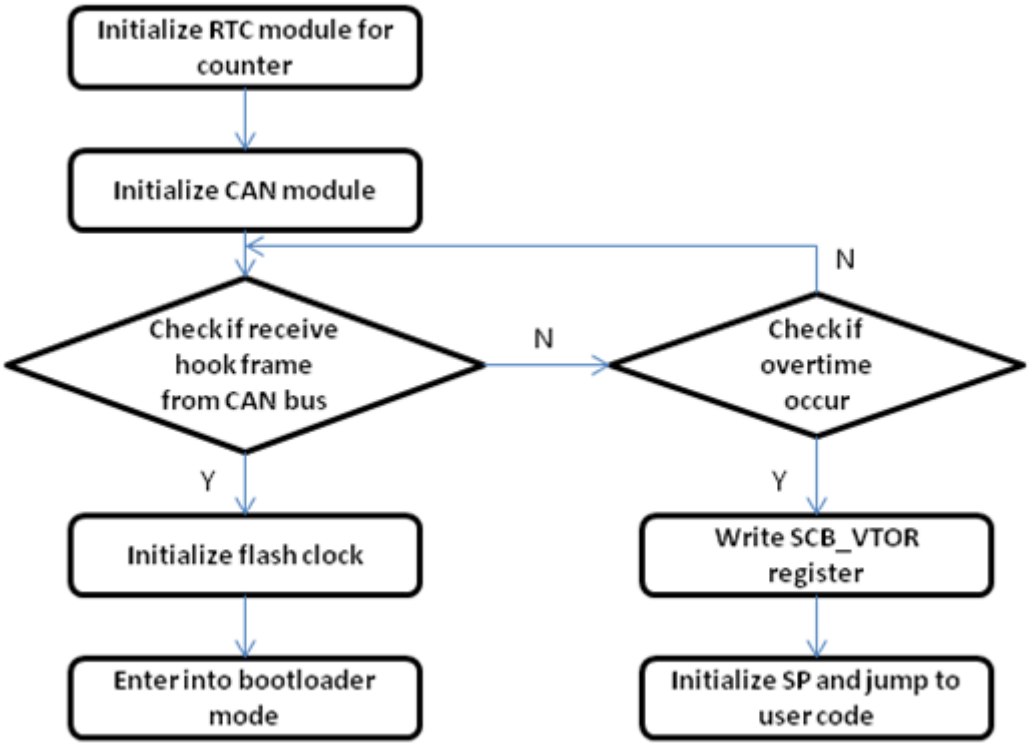

Figure 3. Check flag flowchart

3.2.1 CAN node driver

The target board configures CAN as a CAN node. It receives and transmits data in CAN interrupt service routine. For more information on CAN interrupt flow, see KE06Z reference manual. The sample code snippet of CAN driver, for CAN communication processing, is as follows:

```
void MSCAN_RxProcessing( void )
{
    u32RxInterruptCounter ++;
    if(CAN_IsRxBuffFull(MSCAN))
    {
```

```
        if(u8RxFrameBufferFreeLength!=0)
        {
            CAN_ReadOneFrameFromBuff(MSCAN,&sRxFrame[u8RxFrameBufferIndex++]);
            if(u8RxFrameBufferIndex>=CAN_BUFFER_LENGTH)
            {
                u8RxFrameBufferIndex = 0;
            }
            u8RxFrameBufferFreeLength--;
        }
        else
        {
            //receive frame buffer is full, clear buffer in MSCAN
            // clear receiver full flag
            CAN_ClearRXF_Flag(MSCAN);
}
    }
}

void MSCAN_TxProcessing( void )
{
    if(CAN_IsOverRunFlag(MSCAN))
    {
        // overrun error occur
        CAN_ClearOVRIF_Flag(MSCAN);
    }
    if(CAN_IsWakeUpIntFlag(MSCAN))
    {
        CAN_ClearWUPIF_Flag(MSCAN);
    }
    if(CAN_IsStatusChangeFlag(MSCAN))
    {
        CAN_ClearCSCIF_Flag(MSCAN);
        // Get currently status
        CAN_GetReceiverStatus(MSCAN);
        CAN_GetReceiveErrorCount(MSCAN);
    }
    if(!CAN_CheckSendBufferFrame(MSCAN,&sCAN_TxBuff))
    {
        // no data in transmitting buffer,disbale interrupt
        CAN_TransmitterEmptyIntDisable(MSCAN);
    }
}
```

CAN set gbI2CRecFrameFlag flag after it receives data frame so that the application code can further process data frame.

3.2.2 Command description

Always check gbI2CRecFrameFlag flag in boot loader loop, the boot loader starts to process the received frame when the flag is 1. First of all, use checksum to verify if received frame is correct or not. After verifying, unpack the frame and process the appropriate command.

Table 2. Command package

Total data length(4 bytes)	Command (1 bytes)	Address(4 bytes)	Number of data(1 bytes)	Data	Checksum (1 bytes)

The overview of all the commands is described in the following table:

Table 3. Command list

Command function	Command	Loader positive acknowledge	Loader negative acknowledge
Hook up	0x02	0x82,0xFC	0x82,0x03
Ident	0x49	0xc9,ident information	0xc9,0x03
Erase sector	0x45	0xc5,0xfc	0xc5,0x03
Write	0x57	0xd7,0xfc	0xd7,0x03
Read	0x52	0xd2,data	0xd2,0x03
Quit	0x51	No ack	No ack

3.2.2.1 Hook up

The data package received by hook up command (coded as 0x02) is as follows:

Table 4. Hook up command package

Total data length(4 bytes)	Command (1 byte)	Address(4 bytes)	Number of data(1 byte)	Data	Checksum (1 byte)
6	0x02	-	-	-	CS

The following table represents the command acknowledged by Hook up command:

Table 5. Hook up command acknowledge

Command (1 bytes)	Status
0x82	0xFC/0x03

If the status code received is 0xFC it indicates the target board works in boot loader mode, and is ready to communication with the convert board.

If the status code is 0x03, is the target board is in user mode, and can’t receive any other commands.

3.2.2.2 Ident Command

The data package received by Ident command (coded as 0x49) is as follows:

Table 6. Ident command package

Total data length(4 bytes)	Command (1 bytes)	Address(4 bytes)	Number of data(1 bytes)	Data	Checksum (1 bytes)
6	0x49	-	-	-	CS

The information required for MCU is as follows:

- Protocol version – 1 bytes
- System Device Identification Register (SDID) content (0x06080000) for the KE06 80LQFP, r(23-20 bits) is the chip revision number reflecting the current silicon level – 2 bytes
- Number of reprogrammable memory areas – 4 bytes
- Start address of the reprogrammable area – 4 bytes
- End address of reprogrammable memory area – 4 bytes
- Address of the original vector table (1 KB) – 4 bytes
- Address of the new vector table (1 KB) – 4 bytes
- Length of the MCU erase blocks – 4 bytes
- Length of the MCU write blocks – 4 bytes
- Identification string, zero terminated – n bytes

The following structure is used to identify the MCU information:

```
typedef uint32_t addrtype;
typedef struct
{
          unsigned char Reserve ;              // reserve bytes for 4  bytes allign
          unsigned char Version;              // version
          uint16_t Sdid;                      // Sd Id */
          addrtype BlocksCnt;                 // count of flash blocks
          addrtype FlashStartAddress;         // flash blocks descriptor
          addrtype FlashEndAddress;
          addrtype RelocatedVectors;           // Relocated interrupts vector table
          addrtype InterruptsVectors;         // Interrupts vector table
          addrtype EraseBlockSize;            // Erase Block Size
          addrtype WriteBlockSize;            // Write Block Size
          char IdString[ID_STRING_MAX];       // Id string
}FC_IDENT_INFO;
```

The following table represents the command acknowledged by Ident command:

Table 7. Ident command acknowledge

Command (1 bytes)	Data
0xc9	Ident information

3.2.2.3 Erase command

The data package received by Erase command (coded as 0x45) is as follows:

Table 8. Erase command package

Total data length(4 bytes)	Command (1 bytes)	Address(4 bytes)	Number of data(1 bytes)	Data	Checksum (1 bytes)
10	0x45	Address	-	-	CS

The following table represents the command acknowledged by Erase command:

Table 9. Erase command acknowledge

Command (1 bytes)	Status
0xc5	0xFC/0x03

3.2.2.4 Write command

The data package received by Write command (coded as 0x57) is as follows:

Table 10. Write command package

Total data length(4 bytes)	Command (1 bytes)	Address(4 bytes)	Number of data(1 bytes)	Data	Checksum (1 bytes)
Total length	0x57	Address	Data length	Data	CS

The following table represents the command acknowledged by Write command:

Table 11. Erase command acknowledge

Command (1 bytes)	Status
0xd7	0xFC/0x03

3.2.2.5 Read command

The data package received by Read command (coded as 0x52) is as follows:

Table 12. Read command package

Total data length(4 bytes)	Command (1 bytes)	Address(4 bytes)	Number of data(1 bytes)	Data	Checksum (1 bytes)
11	0x52	Address	Data length to be read	-	CS

The following table represents the command acknowledged by Read command:

Table 13. Read command acknowledge

Command (1 bytes)	Data
0xd2	Data

3.2.2.6 Quit command

Acknowledgement for this command is not required.

After the Quit command is received, the target board will quit the boot loader mode and enter the user mode.

4 Memory allocation

The boot loader code occupies the first region of the FLASH memory (the lowest memory address space).This placement moves the beginning of the available memory space and it is necessary to shift this address in the user application linker files (ICF file in IAR and in LCF file in CodeWarrior). The following code snippet demonstrates the method to modify the ICF linker file in IAR6.5:

```
// default linker file
define symbol __ICFEDIT_region_ROM_start__ = 0x00;
// modified Linker file for KE06Z 128KB flash
define symbol __ICFEDIT_region_ROM_start__ = 0x1000;
```

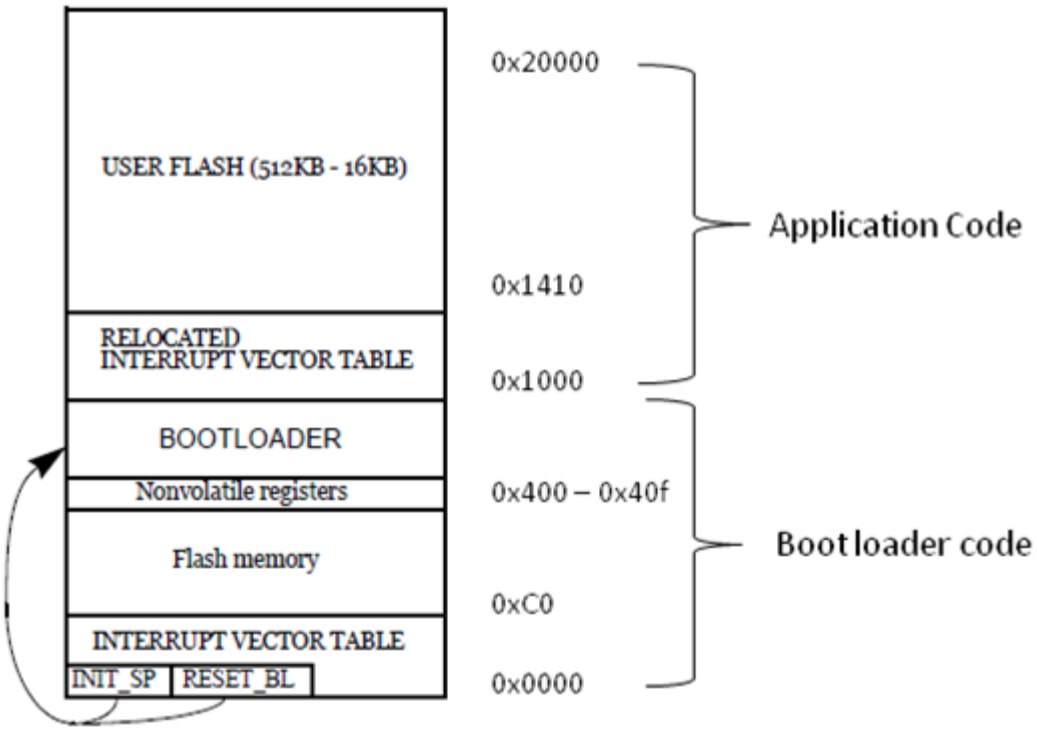

Figure 4. Memory allocation

5 Conclusion

This application note describes the procedure to implement CAN boat loader by using a bridge board as a convert board and other board as target board. Users can also add boot loader functionality by in the application software.

6 References

Following references are available on freescale.com:

- *Developer's Serial Bootloader application note* (document AN2295)

7 Acronyms and abbreviations

Table 14. Acronyms

Term	Meaning
UART	Universal Asynchronous Receiver/Transmitter
CAN	Controller Area Network
FCCOB	Flash Common Command Object
WDOG	Watchdog

Table 14. Acronyms

Term	Meaning
MCG	Multipurpose Clock Generator

8 Revision history

Table 15. Revision history

Revision number	Date	Substantial changes
0	03/2014	Initial release

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number: AN4874
Rev. 0
03/2014