Bootloader Design Considerations

Introduction

Most modern embedded systems require in-field firmware updates to fix errors or improve functionality. Typically, this update functionality is implemented through a bootloader. A bootloader is a special application that is independent from the main application, and capable of updating the main application.

This document describes some of the details that must be considered while designing a robust bootloader.
# Table of Contents

- Introduction......................................................................................................................1
- Types of Bootloaders and Firmware Update Options..................................................3
- Method of Entry............................................................................................................4
- Communication Between the Application and Bootloader..........................................5
- Image Integrity Verification..........................................................................................6
- Security.........................................................................................................................7
- Failure Detection and Recovery....................................................................................8
- The Microchip Web Site...............................................................................................9
- Customer Change Notification Service.........................................................................9
- Customer Support..........................................................................................................9
- Microchip Devices Code Protection Feature..................................................................9
- Legal Notice.................................................................................................................10
- Trademarks..................................................................................................................10
- Quality Management System Certified by DNV............................................................11
- Worldwide Sales and Service......................................................................................12
1. **Types of Bootloaders and Firmware Update Options**

Microcontroller bootloaders can work in several different ways depending on the complexity of the hardware and availability of the external communication interfaces to the user.

The most basic bootloader can receive the image over a communication interface and write the received image after the integrity verification. Mostly the bootloader is not updatable and limited to simple interfaces, such as UART, SPI or I²C. These types of interfaces are typically not accessible to the user of the finished product, hence they are suited for use in a slave MCU, where there is already an established communication interface to the host controller.

The advanced bootloaders will implement interfaces accessible to the user, such as USB, SD Card, or Ethernet. This results in a bigger bootloader size and the need to update the bootloader itself to fix flaws in the protocol implementation, or adapt to changes in external devices. For example, if the original version of the product only supported SD Cards up to 2 GB of capacity, but then over time the application was updated to support higher capacity cards, the old bootloader will still require 2 GB cards, which may be harder to find.

If an advanced update is required, it is worth evaluating if the main application can receive and store the image in a low-level format, hence it can be used by the simple bootloader. This approach makes it possible to implement advanced update interfaces, while keeping the actual bootloader simple and compact.

When an update is required on a wireless system, it becomes impractical to duplicate the wireless stack in the bootloader. Therefore, the only option of updating wireless systems over the air is to receive the new image while application is running, and then let a simple bootloader perform the update. The image may be stored in the free space of the main Flash array, or in the external memory. When stored in the external memory, the image must be kept encrypted, but when stored internally, the image may be decrypted and verified, simplifying the design of the bootloader.
2. **Method of Entry**

The simple bootloaders that are designed to be used with a host controller can use any of the following entry:

- A dedicated pin that must be asserted at the time of reset
- Reuse one of the interface pins
- Application request
- Absence of valid application

A dedicated pin is the most robust way, but it requires the allocation of an additional pin on the host and the slave MCUs, which is not always possible.

Many interfaces include pins in a known idle state, for example, Chip Select in SPI or SCL, and SDA lines in I²C. Those pins are normally held at their idle state using external pull-up resistors, hence the slave may detect an asserted state on reset and remain in the bootloader mode.

Both of those methods require access to the reset pin of the slave MCU. But they have an advantage of complete robustness. The host can always request the bootloader execution, even if the main application is damaged and the primary communication interface is not functional.

The application request does not require any additional pins, and the request for the bootloader to run happens over the normal host to slave communication interface. This method requires a functional application to act on the request.

The bootloader may also execute if the application area is empty or damaged. The detection algorithm may be as simple as looking for a non-erased state of a Flash location, or as complex as calculating a CRC or hash of the entire application image. The balance between the speed and robustness of the error detection must be evaluated for each application.

More advanced bootloaders may detect that an update file exists on a USB drive or an SD Card, and initiate the update this way.
3. Communication Between the Application and Bootloader

In some cases, the application and bootloader need to communicate additional information. This may be a notification of the new image, a status of the update, or some additional requests.

Some MCUs have a set of general purpose registers specifically allocated for this purpose. Those registers are guaranteed to be reset to ‘0’ on power-up, but preserved over reset cycles. This creates a robust communication channel.

If special registers are not available, then system RAM can be used. While RAM is typically preserved during the reset cycles, and it has an undefined state after the power-up. To avoid false communication, it is recommended to use multiple copies of the command. The command must be considered valid if all copies contain the same value. The number of copies is a tradeoff between the probability of a false positive and memory consumption. At least four copies should be used, and typically using 16 copies of a single byte command is sufficient.

If the RAM is deemed to be unreliable, or reset process of the MCU erases the RAM (possible on some security-oriented devices), then the EEPROM or Flash memory can be used for this purpose. The disadvantage is that true EEPROM is not available on many modern MCUs and the Flash memory has erase and write granularity requirements, which will lead to a lot of wasted Flash space.
4. Image Integrity Verification

In a bootloader there are two distinct places where image integrity verification comes into play.

The integrity of the new image needs to be verified. This step is likely to be mandatory for any bootloader design, because this is the part most vulnerable to unintentional damages due to unreliable interfaces, and intentional damages due to an active attacker. This step is often combined with image authentication if encryption is used, because valid authentication automatically proves image integrity.

Independent from the initial verification, the bootloader may perform an image verification on each reset cycle. This step is often optional, because it will delay the application execution. However, in some cases, if device security is a concern, secondary verification becomes necessary. The attacker may attempt to interrupt an otherwise secure update process leaving the Flash in an unknown state. If a bootloader later attempts to execute the application without verification, it will execute the code that was never intended to run, possibly allowing disclosure of the secret information.

Another way to protect against this type of attack is to have a persistent flag indicating image validity. The flag must be cleared right before the update process and set back after the image verification is successfully completed for the first time. The bootloader must always check the flag before passing control to the application. This way image verification on each reset is not necessary.

If the bootloader design allows for receiving the entire image before the Flash programming starts, then the integrity of the received image should be verified before the programming is attempted. However, in many cases the bootloader must receive and program the image in small chunks. In this case, it is important to have a bootloader entry process that is functional without the valid application image, because an interrupted update process may render the device inoperable. Any bootloader that uses secondary image verification (through a complete verification, or using a flag as described above) is automatically protected against this problem. The bootloader will keep running if the application image is damaged. The host controller must be prepared for this situation, and detect that a bootloader is running. This mandates another update attempt.

The choice of the verification algorithm is a compromise between the speed of verification and the performance. Generally, it is better to use slow, but secure algorithms for new image verification, and fast algorithms (like CRC32) for application integrity verification.
5. Security

If the target MCU has hardware support for CRC or secure hashes, then it is good to use those capabilities for image verification, as they are likely to be faster than an equivalent software solution. If the hardware does not support advanced encryption standards, then software implementations of simpler algorithms often provide a sufficient level of security while maintaining high speed and a low footprint. Simple algorithms like XTEA, RC4, Spritz, and others are generally considered too weak for general use, but their weakness is not a concern for bootloader use. A significant part of the weakness comes from the ability to observe significant amounts of repeated data, or even having the ability to influence the plain text. None of those things are true for bootloaders. An image update is a rare event and the image size is relatively short, so the attacker cannot collect significant amounts of the encrypted data.

It is a good security practice to never use the master key for the actual encryption of the data. The master key must be used to derive the image encryption key based on the information from the image header. This way different images will have different encryption keys further limiting the chances of a successful attack.

In all cases care must be taken to understand the limitations of the algorithm used, and how to apply them properly. A review by a security expert is always advised.

One of the most important aspects of the secure implementation is key storage. All devices must be preconfigured with an encryption key. Assuming a generic MCU with no special security features for key storage, the key may be stored either as part of the bootloader, or in a separate data section. Storing the key as part of the bootloader saves some space in the Flash, since the bootloader rarely occupies the whole allocated section. But the disadvantage is that if the key need to be updated, then the whole bootloader will have to be updated as well.

The key may also be stored in a data section that may already be used by the application to store user data. The disadvantage of this approach is that if an application erases the section to write the new data, it must also erase the key. Normally an application will write it back, but if power cuts off during that time, the key will be lost. The most reliable way to store the key is to allocate a separate sector for the key. Because Flash write granularity is smaller than the erase granularity, it can have many active keys simultaneously without having the risk of losing any of them during the key change procedure.

Some microcontrollers may also provide a separate section that may be suitable for the key storage, like Fuses or User Row.

The initial moment of key provisioning must be done at a secure and trusted location. If the original key leaks, all further key updates will be compromised as well.

Sometimes it is impossible to have the devices programmed in a secure location. In this case a dummy key must be used, and no application image should be programmed. This way the host controller can detect that the slave device has no valid application image and perform the first key update followed by the application image update encrypted using the real key. This step must be performed as a secure location.
6. Failure Detection and Recovery

A common problem with firmware updates is failure detection and recovery.

If the bootloader is communicating with a host processor, then the solution is to retry the update. There is no risk of permanent failure if the host controller has a way to request the bootloader without an application running.

In a case of a standalone bootloader, the recovery is not possible if the bootloader is overwriting the existing image with a new one without making a backup image. The process of writing the new image in place of the active one is the most dangerous part of the update process. There are several ways this danger can be mitigated or minimized.

The bootloader must always perform verification and retry the attempt in case of verification failure. Ultimately the update will either succeed or fail due to a possible hardware issue, which cannot be solved by the software. This means that if the update is interrupted by the power cycle or reset, it will end up in an inconsistent state and the bootloader must perform the update again.

One way to avoid the image switching process entirely is to have the bootloader run the application from either part of the Flash. With this approach, the bootloader will have to remember which part of the Flash contains the active application. One disadvantage of this method is that different images must be linked to work at different addresses. With some architectures and toolchains, it is possible to create a position-independent code, but a more general solution is to build several images, that is, one for each possible location. The update process will then download the image that is matching to the inactive slot. This method can be extended to more than two images, provided that enough memory is available.

Some microcontrollers support dual-pane Flash with the ability to switch the panes with a single atomic command. This hardware feature makes it possible to get away with not having a dedicated bootloader at all. In this case the running application writes the new application to the second pane and performs the switch after the written data is verified. The disadvantage of this approach is that there is no way to detect and recover from malfunctioning or corrupt firmware.

Even with dual-pane Flash it is possible to have a bootloader that will check the image integrity on startup and perform recovery operations in case of error. In this case the same bootloader code must be duplicated twice, one for each pane.

Dual-pane Flash makes it easy to perform the recovery. A simple pane switch is required, because the old image is always available and is known to be functional.

It is assumed that by the time bootloader is ready to pass control to the application, the updated image has been verified. This leaves many potential problem, like a valid application is not functional. This may happen due to inadvertent use of the incorrect image, or some incompatibility of the new image with the environment.

To detect this kind of failure, a bootloader may set a persistent flag indicating that it is about to run a new application image. The application will check the flag and run the functionality tests, or at least ensure that the firmware update channel is available. After that the application will clear the flag. If on the next reset cycle, the bootloader detects that the flag is still set, it will assume that the new image was not running correctly or failed the tests. Then the bootloader must perform the recovery from the backup image, if it is available.
The Microchip Web Site

Microchip provides online support via our web site at http://www.microchip.com/. This web site is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the web site contains the following information:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user’s guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQ), technical support requests, online discussion groups, Microchip consultant program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Customer Change Notification Service

Microchip’s customer notification service helps keep customers current on Microchip products. Subscribers will receive e-mail notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.


Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support

Customers should contact their distributor, representative or Field Application Engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document.

Technical support is available through the web site at: http://www.microchip.com/support

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip’s Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Legal Notice

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE. Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer’s risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

Trademarks

The Microchip name and logo, the Microchip logo, AnyRate, AVR, AVR logo, AVR Freaks, BitCloud, chipKIT, chipKIT logo, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, Heldo, JukeBlox, KeeLoq, Kleer, LANCheck, LINK MD, maXSylus, maXTouch, MediaLB, megaAVR, MOST, MOST logo, MPLAB, OptoLyzer, PIC, picoPower, PICSTART, PIC32 logo, Prochip Designer, QTouch, SAM-BA, SpyNIC, SST, SST Logo, SuperFlash, tinyAVR, UNI/O, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

ClockWorks, The Embedded Control Solutions Company, EtherSynch, Hyper Speed Control, HyperLight Load, IntellIMOS, mTouch, Precision Edge, and Quiet-Wire are registered trademarks of Microchip Technology Incorporated in the U.S.A.


SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

Silicon Storage Technology is a registered trademark of Microchip Technology Inc. in other countries.

GestIC is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

All other trademarks mentioned herein are property of their respective companies.
Quality Management System Certified by DNV

ISO/TS 16949
Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company’s quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip’s quality system for the design and manufacture of development systems is ISO 9001:2000 certified.
# Worldwide Sales and Service

<table>
<thead>
<tr>
<th>AMERICAS</th>
<th>ASIA/PACIFIC</th>
<th>ASIA/PACIFIC</th>
<th>EUROPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Corporate Office</td>
<td>Australia - Sydney</td>
<td>Tel: 61-2-9868-6733</td>
<td>India - Bangalore</td>
</tr>
<tr>
<td></td>
<td>China - Beijing</td>
<td>Tel: 86-10-8569-7000</td>
<td>India - New Delhi</td>
</tr>
<tr>
<td></td>
<td>China - Chengdu</td>
<td>Tel: 86-28-8665-5511</td>
<td>India - Pune</td>
</tr>
<tr>
<td></td>
<td>China - Chongqing</td>
<td>Tel: 86-23-8980-9588</td>
<td>Japan - Osaka</td>
</tr>
<tr>
<td></td>
<td>China - Dongguan</td>
<td>Tel: 86-769-8702-9880</td>
<td>Japan - Tokyo</td>
</tr>
<tr>
<td></td>
<td>China - Guangzhou</td>
<td>Tel: 86-20-8755-8029</td>
<td>Korea - Daegu</td>
</tr>
<tr>
<td></td>
<td>China - Hangzhou</td>
<td>Tel: 86-571-8792-8115</td>
<td>Korea - Seoul</td>
</tr>
<tr>
<td></td>
<td>China - Hong Kong SAR</td>
<td>Tel: 852-2943-5100</td>
<td>Malaysia - Kuala Lumpur</td>
</tr>
<tr>
<td></td>
<td>China - Nanjing</td>
<td>Tel: 86-25-8479-2460</td>
<td>Malaysia - Penang</td>
</tr>
<tr>
<td></td>
<td>China - Qingdao</td>
<td>Tel: 86-632-8502-7355</td>
<td>Philippines - Manila</td>
</tr>
<tr>
<td></td>
<td>China - Shanghai</td>
<td>Tel: 86-21-3326-8000</td>
<td>Singapore</td>
</tr>
<tr>
<td></td>
<td>China - Shenyang</td>
<td>Tel: 86-24-2334-2829</td>
<td>Taiwan - Hsin Chu</td>
</tr>
<tr>
<td></td>
<td>China - Shenzhen</td>
<td>Tel: 86-755-8864-2200</td>
<td>Taiwan - Kaohsiung</td>
</tr>
<tr>
<td></td>
<td>China - Suzhou</td>
<td>Tel: 86-186-6233-1526</td>
<td>Taiwan - Taipei</td>
</tr>
<tr>
<td></td>
<td>China - Wuhan</td>
<td>Tel: 86-27-5980-5300</td>
<td>Thailand - Bangkok</td>
</tr>
<tr>
<td></td>
<td>China - Xian</td>
<td>Tel: 86-29-8833-7252</td>
<td>Vietnam - Ho Chi Minh</td>
</tr>
<tr>
<td></td>
<td>China - Xiamen</td>
<td>Tel: 86-592-2388-138</td>
<td>China - Zhuhai</td>
</tr>
</tbody>
</table>

© 2018 Microchip Technology Inc.