GPM32F M0 系列 Flash Memory 操作

USER MANUAL

V1.0 – Mar. 30, 2023

Generalplus Technology Inc. reserves the right to change this documentation without prior notice. Information provided by Generalplus Technology Inc. is believed to be accurate and reliable. However, no responsibility is assumed by Generalplus Technology Inc. for errors, omissions or any loss of profit resulting from the use of information contained in this documentation. In addition, Generalplus products are not authorized for use as critical components in life support devices/systems or aviation devices/systems, where a malfunction or failure of the product may reasonably be expected to result in significant injury to the user, without the express written approval of Generalplus.

Generalplus Technology
No.19, Industry E. Rd. IV, Hsinchu Science Park, Hsinchu City, 30077, Taiwan, R.O.C.
Tel: 886-3-666-2118
Fax: 886-3-666-2117
www.generalplus.com

REVISION HISTORY

Revision	Date	By	Remark
V1.0	2023.3.30	Henry Hsieh	

TABLE OF CONTENT

1. GPM32F M0 系列 Flash Memory 操作 5

1.1. 概述 5

1.2. Flash Memory 操作注意事項 5

1.3. Flash Memory 操作流程說明 5

1.4. GPM32F6006U Flash Memory 操作範例程序 8

1. GPM32F M0 系列 Flash Memory 操作

1.1. 概述

在 MCU 應用中，有時需要利用 Flash memory 來存取參數，若操作不當有可能寫錯地址，甚至把正常程序改掉。本文旨在說明 GPM32F M0 系列 Flash Memory Erase/Write 操作流程，以及需要注意的事項。

1.2. Flash Memory 操作注意事項

使用此程序功能之前，有二個地方必須注意，如下所敘：

- 使用者先用示波器量測操作 Flash memory 所需時間，接著再量測 MCU VDD 從 LVD 準位掉到 LVR 準位時間，要大於操作 Flash memory 所需時間。
- 確認產品工作環境溫度。由於電容在高溫下的電容值比常溫下小，因此會縮短電容放電時間，導致 Flash memory 還沒完全寫完 MCU VDD 就觸到 LVR 準位，而來不及把全部資料寫入。

1.3. Flash Memory 操作流程說明

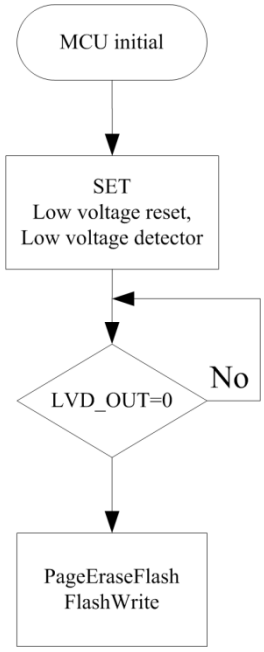


Figure 1.1 Flash Memory 操作流程圖

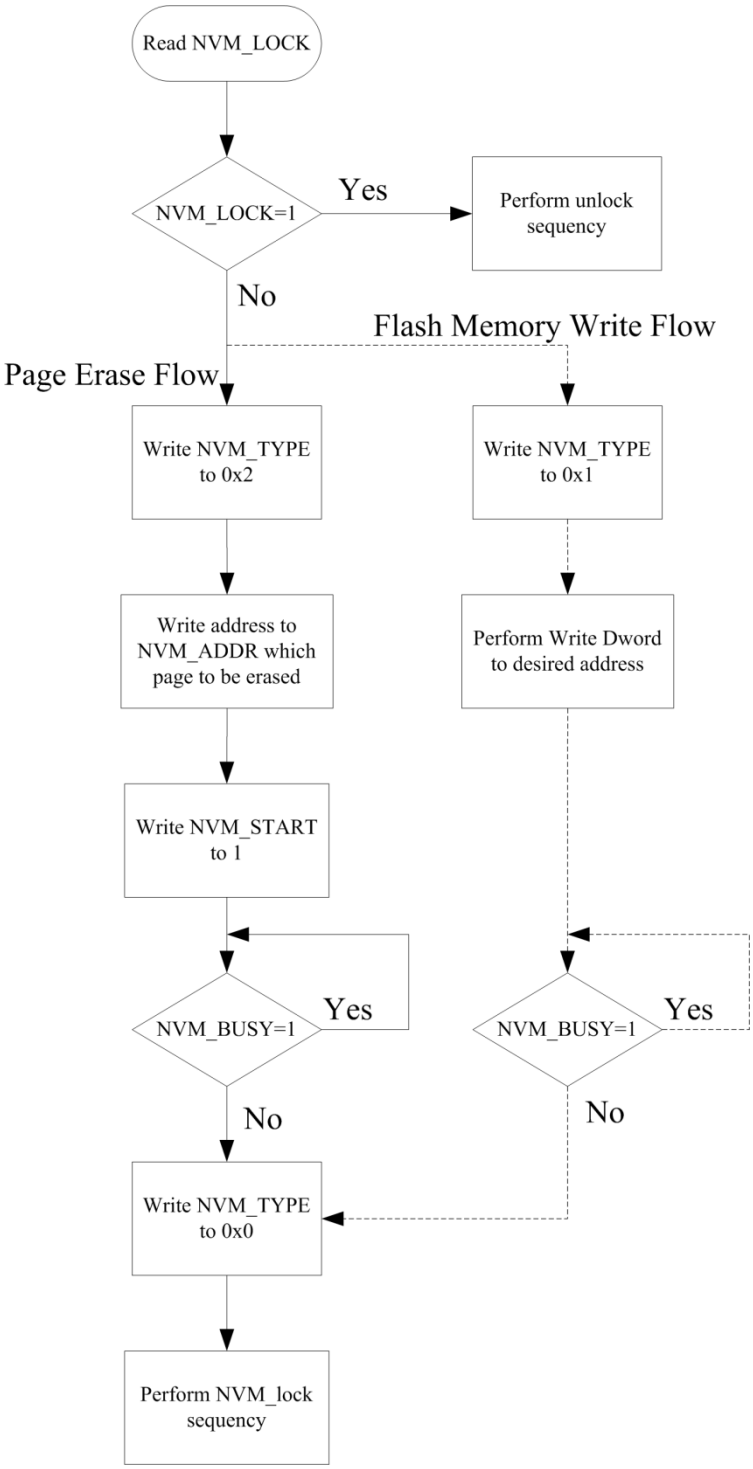


Figure 1.2 Page Erase, Flash Memory Write 操作流程圖

Flash Memory 操作流程說明

■ 步驟一：

設定合適的 LVD/LVR 準位。例如 MCU VDD 為 5V，可以設定 LVD 4.2V 觸發，LVR 2.3V 觸發。須注意 MCU VDD 從 LVD 準位掉到 LVR 準位時間，要大於操作 Flash Memory 所需時間。

以下圖為例，Ch1 為 Page Erase 時間，Ch2 為 Flash Memory Write 時間(256bytes)，Ch3 為 Flash Memory Read 時間(256bytes)，MCU VDD 從 LVD 準位掉到 LVR 準位時間需要大於 3.9ms。

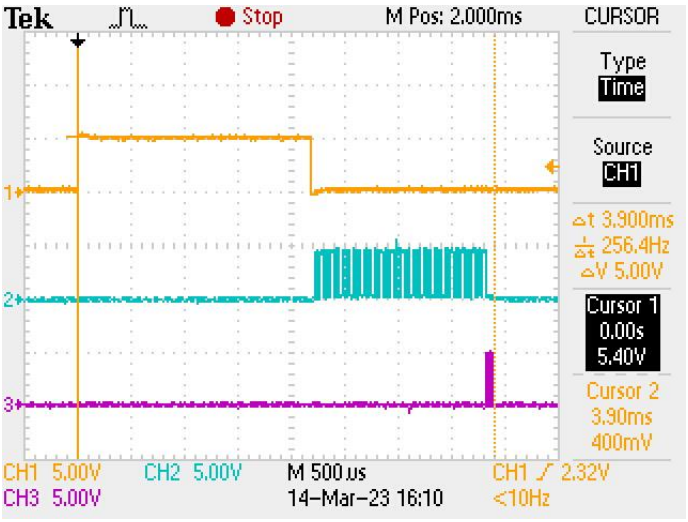


Figure 1.3 操作 Flash Memory 所需時間

■ 步驟二：

操作 Flash Memory 之前先判別 LVD_OUT 是否動作，發生 LVD 就不操作 Flash Memory。

■ 步驟三：

操作 Flash Memory Write 之前需要先做 Page Erase 再做 Flash Memory Write，接著，使用者可以把 Flash memory 讀出來確認與寫入的資料是否一致。操作 Flash Memory 相關暫存器需要先執行兩個程序 “(NVM->FSHKEY =0xabcd5678);(NVM->FSHKEY =0x1234FEDC);”，使其解鎖後才能有效操作 Flash Memory 相關暫存器。

1.4. GPM32F6006U Flash Memory 操作範例程序

- Main Code

```
/*;====================================================================
Main code
The example code erase flash 0x01007800~0x01007BFF, Write and verify data between
0x01007800~0x010078FF
;====================================================================*/
int main(void)
{
  uint32_t uiFLASH_ADDRESS, uiFLASH_ADDRESS_START, uiFLASH_ADDRESS_END,
  uiFLASH_DATA;
  unsigned int u32Temp;

  GPIO0->CFG_b.P00 = GPIO_b_PUSHPULL;
  GPIO0->CFG_b.P01 = GPIO_b_PUSHPULL;
  GPIO0->CFG_b.P02 = GPIO_b_PUSHPULL;
  GPIO0->CFG_b.P03 = GPIO_b_PUSHPULL;

  if(ACU->STS_b.LVD_OUT)
  {//LVD occur don't program flash
    GPIO0->BTGL_b.P00=1;
  }
  else
  {//User have to take care program flash time is between LVD to LVR.
     if(NVM->CTRL & NVM_LOCK)
     {
//----------Flash Control Register Unlock----------//
        NVM_FlashControlRegisterUnlock;
     }
       //------    NVM Operation Type is PAGE_ERASE---------//
        NVN_OperationType(NVM_TYPE_PAGE_ERASE);

       //------- Set NVM Erase Address--------------------//
        NVM_EraseAddress(0x01007800); // Erase Flash 0x01007800~0x01007BFF

       //--------NVM Operation Start----------------------//
       GPIO0->ODR_b.P01=1;
        NVM_NVMOperationStart;

        //---------Wait NVM Operation Busy Flag------------//
        while(NVM_OperationBusyFlag);
        GPIO0->ODR_b.P01=0;

        //---------- NVM Operation Type is IDLE------------//
```

```
    NVN_OperationType(NVM_TYPE_IDLE);

    //Program address from 0x01007800 to 0x010078FF
    //Program data 0x98765432
    uiFLASH_ADDRESS_START =0x01007800;
    uiFLASH_ADDRESS_END =0x010078FF;
    uiFLASH_DATA =0x98765432;

    //------    NVM Operation Type is Program----------//
    NVN_OperationType(NVM_TYPE_PROGRAM);

    //Program certain data to a certain address
   for(uiFLASH_ADDRESS =uiFLASH_ADDRESS_START; uiFLASH_ADDRESS
                <uiFLASH_ADDRESS_END; uiFLASH_ADDRESS +=0x00000004)
    {
        //Program data
        *(uint32_t *)uiFLASH_ADDRESS =uiFLASH_DATA;

        //Wait until program is completed
        while(NVM_OperationBusyFlag);
       GPIO0->BTGL_b.P02=1;
    }

   //Read certain data from a certain address
   for(uiFLASH_ADDRESS =uiFLASH_ADDRESS_START; uiFLASH_ADDRESS
                <uiFLASH_ADDRESS_END; uiFLASH_ADDRESS +=0x00000004)
    {
        //Read data form code data
       u32Temp = *((unsigned int *)uiFLASH_ADDRESS);
       if(u32Temp != uiFLASH_DATA)
          while(1); // verify flash error
        GPIO0->BTGL_b.P03=1;
    }

    //---------- NVM Operation Type is IDLE----------//
    NVN_OperationType(NVM_TYPE_IDLE);

   //----------Flash Control Register lock----------//
    NVM_FlashControlRegisterlock;
}
  while(1);
```

```
}
```

- Marco Define

```
/*****************************************************************************
 * @brief
 * Flash Control Register Unlock Key
 * @param
 *    NVM : The pointer of the specified NVM module.
 * @return
 *    None
 *****************************************************************************/
 #define NVM_FlashControlRegisterUnlock    {(NVM->FSHKEY =0xabcd5678);(NVM->FSHKEY
=0x1234FEDC);}

/*****************************************************************************
 * @brief
 * Flash Control Register lock Key
 * @param
 *    NVM : The pointer of the specified NVM module.
 * @return
 *    None
 *****************************************************************************/
 #define NVM_FlashControlRegisterlock    (NVM->FSHKEY =0x00000000)

/*****************************************************************************
 * @brief
 * NVM Operation Start
 * @param
 *    NVM : The pointer of the specified NVM module.
 * @return
 *    None
 *****************************************************************************/
 #define NVM_NVMOperationStart (NVM->CTRL |=NVM_START)

/*****************************************************************************
 * @brief
 *    NVM Operation Type
 * @param
 *    NVM : The pointer of the specified NVM module.
 *    u32OperationType:NVM Operation Type
 *                      \ref    NVM_TYPE_IDLE
 *                      \ref    NVM_TYPE_PROGRAM
 *                      \ref    NVM_TYPE_PAGE_ERASE
 *                      \ref    NVM_TYPE_CHIP_ERASE
```

```
 * @return
 *    None
 *****************************************************************************/
  #define NVN_OperationType(u32OperationType)    (NVM->CTRL = ((NVM->CTRL &~
NVM_CTRL_TYPE_Msk)|(u32OperationType)))

/*****************************************************************************
 * @brief
 *    NVM_CTRL are locked
 * @param
 *    NVM : The pointer of the specified NVM module.
 * @return
 *    None
 *****************************************************************************/
 #define NVM_CTRLock (NVM->CTRL |=NVM_LOCK)

/*****************************************************************************
 * @brief
 *    NVM_CTRL are unlocked
 * @param
 *    NVM : The pointer of the specified NVM module.
 * @return
 *    None
 *****************************************************************************/
 #define NVM_CTRLUnlock (NVM->CTRL &=~NVM_LOCK)

/*****************************************************************************
 * @brief
 *    NVM Erase Address Register
 * @param
 *    NVM : The pointer of the specified NVM module.
 *    u32EraseAddress:Data Memory Start Address
 * @return
 *    None
 *****************************************************************************/
 #define NVM_EraseAddress(u32EraseAddress)    (NVM->ADDR =u32EraseAddress )

/*****************************************************************************
 * @brief
 *    NVM Operation Busy Flag
 * @param
 *    NVM : The pointer of the specified NVM module.
 *    u32FlashAddress: Flash Address for NVM Erase
 * @return
```

```
*    None
*******************************************************************************/
#define NVM_OperationBusyFlag (NVM->STS & NVM_STS_BUSY ? 1:0)
```