IXF3208 RAM Loader Implementation Guidelines

Application Note

February 2002

Order Number: 250650-001

Contents

Figures

Revision History

Date	Revision	Description
02/11/02	-001	Initial Release.

1.0 Introduction

This document describes the algorithm that should be used in order to load the IXF3208 firmware program into the Intel® IXF3208 device internal RAM. This algorithm should be implemented as part of the microprocessor base system routines. The microprocessor will control all writes and verification from the IXF3208 RAM via its address, data and control bus. The microprocessor is responsible to provide exception handling when an error is detected. The exception can be a retry, or user notification, depending on the implementation.

Throughout the document, BASE is referred as the base address of the IXF3208 device. This address is defined by the user.

1.1 Related Documents

Table 1. Related Documents

Document Title	Document #
IXF3208 Datasheet	249544-001
IXF3208 API Developer's Manual	249880-001
IXF3208D Development CD	

1.2 Firmware Program

The IXF3208 firmware is available in two forms:

1. IXF3208.HEX file
2. IXF3208.H header file

Both files and the RAM Loader example code are included in the IXF3208D Development CD.

Using the IXF3208.HEX file requires the software developer to write software routines that read from the IXF3208.HEX file, load data into an array and write the data to the IXF3208 device. The rest of this document describes this procedure.

The second alternative is more effective for the developer, as all that is needed is to include the IXF3208.H header file into the system code with one build.

The advantage of the first approach is that a developer is not required to touch the entire code every time there is a firmware update, as the IXF3208.HEX is independent of the user code. However, it is not as effective as the second approach, and there is some latency for accessing the external file.

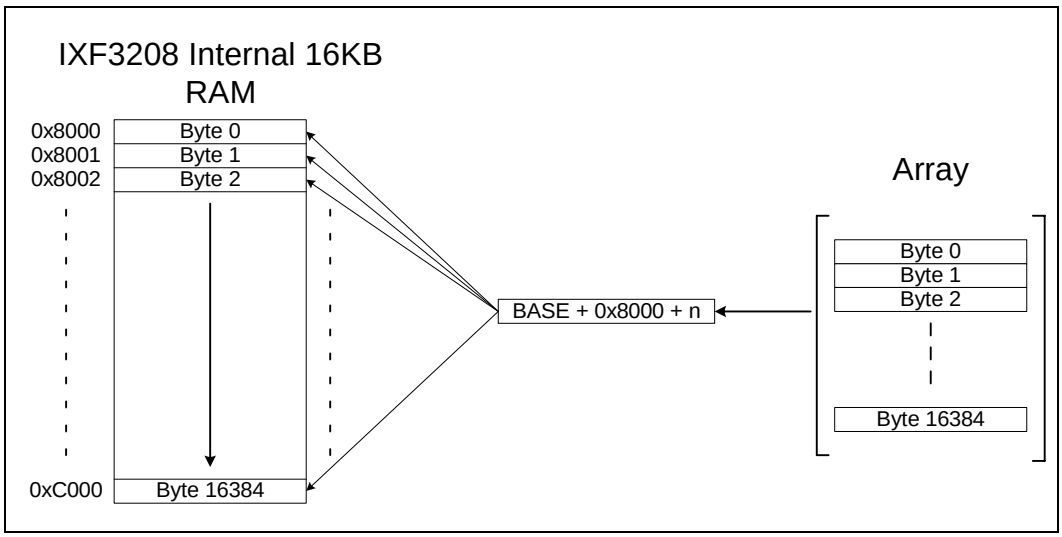
The second approach is more effective and takes into consideration that the firmware is not often updated. However, when there is a firmware update, the user must obtain the updated header file, integrate that into his code and do a complete re-compile.

The choice is left up to the developer.

2.0 IXF3208 RAM Structure

The following diagram describes the internal IXF3208 RAM. The RAM size is 16KB and is organized as 16K x 8-bit.

Figure 1. IXF3208 Internal RAM Structure



In order to load the firmware into the RAM, the microprocessor must load it one byte at a time. First, the firmware is stored in an array of 16KB (16384 bytes). Before loading the first byte from the array, the IXF3208 internal CPU has to be disabled.

The RAM Loader software has to have a pointer to point to BASE +0x8000 address and a counter that increments every time a byte is loaded. BASE is the base address that has been assigned to the device by the user plus 0x8000, which is the starting address of the RAM.

Once the pointer is setup to point to BASE + 0x8000, the RAM Loader software starts to load the first byte into the first address location of the RAM. The software is responsible to increment the pointer in the array for each byte loaded, until all RAM locations are filled (or array is complete). After the firmware is loaded completely, the RAM Loader software will read back all byte locations for comparison.

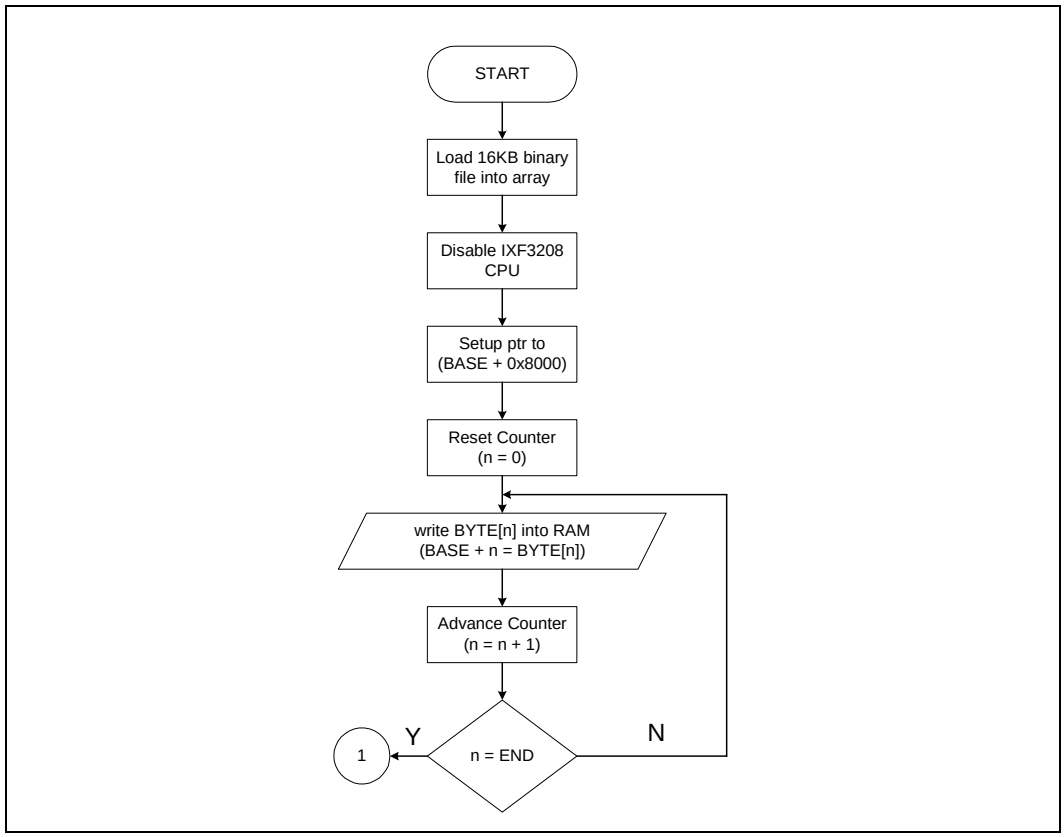
Note: The IXF3208 firmware size is normally smaller than 16KB, therefore, the RAM Loader does not need to load all 16KB into the RAM. In this application note, the algorithm describes loading and verification up to the firmware size (end of bytes in the array).

3.0 RAM Write (Load) Algorithm

The following diagram describes the algorithm and the sequence that should be used for loading the IXF3208 RAM.

- The microprocessor loads the 16KB binary file into an array.
- Disable the IXF3208 CPU using ***IXF3208SetCPUReset(Handle, TE_Enable)*** message.
- Setup the pointer to point to the RAM starting address, which is BASE + 0x8000.
- Reset counter (n = 0)
- CPU writes the first byte pointed by 'n' into (BASE + n) location.
- Each time the CPU writes into the RAM, the counter increments by 1.
- This loop is repeated until all bytes in the array are loaded. Once completed, the software may go through the verification process or terminate the load, depending on the developer's preference.

Figure 2. RAM Load Algorithm



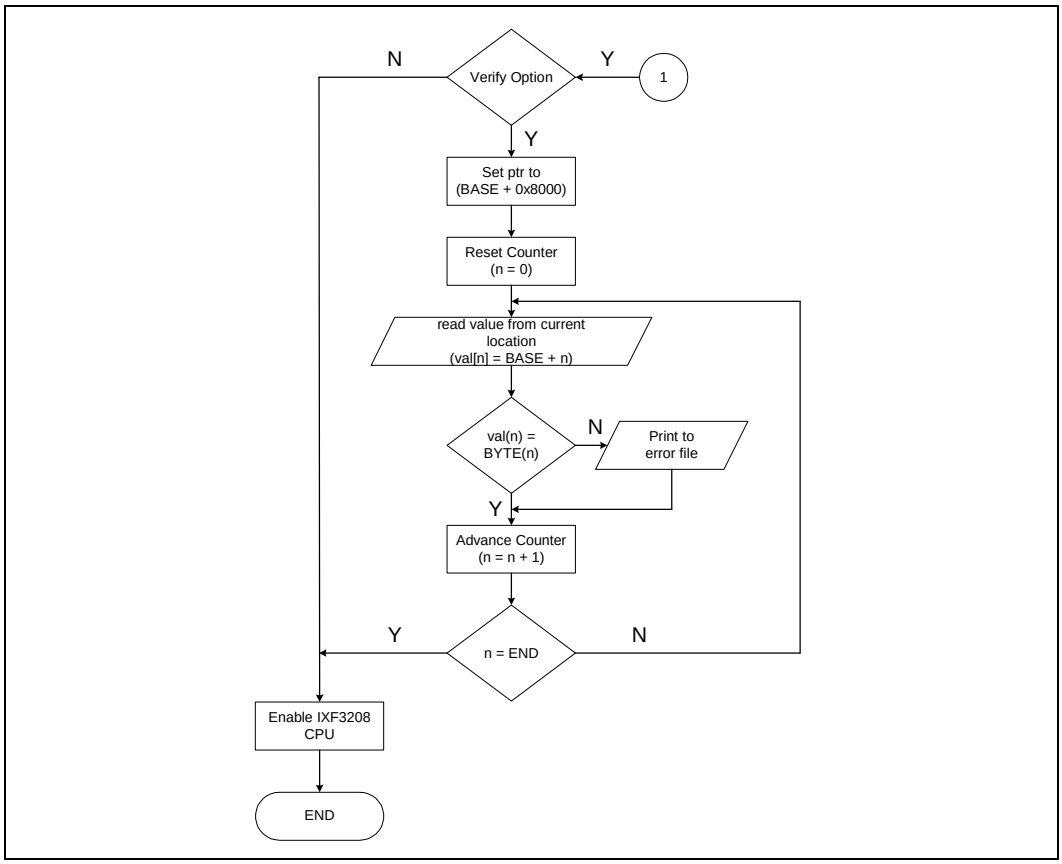
4.0 RAM Read (Verify) Algorithm

The following algorithm can be implemented as an integral part of the RAM Loader algorithm. Once the RAM is loaded completely with the firmware, verification may be used to verify the integrity of the data. In this example, a simple read back and compare is performed for each read byte.

The following is the sequence that should be implemented for verification:

- Setup RAM pointer to point to BASE + 0x8000.
- Reset counter (n = 0)
- The CPU reads the first byte pointed by ‘n’ into [BASE + n] location.
- The RAM Load software performs a comparison, if the read back value is not equal to the corresponding byte location in the array, the CPU performs exception handling or notifies the user of an error. In this example, a simple log file is used.
- If the read byte is equal, the counter is advanced by 1.
- This process is repeated until all bytes are read.
- Enable IXF3208 CPU using ***IXF3208SetCPUReset(Handle, TE_Disable)*** message and terminate.

Figure 3. RAM Read Algorithm



5.0 Acronyms

CPU - Central Processing Unit

RAM - Ramdom Access Memory