



EEPROM Checksum Information for the ADM1060

by Marcus O'Sullivan

INTRODUCTION

This document shows how the checksum for the ADM1060 is calculated, and the locations that the checksum information is then stored in EEPROM at the preprogramming stage. Using this information, it is then possible to extract the data from the preprogrammed device and cross-reference the checksum data with the master .fac file, and also verify that the device has not been modified since preprogramming.

The ADM1060 has a 512 byte EEPROM split into two equal blocks. The first 256 locations (F8 Block) contain all the configuration data for the ADM1060; the second F9 block is for user data.

0xF8 EEPROM	0xF9 EEPROM
0xF800	0xF900
0xF801	0xF901
.	.
ADM1060 CONFIGURATION EEPROM	ADM1060 USER SCRATCHPAD EEPROM
.	.
0xF8DF	.
0xF8E0	.
.	.
FACTORY RESERVED	.
.	0xF9FE
0xF8FF	0xF9FF

Figure 1. ADM1060 EEPROM Structure

The checksums for each of these blocks are saved in a *locked page* of the 0xF8 block. There are three checksums of significance.

1. Configuration File (.fac file) 16-Bit Checksum

This is only a partial checksum of the 0xF8 EEPROM bitmap. Locations such as factory calibration and status registers are excluded. Therefore, the checksum remains the same for all parts programmed with the same configuration. This checksum is listed at the top of the .fac file. Since this checksum is 16 bits and the EEPROM locations are only 8 bits, to store it on each device we must split it into MSB and LSB. The checksum is stored in the following locations:

0xF8FE(MSB) and **0xF8FF**(LSB)

This checksum could be used to verify that the correct program was selected to program the device.

This checksum is calculated from the .fac file as follows:

Add the data in locations “from 0xF800 to 0xF890 inclusive,” then add this to the sum of the data in the following locations “from 0xF898 to 0xF89C inclusive,” then add this to the sum of the data in the following locations “from 0xF8A0 to 0xF8D7 inclusive.” Now invert (bitwise complement) this number then “AND” (&) the result with 0xFFFF.

For example:

checksum = (~(sum of locations: “0xF800 -> 0xF890” + “0xF898 -> 0xF89C” + “0xF8A0 -> 0xF8D7”)) & 0xFFFF

2. Full Configuration EEPROM 8-Bit Checksum

This is an 8-bit checksum of the full configuration 0xF8 EEPROM (excluding the last 32 bytes of locked factory locations). This checksum may be unique for each individual part programmed with the same configuration due to the fact that all calibration data will be included in the checksum calculation. This checksum is stored in the following location:

0xF8F3

This checksum could be used to verify that no configuration data has changed since the device was programmed.

This checksum is calculated as follows:

Add the data in locations “from 0xF800 to 0xF8D7 inclusive,” then invert (bitwise complement) this number then “AND” (&) the result with 0xFF.

For example:

```
checksum = (~sum of locations: "0xF800 -> 0xF8D7")) & 0xFF
```

3. User Scratchpad EEPROM 8-Bit Checksum

This is an 8-bit checksum of the full user 0xF9 EEPROM.

This checksum is stored in the following location:

0xF8F5

This checksum is calculated as follows:

Add the data in locations “from 0xF900 to 0xF9FF inclusive,” then invert (bitwise complement) this number, then “AND” (&) the result with 0xFF.

For example:

```
checksum = (~sum of locations: "0xF900 -> 0xF9FF")) & 0xFF
```