**LDM**

Bootloader

**Contents**

[**Overview** 2](#_Toc38901515)

[**Update functionality** 2](#_Toc38901516)

[**Bootloader first start** 3](#_Toc38901517)

[**Bootloader logic** 4](#_Toc38901518)

[**Application configuration** 5](#_Toc38901519)

[**Crc32 for application verification** 5](#_Toc38901520)

# **Overview**

Bootloader located in MCU Boot Flash memory.

Flash memory of PIC32 divided on two regions Program and Boot memory.

**CONFIG** is current configuration of device.

3 first cells of configuration memory is used for CRC of configuration and two application CRCs.

**APPLICATION** – is a memory for main program. Size equals half of all PIC32MZ1024 FLASH memory.

**BACKUP** – is a memory for LiveUpdate function of bootloader. It has the same size as APPLICATION.

application version OFFSET:0x490

et 0x490

**BACKUP**

CRC1 – Application CRC

CRC2 – Backup CRC

et 0x490

application version OFFSET:0x490

et 0x490

**0x9D000000**

len=0x80000

crc2

crc1

**CONFIG**

VER

VER

**0x9D080000**

len=0x80000

**0xBFC20000**

len=0x4000

**0xBFC00000**

**0x00000000**

**BOOTLOADER**

**APPLICATION**

Microcontroller (MCU) PIC32MZ1024 has 1024kB of Flash memory. APPLICATION size is the same as BACKUP and equals 512kb. Both APPLICATION and BACKUP has version identifier. Offset is 0x490.

CRC of APPLICATION in configuration (CRC1) should be the same as calculated one. The same for BACKUP area.

# **Update functionality**

There are two Updaters in Bootloader program code:

* UartUpdater
* LiveUpdater

**UartUpdater** uses PIC32.UART3 of MCU. BBaudrate is 115200 (8n1). This updater started automatically after power on reset(POR) if following conditions meet:

* APPLICATION area is empty
* Two byte sequence issued to DebugUart(PIC32.UART3) and DebugPin(RB2) shorted to GND

Start sequence is 0xAA,0x55 it should be sent immediately within 300 ms after MCU start (POR).

**LiveUpdater** used to copy application code from BACKUP to APPLICATION if following conditions meet after POR:

* BACKUP version is higher than APPLICATION version
* BACKUP and APPLICATION has different CRC and DebugPin (RB2) is shorted to GND.

In all cases LiveUpdater checks if appropriate CRC in config is the same as calculated CRC. It means that Updater will avoid copying code from BACKUP if CRC is corrupted.

After POR of MCU bootloader will check CRC of CONFIGURATION area. If CRC is corrupted CONFIGURATION will be replaced by default and BACKUP code erased.

# **Bootloader first start**

Starting first application:

1. Flash bootloader code to MCU using PICKIT3 or 4.
2. On power DebugLed will blink (because APPLICATION area is empty)
3. Flash first APPLICATION with PC software.
   1. Press connect button
   2. Load application HEX file
   3. Press Erase and Verify

Flash of new application using DebugUart:

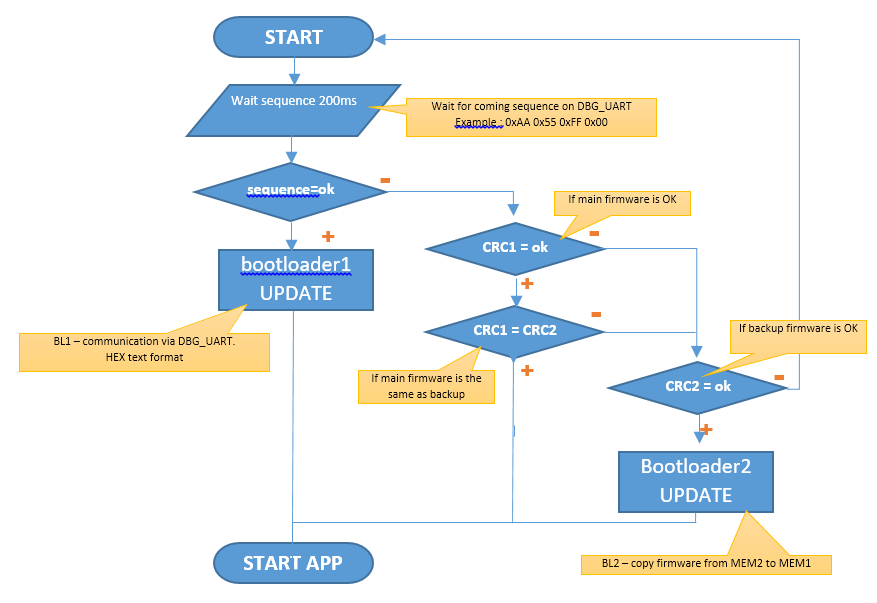
1. Short DebugPin to GND.
2. Restart MCU
3. Send Bootloader start sequence 0xAA,0x55 within 300ms to DebugUart.
4. Use PC software.

4.1. Press connect button

4.2. Load application HEX file

4.3. Press Erase and Verify

# **Bootloader logic**



# **Application configuration**

Linker file:

kseg1\_boot\_mem : ORIGIN = 0x9D000000, LENGTH = 0x480

sw\_version : ORIGIN = 0x9D000490, LENGTH = 0x4

some \*.c file should contain:

**volatile const uint32\_t \_\_attribute\_\_((section(".sw\_version"))) VERSION =** \_VERSION\_;

if you use template project all of above is included.

This application can start from program flash of MCU. And project doesn’t require bootloader. The same project can be loaded to MCU without changing linkerscript.

If bootloader not exist in flash memory it will not start.

# **Crc32 for application verification**

**const uint32\_t flashmem\_crc32\_tab[] = {**

**0x00000000, 0x77073096, 0xee0e612c, 0x990951ba, 0x076dc419, 0x706af48f,**

**0xe963a535, 0x9e6495a3, 0x0edb8832, 0x79dcb8a4, 0xe0d5e91e, 0x97d2d988,**

**0x09b64c2b, 0x7eb17cbd, 0xe7b82d07, 0x90bf1d91, 0x1db71064, 0x6ab020f2,**

**0xf3b97148, 0x84be41de, 0x1adad47d, 0x6ddde4eb, 0xf4d4b551, 0x83d385c7,**

**0x136c9856, 0x646ba8c0, 0xfd62f97a, 0x8a65c9ec, 0x14015c4f, 0x63066cd9,**

**0xfa0f3d63, 0x8d080df5, 0x3b6e20c8, 0x4c69105e, 0xd56041e4, 0xa2677172,**

**0x3c03e4d1, 0x4b04d447, 0xd20d85fd, 0xa50ab56b, 0x35b5a8fa, 0x42b2986c,**

**0xdbbbc9d6, 0xacbcf940, 0x32d86ce3, 0x45df5c75, 0xdcd60dcf, 0xabd13d59,**

**0x26d930ac, 0x51de003a, 0xc8d75180, 0xbfd06116, 0x21b4f4b5, 0x56b3c423,**

**0xcfba9599, 0xb8bda50f, 0x2802b89e, 0x5f058808, 0xc60cd9b2, 0xb10be924,**

**0x2f6f7c87, 0x58684c11, 0xc1611dab, 0xb6662d3d, 0x76dc4190, 0x01db7106,**

**0x98d220bc, 0xefd5102a, 0x71b18589, 0x06b6b51f, 0x9fbfe4a5, 0xe8b8d433,**

**0x7807c9a2, 0x0f00f934, 0x9609a88e, 0xe10e9818, 0x7f6a0dbb, 0x086d3d2d,**

**0x91646c97, 0xe6635c01, 0x6b6b51f4, 0x1c6c6162, 0x856530d8, 0xf262004e,**

**0x6c0695ed, 0x1b01a57b, 0x8208f4c1, 0xf50fc457, 0x65b0d9c6, 0x12b7e950,**

**0x8bbeb8ea, 0xfcb9887c, 0x62dd1ddf, 0x15da2d49, 0x8cd37cf3, 0xfbd44c65,**

**0x4db26158, 0x3ab551ce, 0xa3bc0074, 0xd4bb30e2, 0x4adfa541, 0x3dd895d7,**

**0xa4d1c46d, 0xd3d6f4fb, 0x4369e96a, 0x346ed9fc, 0xad678846, 0xda60b8d0,**

**0x44042d73, 0x33031de5, 0xaa0a4c5f, 0xdd0d7cc9, 0x5005713c, 0x270241aa,**

**0xbe0b1010, 0xc90c2086, 0x5768b525, 0x206f85b3, 0xb966d409, 0xce61e49f,**

**0x5edef90e, 0x29d9c998, 0xb0d09822, 0xc7d7a8b4, 0x59b33d17, 0x2eb40d81,**

**0xb7bd5c3b, 0xc0ba6cad, 0xedb88320, 0x9abfb3b6, 0x03b6e20c, 0x74b1d29a,**

**0xead54739, 0x9dd277af, 0x04db2615, 0x73dc1683, 0xe3630b12, 0x94643b84,**

**0x0d6d6a3e, 0x7a6a5aa8, 0xe40ecf0b, 0x9309ff9d, 0x0a00ae27, 0x7d079eb1,**

**0xf00f9344, 0x8708a3d2, 0x1e01f268, 0x6906c2fe, 0xf762575d, 0x806567cb,**

**0x196c3671, 0x6e6b06e7, 0xfed41b76, 0x89d32be0, 0x10da7a5a, 0x67dd4acc,**

**0xf9b9df6f, 0x8ebeeff9, 0x17b7be43, 0x60b08ed5, 0xd6d6a3e8, 0xa1d1937e,**

**0x38d8c2c4, 0x4fdff252, 0xd1bb67f1, 0xa6bc5767, 0x3fb506dd, 0x48b2364b,**

**0xd80d2bda, 0xaf0a1b4c, 0x36034af6, 0x41047a60, 0xdf60efc3, 0xa867df55,**

**0x316e8eef, 0x4669be79, 0xcb61b38c, 0xbc66831a, 0x256fd2a0, 0x5268e236,**

**0xcc0c7795, 0xbb0b4703, 0x220216b9, 0x5505262f, 0xc5ba3bbe, 0xb2bd0b28,**

**0x2bb45a92, 0x5cb36a04, 0xc2d7ffa7, 0xb5d0cf31, 0x2cd99e8b, 0x5bdeae1d,**

**0x9b64c2b0, 0xec63f226, 0x756aa39c, 0x026d930a, 0x9c0906a9, 0xeb0e363f,**

**0x72076785, 0x05005713, 0x95bf4a82, 0xe2b87a14, 0x7bb12bae, 0x0cb61b38,**

**0x92d28e9b, 0xe5d5be0d, 0x7cdcefb7, 0x0bdbdf21, 0x86d3d2d4, 0xf1d4e242,**

**0x68ddb3f8, 0x1fda836e, 0x81be16cd, 0xf6b9265b, 0x6fb077e1, 0x18b74777,**

**0x88085ae6, 0xff0f6a70, 0x66063bca, 0x11010b5c, 0x8f659eff, 0xf862ae69,**

**0x616bffd3, 0x166ccf45, 0xa00ae278, 0xd70dd2ee, 0x4e048354, 0x3903b3c2,**

**0xa7672661, 0xd06016f7, 0x4969474d, 0x3e6e77db, 0xaed16a4a, 0xd9d65adc,**

**0x40df0b66, 0x37d83bf0, 0xa9bcae53, 0xdebb9ec5, 0x47b2cf7f, 0x30b5ffe9,**

**0xbdbdf21c, 0xcabac28a, 0x53b39330, 0x24b4a3a6, 0xbad03605, 0xcdd70693,**

**0x54de5729, 0x23d967bf, 0xb3667a2e, 0xc4614ab8, 0x5d681b02, 0x2a6f2b94,**

**0xb40bbe37, 0xc30c8ea1, 0x5a05df1b, 0x2d02ef8d**

**};**

**uint32\_t calculateFlashMemCrc(const uint8\_t \*buf, size\_t size) {**

**const uint8\_t \*p = buf;**

**uint32\_t crc;**

**crc = ~0U;**

**while (size--)**

**crc = flashmem\_crc32\_tab[(crc ^ \*p++) & 0xFF] ^ (crc >> 8);**

**return crc ^ ~0U;**

**}**