Design of Smart Meters with Upgrade Capability

Authors: Zhang Yuming, Zhang Zhen, Xu Jingsheng

Date: 2024-01-16T00:00:00+00:00

Abstract

In the power industry, the near-infrared interface of smart meters offers an innovative solution for enabling online upgrades. This paper presents a design that effectively leverages the near-infrared interface through both software and hardware implementations to facilitate online upgrades of smart meters. With respect to software design, advanced algorithms and programmatic architectures are employed to ensure the efficiency and stability of the upgrade process. Regarding hardware design, emphasis is placed on hardware compatibility of smart meters to ensure adaptability across devices from different models and manufacturers.

Full Text

Design of Intelligent Electricity Meter with Upgrade Function

ZHANG Yuming[1], **ZHANG Zhen**[2], **XU Jingsheng**[1] [1]Holley Technology Co., Ltd., Hangzhou, Zhejiang 310023, China [2]Huaneng Jinan Huangtai Power Generation Co., Ltd., Jinan, Shandong 250100, China

Abstract: In the power industry, the near-infrared interface of smart meters provides an innovative solution for online upgrades. This paper presents software and hardware designs that effectively utilize the near-infrared interface to achieve online upgrades of smart meters. In terms of software design, advanced algorithms and program design ensure the efficiency and stability of the upgrade process. In terms of hardware design, emphasis is placed on hardware compatibility to ensure adaptability to different models and manufacturers.

Keywords: smart electricity meter; near-infrared interface; power industry

There are two methods for upgrading smart meter programs. The first method uses an offline programmer: the program to be upgraded is first burned from a computer into the offline programmer via specialized software, and then the offline programmer burns the program into the module requiring the upgrade.

The second method directly uses the chip's development environment with an emulator for direct burning, which is generally only possible in factory production mode.

2 Program Upgrade Scheme

[Figure 1: see original paper] Overall Scheme Block Diagram

For smart meters already deployed in the field or sealed finished products ready for shipment, the MCU is HT5017, whose FLASH and RAM cannot meet the program space and data space requirements for upgrades. Therefore, one EEPROM chip and one FLASH chip are externally configured. The program storage area diagram is shown in Figure 2 [Figure 2: see original paper].

The smart meter hardware consists of four main modules: a communication module, power supply module, EEPROM storage module, and FLASH storage module. The communication module exchanges data with external devices via serial port, which can be either near-infrared or other communication methods. The power supply module provides required power for different units of the smart meter. The EEPROM storage module provides and stores information needed for upgrades. The FLASH storage module can store application program image files requiring upgrades. The overall scheme block diagram is shown in Figure 1 [Figure 1: see original paper].

The specific program upgrade scheme divides the HT5017 internal FLASH space into two regions: BOOT and APP. The BOOT region stores relevant boot code with a starting address of 0x0000H. Upon program reset, the BOOT region at address 0x0000H executes first. The BOOT region stores jump instructions that directly execute the APP program content at address 0x4000H. The BOOT program continuously checks in a `while(1)` loop whether an upgrade is needed. If an upgrade is required, the upgrade program is placed into external FLASH. When the upgrade completes, the upgrade completion flag is set. Upon reset, the program is copied from FLASH to the MCU's internal FLASH. After copying completes, the system resets again and executes the new APP program from BOOT [1].

3.1 Hardware Scheme

As program functions expand, one EEPROM chip and one FLASH chip can be used to expand storage space by selecting storage chip models that meet customer requirements—these chips have identical interfaces. Two communication interfaces are simulated through the MCU HT5017's GPIO ports via software, including one I2C bus and one FLASH bus.

3.2 BOOT Software

The BOOT software handles initialization, upgrade judgment, and exception handling. The main BOOT software flow is as follows:

```
int main(void)
{
    Init_{System}(); // Initialize system
    if (upgrade_{judge}() == TRUE) // Check if upgrade is needed
    {
        Upgrade(); // Upgrade module
    }
    // Jump to APP program execution
}
```

3.3 APP Software

APP software primarily completes metering functions including measurement, display, time-of-use switching, communication, and upgrades. The main APP software flow is as follows [2]:

```
int main(void)
{
    Init_{System}(); // Initialize system
    while(1)
    {
        if(GetAcPowerStatus() == Power_{Off}) // Check if power is off
        {
            SetEventCurrStatus(STATUS_{{{POWER}}_{{OFF}}}, OCCURRENCE); // Set power-off oc
            PowDownSaveData(); // Save data on power down
        }
        WatchDog(); // Feed watchdog
        Metering(); // Metering module
        WatchDog(); // Feed watchdog
        Time(); // Time-of-use module
        WatchDog(); // Feed watchdog
        Display(); // Display module
        WatchDog(); // Feed watchdog
        Comm(); // Communication module
        WatchDog(); // Feed watchdog
    }
}
```

4 Upgrade Function Operation Process

(1) Modify the FLASH offset address in the corresponding .icf file in BOOT as follows:

```
/* User application program offset address */
define symbol Flash_{Offset} = 0x4000;

/*-Specials-*/
```

```
define symbol __{{{ICFEDIT}}_{{intvec}}}_{start}_ = 0x0+Flash_{Offset};
define symbol __{{{ICFEDIT}}_{{password}}}_{start}_ = 0x00000FC0 + Flash_{Offset};

/*-Memory Regions-*/
define symbol __{{{ICFEDIT}}_{{region}}}_{{{ROM}}_{{start}}}_ = 0x0 + Flash_{Offset};
define symbol __{{{ICFEDIT}}_{{region}}}_{{{ROM}}_{{end}}}_ = 0x0003FFFF;
define symbol __{{{ICFEDIT}}_{{region}}}_{{{RAM}}_{{start}}}_ = 0x20000000;
define symbol __{{{ICFEDIT}}_{{region}}}_{{{RAM}}_{{end}}}_ = 0x20007FFF;

/*-Sizes-*/
define symbol __{{{ICFEDIT}}_{{size}}}_{cstack}_ = 0x0400;
define symbol __{{{ICFEDIT}}_{{size}}}_{heap}_ = 0x0000;
/**** End of ICF editor section. ###ICF###*/
```

Use Notepad software to modify the bolded portions in the HT501X.icf file above. For BOOT, if the HT501X.icf file configuration needs the bold portion changed to `define symbol Flash_{Offset}=0x0000`, then when the user application program offset address is 0x4000, it indicates BOOT is required. When the user application program offset address is 0x0000, it indicates BOOT is not required.

(2) After compiling the entire project source code with the compiler, generate the corresponding application program app.bin file.

(3) Use the manufacturer's specialized download software (shown in Figure 3 [Figure 3: see original paper]) to convert the app.bin file into app.hex file, which can then be directly downloaded to the smart meter during compilation.

(4) After downloading to the smart meter, the host computer can read the software version number after the upgrade to confirm whether the upgrade was successful.

5 Upgrade Process

The overall program upgrade process is shown in Figure 4 [Figure 4: see original paper].

If the upgrade image package is not fully received, missing image packages must be retransmitted. There are two specific retransmission processes: when multiple blocks have not received data, the retransmission process shown in Figure 5 [Figure 5: see original paper] can be used; when a single block has not received data, the retransmission process shown in Figure 6 [Figure 6: see original paper] can be used.

[Figure 5: see original paper] Multi-block Retransmission Flowchart [Figure 6: see original paper] Single-block Retransmission Flowchart

The packet number starts from 0, meaning if the first packet is lost, the first

successfully transmitted block number during retransmission is 0. When all data blocks have been transmitted, the first unsuccessfully transmitted data block number equals the total number of packets, indicating successful data block transmission.

Conclusion

This paper designs a smart meter capable of program upgrades based on the characteristics of smart meters requiring program updates. When upgrade functionality is needed, there is no need to open the meter; the meter's functions can be upgraded simply through the communication port (near-infrared interface). If the meter was designed with external interfaces such as GPRS, RF, or RS485, the upgrade functionality can also be extended to smart meters with these communication interfaces.

[1] HU Baoling, CUI Yuhang, XIE Sibo, et al. Design of multifunctional digital electricity meter based on wireless communication [J]. Henan Science and Technology, 2023, 42(15): 5-8. [2] JIN Song, XU Zhong. Design of smart meter communication scheme based on LoRa technology [J]. Automation Application, 2023, 64(6): 123-124+136.

Note: Figure translations are in progress. See original paper for figures.