Vol.11, Issue No 3, Sep 2021

# Flash Memory Design

S Prakasha, Sindhu R, Firdosh Parveen S

Asst. Prof & HOD, Asst. Prof, Asst. Prof

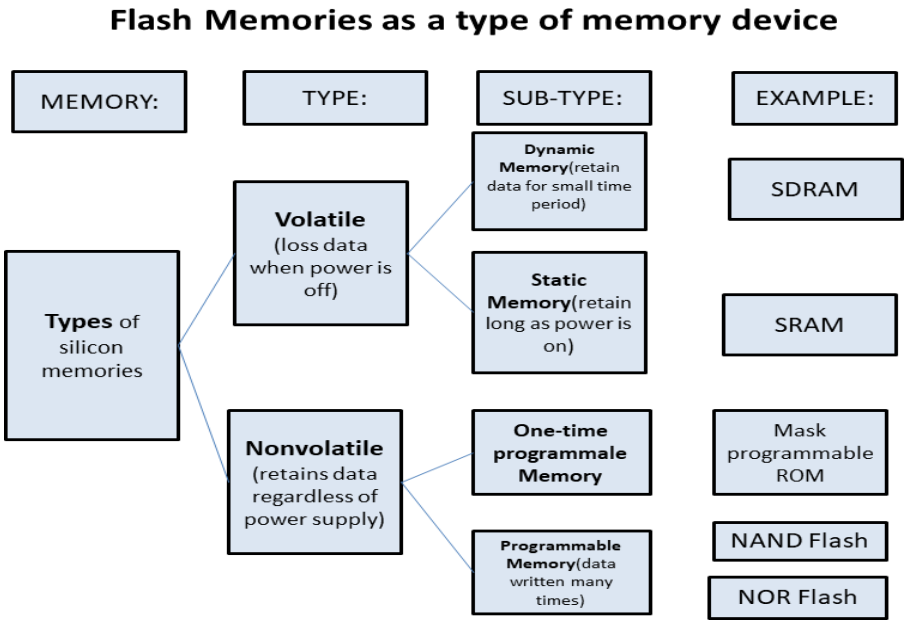
prakashshanbog@gmail.com, rethisindhoo@gmail.com, firdoseks@gmail.com

Department of EEE, Proudhadevaraya Institute of Technology, Abheraj Baldota Rd, Indiranagar, Hosapete, Karnataka-583225

## 1. Introduction

In this design, a basic design for a named flash Memory is presented based on the existing and manufactured ICs. The Micron company has produced a variety of memories, which I will mention in this article one of the ICs. At first the specification of Flash Memories will be introduced and then structure of a nominal Flash will be,

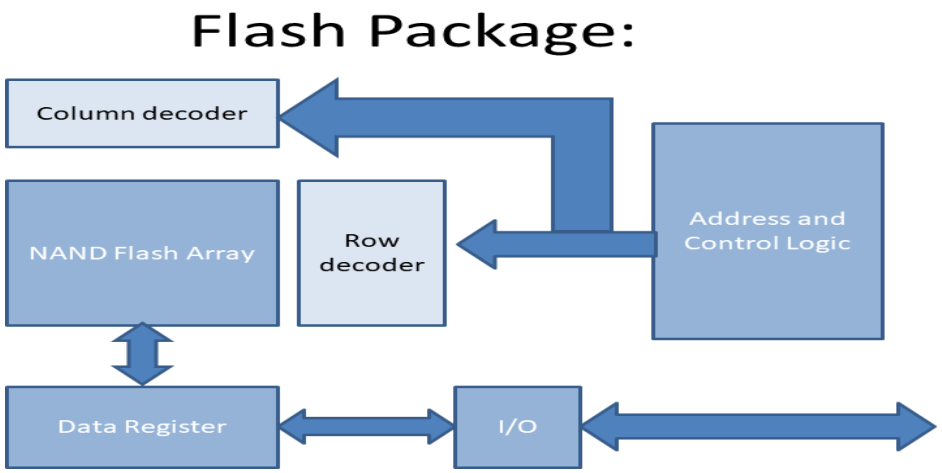
**• Different Kinds of Memories**




In this chart, you can see a flash memory is a nonvolatile memory that is programmable means it can be written many times in 2 kind categories: NAND and NOR architecture [1-4].

**• Architecture**

Below, the architecture of a memory revelation is depicted. As you can see, in addition to memory design, we need a controller and the right input/output, as well as a register.




**• Hardware**

For hardware design, the following ICs are mentioned, which are related to the products of several companies.

In below, it will be introduced a Flash memory from Micron Chip Company. This company has produced flash memory IC toward 128GB .

**• QLED NAND FLASH 16GB: Micron Chip Manufacture**

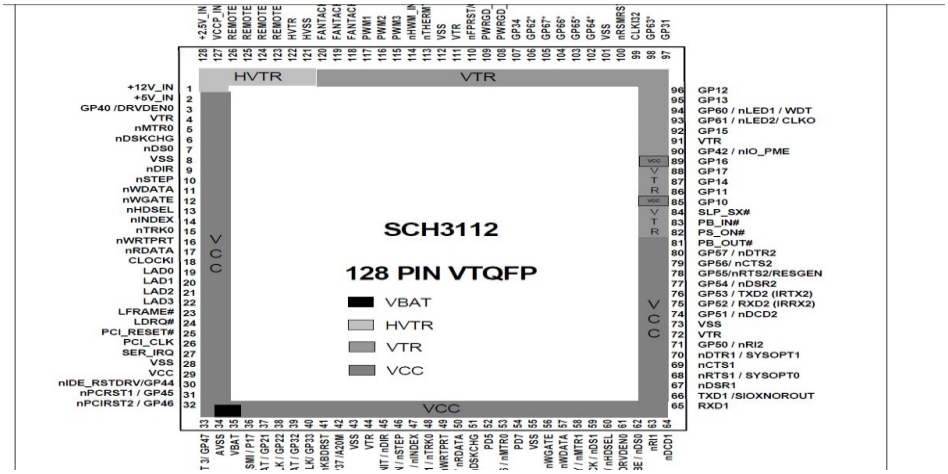
Details:

| | |
|---|---|
| **I/O Voltage** | 1.2V |
| **Operating Temp** | 0C to +70C |
| **Component Density** | 16Tb |

**• Part Spec For QLC NAND Flash Memory PART MT29F16T08GWLDHD8-T:D**

| | |
|---|---|
| Component Density | 16Tb |
| MT/s | NSR |
| Operating Temp | OC to +70C |
| Technology | MAS FLASH |
| Family | NAND FLASH |
| Part Status Code | Contact Factory |
| PLP | NO |
| Pin Count | 154-ball |
| Package | LFBGA |
| I/O Voltage | 1.2 v |
| Package Type | GREEN |
| Number of Components | 16 |
| Bus Width | X8 |
| Component Config | 2T*8 |
| Speed | 800MHz |
| Capacity | 16Tb |

**• Microchip Manufacture**
**SCH3112/SCH3114/SCH3116: I/O CONTROLLER**
**LPC IO with 8042 KBC, Reset Generation, HWM and Multiple Serial Ports**




Programming this I/O, can be designed for work with this memory.

**• Software**
Below, a

sample for Software Driver has com.

```
//Software Driver

//SST39VF6401B/SST39VF6402B
//64 Mbit Multi-Purpose Flash (MPF+)
//
//June 2005
//
//ABOUT THE SOFTWARE
//
//This application note provides a software driver example for 39VF6401B/39VF6402B
//64 Mbit Multi-Purpose Flash (MPF+) that can be used in any microprocessor based
//system.
//
//The SST39VF6401B supports bottom boot block protection, and the SST39VF6402B
//supports top boot block protection. The boot block memory area is protected when
//WP# is low and unprotected when WP# is high.
//
//Software driver example routines provided in this document utilize high-level
//"C" programming language for broad platform support. In many cases, software
//driver routines can be inserted "as is" into the main body of code being
//developed by the system software developers. Extensive comments are included
//in each routine to describe the function of each routine. The software driver
//routines in "C" can be used with many microprocessors and microcontrollers.
//
//ABOUT THE SST39VF6401B/SST39VF6402B
//
//Companion product datasheet for 39VF6401/39VF6402 should be reviewed in
//conjunction with this application note for a complete understanding of the device.
//
//The C code in this document contains the following routines, which are listed
//in this order:
//
//Name                      Function
//-------------------------------------------------------------------------------------
```

```
//Check_SST_39VF640XB          Check manufacturer and device ID
//CFI_Query                   CFI Query Entry/Exit command sequence
//SecID_Query                    SecID Query Entry/Exit command sequence
//Erase_One_Sector            Erase a sector of 2048 words
//Erase_One_Block             Erase a block of 32K words
//Erase_Entire_Chip           Erase the contents of the entire chip
//Program_One_Word            Alter data in one word
//Program_One_Sector          Alter data in 2048-word sector
//Program_One_Block           Alter data in 32K-word block
//SecID_Lock_StatusCheck the Lock Status of Security ID segment
//User_SecID_Word_Program Write data into User Security ID Segment
//User_SecID_Lock_Out            Lock out the User Security ID Segment
//Erase_Suspend                  Suspend Sector/Block Erase operation
//Erase_Resume                   Resume Sector/Block Erase operation
//Check_Toggle_Ready          End of internal program or erase detection using
//                            Toggle bit
//Check_Data_Polling          End of internal program or erase detection using
//                            Data# polling
//
//"C" LANGUAGE DRIVERS


/************************************************************************/
/* Copyright Silicon Storage Technology, Inc. (SST), 1994-2005                  */
/* Example "C" language Driver of 39VF640X 64 Mbit MPF+ Device                     */
/* Verified by Yonglin, Silicon Storage Technology, Inc.                                */
/*                                                                              */
/* Revision 1.0, June 17, 2005                                                  */
/*                                                                              */
/* This file requires these external "timing"  routines:                        */
/*                                                                              */
/*       1.)  Delay_10_Micro_Seconds                                            */
/*       2.)  Delay_20_Micro_Seconds                                            */
/*       3.)  Delay_150_Nano_Seconds                                            */
/*       4.)  Delay_25_Milli_Seconds                                            */
/*       5.)  Delay_50_Milli_Seconds                                            */
/************************************************************************/ #define

FALSE                           0

#define TRUE                    1

#define SECTOR_SIZE             2048     // Must be 2048 words for 39VF640XB
#define BLOCK_SIZE              32768    // Must be 32K words for 39VF640XB

#define SST_ID                  0x00BF // SST Manufacturer's ID code
#define SST_39VF6401B            0x236D // SST39VF6401B device code
#define SST_39VF6402B            0x236C  // SST39VF6402B device code

typedef unsigned char           BYTE;       // BYTE is 8-bit in length
//typedef unsigned int             WORD;       // WORD is 16-bit in length typedef
unsigned short                   WORD;       // WORD is 16-bit in length, ARM
//typedef unsigned long int    Uint32; // Uint32  is  32-bit  in  length typedef
unsigned int                 Uint32; // Uint32 is 32-bit in length, ARM

//Uint32 system_base = 0x4000000; // 4GByte System Memory Address.#define

system_base  0x4000000
          // This sample code uses 0x4000000 as the system_base address in ARM system.
          // The user should modify this address accordingly.

//#define sysAddress(offset) ((volatile WORD *)(system_base + offset))
#define sysAddress(offset) ((volatile WORD *)(system_base + (((WORD)offset)<<1))) //For ARM CPU


#define MAX_TIMEOUT0x07FFFFFF       // A ceiling constant used by Check_Toggle_
          // Ready() and Check_Data_Polling().
          // The user should modify this constant accordingly.

// ----------------------------------------------------------------------------------------
//                           EXTERNAL ROUTINES
// ----------------------------------------------------------------------------------------
extern void  Delay_10_Micro_Seconds();
extern  void  Delay_20_Micro_Seconds();  extern
void   Delay_150_Nano_Seconds();   extern   void
Delay_25_Milli_Seconds();      extern      void
Delay_50_Milli_Seconds();
```

```
// ------------------------------------------------------------------------------------------

int Check_SST_39VF640XB(void);void
CFI_Query(WORD*);
void SecID_Query(WORD*, WORD*);int
Erase_One_Sector(Uint32); int
Erase_One_Block (Uint32); int
Erase_Entire_Chip(void);
int Program_One_Word (WORD*, Uint32); int
Program_One_Sector (WORD*, Uint32);
int Program_One_Block (WORD *Src, Uint32 Dst);int
SecID_Lock_Status(void);
int User_SecID_Word_Program (WORD*, WORD*, int);void
User_SecID_Lock_Out (void);
void Erase_Suspend (void);
void Erase_Resume (void);
int Check_Toggle_Ready (Uint32);
int Check_Data_Polling (Uint32, WORD);


/*********************************************************************/
/* PROCEDURE:     Check_SST_39VF640XB                                 */
/*                                                                    */
/* This procedure decides whether a physical hardware device has a    */
/* SST39VF640XB 64 Mbit MPF+ Device installed or not.                      */
/*                                                                    */
/* Input:                                                             */
/*            None                                                    */
/*                                                                    */
/* Output:                                                            */
/*            return TRUE:  indicates a SST39VF640XB                   */

/*            return FALSE: indicates not a SST39VF640XB               */
/*********************************************************************/ int
Check_SST_39VF640XB(void)
{
     WORD SST_id1;
     WORD SST_id2;
     int  ReturnStatus;


     //  Issue the Software Product ID code to 39VF640XB
     *sysAddress(0x555) = 0x00AA;  // write data 0x00AA to device addr 0x555

     *sysAddress(0x2AA) = 0x0055;  // write data 0x0055 to device addr 0x2AA

     *sysAddress(0x555) = 0x0090;  // write data 0x0090 to device addr 0x555

     Delay_150_Nano_Seconds();       // Tida Max 150ns for 39VF640XB

     // Read the product ID from 39VF640XB
     SST_id1  = *sysAddress(0x0000);        // get first ID byte
     SST_id2  = *sysAddress(0x0001);        // get second ID byte

     // ------------------------------------------------------------------------------------
     // Determine whether there is a SST 39VF6401B installed or not
     // use the following code:

     //if ((SST_id1 == SST_ID) && (SST_id2 == SST_39VF6401B))
     //         ReturnStatus = TRUE;
     //else
     //         ReturnStatus = FALSE;
     // ------------------------------------------------------------------------------------
     // Or determine whether there is a SST 39VF6402B installed or not
     // use the following code:

     if ((SST_id1 == SST_ID) && (SST_id2 == SST_39VF6402B))
               ReturnStatus = TRUE;
     else
               ReturnStatus = FALSE;
     // ------------------------------------------------------------------------------------

     // Issue the Software Product ID Exit code, thus returning the
     // 39VF640X to the normal operation.
     *sysAddress(0x555) = 0x00AA;  // write data 0x00AA to device addr 0x555
     *sysAddress(0x2AA) = 0x0055;  // write data 0x0055 to device addr 0x2AA
     *sysAddress(0x555) = 0x00F0;  // write data 0x00F0 to device addr 0x555
     Delay_150_Nano_Seconds();       // Tida Max 150ns for 39VF640XB
```

```
    return (ReturnStatus);
}
*********************************************************************/
/* PROCEDURE:     CFI_Query                                                    */
/*                                                                             */
/* This procedure should be used to query for CFI information                  */
/*                                                                             */
/* Input:                                                                      */
/*            Src     Source address to store CFI_Query data string                  */
/*                                                                             */
/* Output:                                                                     */
/*            None                                                             */
/*********************************************************************/ void
CFI_Query(WORD *Src)
{
    WORD index;
    //  Issue the CFI Query entry code to 39VF640X
    *sysAddress(0x555) = 0x00AA;  // write data 0x00AA to device addr 0x555
    *sysAddress(0x2AA) = 0x0055;  // write data 0x0055 to device addr 0x2AA
    *sysAddress(0x555) = 0x0098;  // write data 0x0098 to device addr 0x555
    Delay_150_Nano_Seconds();         // insert delay time = Tida
    // ---------------------------------------------------------------------------
    //  Perform all CFI operations here:
    //  CFI_Query_address is from 0010H--0034H

    for ( index = 0x0010; index <= 0x0034; index++)
    {
           *Src = *sysAddress(index);
         ++Src;
          // CFI query data is stored in user-defined memory space.
    }
    // ---------------------------------------------------------------------------

         // Issue the CFI Exit code thus returning the 39VF640X
         // to the read operating mode

    *sysAddress(0x555) = 0x00AA;  // write data 0x00AA to device addr 0x555
    *sysAddress(0x2AA) = 0x0055;  // write data 0x0055 to device addr 0x2AA
    *sysAddress(0x555) = 0x00F0;  // write data 0x00F0 to device addr 0x555
    Delay_150_Nano_Seconds();         // insert delay time = Tida
}


/*********************************************************************/
/* PROCEDURE:     SecID_Query                                                  */
/*                                                                             */
/* This procedure should be used to query for Security ID information.  */
/*                                                                             */
/* Input:                                                                      */
/*            SST_SecID          Source address to store SST SecID string          */
/*            User_SecID         Source address to store User SecID string         */
/*                                                                             */
/* Output:                                                                     */
/*            None                                                             */
/*********************************************************************/ void
SecID_Query(WORD *SST_SecID, WORD *User_SecID)
{
    WORD index;
         //  Issue the SecID Entry code to 39VF640X
    *sysAddress(0x555) = 0x00AA;  // write data 0x00AA to device addr 0x555
    *sysAddress(0x2AA) = 0x0055;  // write data 0x0055 to device addr 0x2AA
    *sysAddress(0x555) = 0x0088;  // write data 0x0088 to device addr 0x555
    Delay_150_Nano_Seconds();          // insert delay time = Tida

          //  Perform all Security ID operations here:
          //         SST programmed segment is from address 000000H--000007H,
          //         User programmed segment is from address 000010H--000017H.

    for (index = 0x0000; index <= 0x0007; index++)
    {
           *SST_SecID = *sysAddress(index);
         ++SST_SecID;
         *User_SecID = *sysAddress(index+0x0010);
         ++User_SecID;
          // Security query data is stored in user-defined memory space.
    }
```

```
        // Issue the Sec ID Exit code thus returning the 39VF640X
        // to the read operating mode
    *sysAddress(0x555) = 0x00AA;  // write data 0x00AA to device addr 0x555
    *sysAddress(0x2AA) = 0x0055;  // write data 0x0055 to device addr 0x2AA
    *sysAddress(0x555) = 0x00F0;  // write data 0x00F0 to device addr 0x555
    Delay_150_Nano_Seconds();          // insert delay time = Tida
}
/*************************************************************************/
/* PROCEDURE:     Erase_One_Sector                                       */
/*                                                                       */
/* This procedure can be used to erase a total of 2048 words.            */
/*                                                                       */
/* Input:                                                                */
/*       Dst      DESTINATION address where the erase operation starts   */

/*                                                                       */
/* Output:                                                               */
/*            return TRUE:  indicates success in sector-erase            */
/*            return FALSE: indicates failure in sector-erase            */
/*************************************************************************/ int
Erase_One_Sector(Uint32 Dst)
{
           Uint32 DestBuf = Dst;int
   ReturnStatus;

           //  Issue the Sector Erase command to 39VF640X
           *sysAddress(0x555) = 0x00AA;          // write data 0x00AA to device addr 0x555
           *sysAddress(0x2AA) = 0x0055;          // write data 0x0055 to device addr 0x2AA
           *sysAddress(0x555) = 0x0080;          // write data 0x0080 to device addr 0x555
           *sysAddress(0x555) = 0x00AA;          // write data 0x00AA to device addr 0x555
           *sysAddress(0x2AA) = 0x0055;          // write data 0x0055 to device addr 0x2AA
           *sysAddress(DestBuf) = 0x0050;         // write data 0x0050 to device sector addr

   ReturnStatus = Check_Toggle_Ready(DestBuf);  // wait for TOGGLE bit ready

   return ReturnStatus;
}


/*************************************************************************/
/* PROCEDURE:     Erase_One_Block                                        */
/*                                                                       */
/* This procedure can be used to erase a total of 32K words.             */
/*                                                                       */
/* Input:                                                                */
/*       Dst      DESTINATION address where the erase operation starts   */
/*                                                                       */
/* Output:                                                               */
/*            return TRUE:  indicates success in block-erase             */
/*            return FALSE: indicates failure in block-erase             */
/*************************************************************************/ int
Erase_One_Block (Uint32 Dst)
{
   Uint32 DestBuf = Dst;int
   ReturnStatus;

    //  Issue the Block Erase command to 39VF640X
          *sysAddress(0x555) = 0x00AA;          // write data 0x00AA to device addr 0x555
          *sysAddress(0x2AA) = 0x0055;          // write data 0x0055 to device addr 0x2AA
          *sysAddress(0x555) = 0x0080;          // write data 0x0080 to device addr 0x555
          *sysAddress(0x555) = 0x00AA;          // write data 0x00AA to device addr 0x555
          *sysAddress(0x2AA) = 0x0055;          // write data 0x0055 to device addr 0x2AA
          *sysAddress(DestBuf) = 0x0030;         // write data 0x0030 to device sector addr

   ReturnStatus = Check_Toggle_Ready(DestBuf);    // wait for TOGGLE bit ready

   return ReturnStatus;
}


/*************************************************************************/
/* PROCEDURE:     Erase_Entire_Chip                                      */
/*                                                                       */
/* This procedure can be used to erase the entire chip.                  */
/*                                                                       */
/* Input:                                                                */
/*       NONE                                                            */
/*                                                                       */
```

```
/* Output:                                                                  */
/*       NONE                                                               */
/**************************************************************************/ int
Erase_Entire_Chip(void)
{
   //  Issue the Chip Erase command to 39VF640X
          *sysAddress(0x555) = 0x00AA;          // write data 0x00AA to device addr 0x555
          *sysAddress(0x2AA) = 0x0055;          // write data 0x0055 to device addr 0x2AA

          *sysAddress(0x555) = 0x0080;          // write data 0x0080 to device addr 0x555
          *sysAddress(0x555) = 0x00AA;          // write data 0x00AA to device addr 0x555
          *sysAddress(0x2AA) = 0x0055;          // write data 0x0055 to device addr 0x2AA
          *sysAddress(0x555) = 0x0010;          // write data 0x0010 to device addr 0x555

   //Delay_50_Milli_Seconds();             // Delay Tsce timeif
   (Check_Data_Polling (0,0xFFFF))
          return TRUE;
   else
    return FALSE;
}
/**************************************************************************/
/* PROCEDURE:     Program_One_Word                                        */
/*                                                                        */
/* This procedure can be used to program ONE word of data to the          */
/* 39VF640X.                                                              */
/*                                                                        */
/* NOTE:  It is necessary to first erase the sector containing the        */
/*         word to be programmed.                                    */
/*                                                                        */
/* Input:                                                                 */
/*             SrcWord The WORD which will be written to the 39VF640XB    */
/*             Dst      DESTINATION address which will be written with the */
/*                      data passed in from Src                           */
/*                                                                        */
/* Output:                                                                */
/*           return TRUE:  indicates success in word-program              */
/*           return FALSE: indicates failure in word-program              */
/**************************************************************************/ int
Program_One_Word (WORD *SrcWord,            Uint32 Dst)
{
    Uint32 DestBuf = Dst;
    WORD *SourceBuf = SrcWord;int
    ReturnStatus;

    *sysAddress(0x555) = 0x00AA; // write data 0x00AA to device addr 0x555
    *sysAddress(0x2AA) = 0x0055; // write data 0x0055 to device addr 0x2AA
    *sysAddress(0x555) = 0x00A0; // write data 0x00A0 to device addr 0x555
    *sysAddress(DestBuf) = *SourceBuf; // transfer the WORD to destination

    ReturnStatus = Check_Toggle_Ready(DestBuf); // wait for TOGGLE bit ready

    return ReturnStatus;
}
/**************************************************************************/
/* PROCEDURE:     Program_One_Sector                                      */
/*                                                                        */
/* This procedure can be used to program a total of 2048 words of data  */
/* to the SST39VF640X.                                                    */
/*                                                                        */
/* NOTES: 1. It is necessary to first erase the sector before the  */
/*         programming.                                                        */
/*         2. This sample code assumes the destination address passed         */
/*         from the calling function is the starting address of the */
/*         sector.                                                            */
/*                                                                        */
/* Input:                                                                 */
/*             Src      SOURCE address containing the data which will be  */
/*                      written to the 39VF640XB                           */
/*             Dst      DESTINATION address which will be written with the */
/*                      data passed in from Src                           */
/*                                                                        */
/* Output:                                                                */
/*           return TRUE:  indicates success in sector-program            */
/*           return FALSE: indicates failure in sector-program            */
/**************************************************************************/ int
Program_One_Sector (WORD *Src, Uint32 Dst)
```

```
{

     WORD *SourceBuf;
     Uint32 DestBuf;
     int Index, ReturnStatus;

     SourceBuf = Src;
     DestBuf = Dst;
     ReturnStatus = Erase_One_Sector(DestBuf);         // erase the sector firstif
     (!ReturnStatus)
           return ReturnStatus;

     for (Index = 0; Index < SECTOR_SIZE; Index++)
     {     // transfer data from source to destination ReturnStatus =
           Program_One_Word( SourceBuf, DestBuf);
           ++DestBuf;
           ++SourceBuf;

          if  (!ReturnStatus) return
               ReturnStatus;
     }

     return ReturnStatus;
}
/*************************************************************************/
/* PROCEDURE:     Program_One_Block                                      */
/*                                                                       */
/* This procedure can be used to program a total of 32k words of data    */
/* to the SST39VF640XB.                                                   */
/*                                                                       */
/* NOTES: 1. It is necessary to first erase the block before the            */
/*          programming.                                                       */
/*          2. This sample code assumes the destination address passed         */
/*          from the calling function is the starting address of the */
/*          block.                                                             */
/*                                                                                    */
/* Input:                                                                */
/*              Src      SOURCE address containing the data which will be */
/*                       written to the 39VF640X                          */
/*              Dst      DESTINATION address which will be written with the */
/*                       data passed in from Src                          */
/*                                                                       */
/* Output:                                                               */
/*            return TRUE:  indicates success in block-program           */
/*            return FALSE: indicates failure in block-program           */
/*************************************************************************/ int
Program_One_Block (WORD *Src, Uint32 Dst)
{
     WORD *SourceBuf;
     Uint32 DestBuf;
     int Index, ReturnStatus;

     SourceBuf = Src;
     DestBuf = Dst;
     ReturnStatus = Erase_One_Block(DestBuf);          // erase the block firstif
     (!ReturnStatus)
           return ReturnStatus;

     for (Index = 0; Index < BLOCK_SIZE; Index++)
     {     // transfer data from source to destination ReturnStatus =
           Program_One_Word( SourceBuf, DestBuf);
           ++DestBuf;
           ++SourceBuf;

          if (!ReturnStatus) return
            ReturnStatus;
     }

     return ReturnStatus;
}


/*************************************************************************/
/* PROCEDURE:     SecID_Lock_Status                                      */
/*                                                                       */
/* This procedure should be used to check the Lock Status of SecID */
```

```
/*                                                                          */
/* Input:                                                                   */
/*            None                                                          */
/*                                                                          */
/* Output:                                                                  */
/*            return TRUE:  indicates SecID is Locked                                 */
/*            return FALSE: indicates SecID is Unlocked                  */
/*************************************************************************/ int
SecID_Lock_Status(void)
{
    WORD SecID_Status;

          //  Issue the Sec ID Entry code to 39VF640X
    *sysAddress(0x555) = 0x00AA;  // write data 0x00AA to device addr 0x555
    *sysAddress(0x2AA) = 0x0055;  // write data 0x0055 to device addr 0x2AA
    *sysAddress(0x555) = 0x0088;  // write data 0x0088 to device addr 0x555
    Delay_150_Nano_Seconds();          // insert delay time = Tida

           // Read Lock Status of SecID segment
    SecID_Status = *sysAddress(0x00FF);
    SecID_Status &= 0x0008; // Unlocked: DQ3=1; Locked: DQ3=0

         // Issue the Sec ID Exit code thus returning the 39VF640X
         // to the read operating mode
    *sysAddress(0x555) = 0x00AA;  // write data 0x00AA to device addr 0x555
    *sysAddress(0x2AA) = 0x0055;  // write data 0x0055 to device addr 0x2AA
    *sysAddress(0x555) = 0x00F0;  // write data 0x00F0 to device addr 0x555
    Delay_150_Nano_Seconds();          // insert delay time = Tida

    if (!SecID_Status)
          return TRUE;                       // SecID segment is Locked

    return FALSE;                            // SecID segment is Unlocked
}


/*************************************************************************/
/* PROCEDURE:     User_SecID_Word_Program                                   */
/*                                                                          */
/* This procedure can be used to program data into the User SecID           */
/* segment (from 000010H--000017H) in 39VF640XB.                                  */
/*                                                                          */
/* NOTE:  1. It's recommended to lock out the SecID segment after the            */
/*           completion of program.                                         */
/*           2. There's no way to unlock the SecID segment once it's*/
/*           locked.                                                        */
/*                                                                          */
/* Input:                                                                   */
/*             SrcWord           Source address to fetch data                      */
/*             Dst    Destination address to write data                 */
/*             length            number of word needs to be programmed           */
/*                                                                          */
/* Output:                                                                  */
/*            return TRUE:  indicates SecID program is successful           */
/*            return FALSE: indicates SecID program is failed or SecID      */
/*                            is locked.                                                    */
/*************************************************************************/ int
User_SecID_Word_Program (WORD *SrcWord, WORD *Dst, int length)
{
    WORD *DestBuf;
    WORD *SourceBuf;
    int test, index=length;

    DestBuf = Dst;

    SourceBuf = SrcWord;

    test = SecID_Lock_Status ();  // check whether the SecID is Locked or notif (test)
                                  // TRUE: SecID is Locked
          return FALSE;

    while (index--){
          *sysAddress(0x555) = 0x00AA; // write data 0x00AA to device addr 0x555
          *sysAddress(0x2AA) = 0x0055; // write data 0x0055 to device addr 0x2AA
          *sysAddress(0x555) = 0x00A5; // write data 0x00A5 to device addr 0x555
          *sysAddress(DestBuf) = *SourceBuf; // transfer the WORD to destination
      ++DestBuf;
      ++SourceBuf;
```

```
        // Read the toggle bit to detect end-of-write for the Sec ID.
        // Do Not use Data# Polling for User_SecID_Word_Program.
      test = Check_Toggle_Ready((Uint32)DestBuf); // wait for TOGGLE bit to get readyif (!test)
        return FALSE;     // SecID Word-Program failed!
    }

    return TRUE;
}

/*****************************************************************************/
/* PROCEDURE:     User_SecID_Lock_Out                                        */
/*                                                                           */
/* This procedure can be used to Lock Out the User Seccurity ID.             */
/* User Security ID segment, from 000010H--000017H, in 39VF640XB.             */
/*                                                                           */
/* NOTE:  1. Call SecID_Lock_Status() first to verify the SecID is                 */
/*           unlocked.                                                                */
/*           2. SecID segment can't be erased.                               */
/*           3. SecID segment can't be unlocked once it's locked.            */
/*                                                                           */
/* Input:      None                                                          */
/*                                                                           */
/* Output:     None                                                          */
/*****************************************************************************/ void
User_SecID_Lock_Out (void)
{
    *sysAddress(0x555) = 0x00AA; // write data 0x00AA to device addr 0x555
    *sysAddress(0x2AA) = 0x0055; // write data 0x0055 to device addr 0x2AA
    *sysAddress(0x555) = 0x0085; // write data 0x0085 to device addr 0x555
    *sysAddress(0x00FF) = 0x0000; // write data 0x0000 to any addr

    Delay_10_Micro_Seconds();              // Wait for Word-Program timeout, Tbp=10us
}


/*****************************************************************************/
/* PROCEDURE:     Erase_Suspend                                              */
/*                                                                           */
/* This procedure can be used to temporarily suspend a Sector/Block-         */
/* Erase operation in 39VF640XB.                                             */
/*                                                                           */
/* Input:      None                                                          */
/*                                                                           */
/* Output:     None                                                          */
/*****************************************************************************/ void
Erase_Suspend (void)
{
    *sysAddress(0x555) = 0x00B0; // write data 0x00B0 to any addr, i.e. 0x555

    Delay_20_Micro_Seconds();              // The device automatically enters read mode
                     // typically within 20 us after the Erase-Suspend command issued.
}
/*****************************************************************************/

/* PROCEDURE:     Erase_Resume                                               */
/*                                                                           */
/* This procedure can be used to resume a Sector-Erase or Block-Erase        */
/* operation that had been suspended in 39VF640XB.                           */
/*                                                                           */
/* Input:      None                                                          */
/*                                                                           */
/* Output:     None                                                          */
/*****************************************************************************/ void
Erase_Resume (void)
{
    *sysAddress(0x555) = 0x0030; // write data 0x0030 to any addr, i.e. 0x555
}
/*****************************************************************************/
/* PROCEDURE:      Check_Toggle_Ready                                        */
/*                                                                           */
/* During the internal program cycle, any consecutive read operation         */
/* on DQ6 will produce alternating 0's and 1's i.e. toggling between         */
/* 0 and 1. When the program cycle is completed, DQ6 of the data will        */
```

```
/* stop toggling. After the DQ6 data bit stops toggling, the device is            */
/* ready for next operation.                                                      */
/*                                                                                */
/* Input:                                                                         */
/*             Dst          must already be set-up by the caller                  */
/*                                                                                */
/* Output:     TRUE    Data toggling success                                      */
/*             FALSE  Time out                                                    */
/**********************************************************************/ int
Check_Toggle_Ready (Uint32 Dst)
{
     WORD PreData;
     WORD CurrData;
     unsigned long TimeOut = 0;

     PreData = *sysAddress(Dst);
     PreData = PreData & 0x0040;               // read DQ6
     while (TimeOut < MAX_TIMEOUT)           // MAX_TIMEOUT=0x07FFFFFF
     {
          CurrData = *sysAddress(Dst);
          CurrData = CurrData & 0x0040;         // read DQ6 againif
          (PreData == CurrData)
          {
                 return TRUE;
          }
             PreData = CurrData;
          TimeOut++;
     }
     return FALSE;
}
/**********************************************************************/
/* PROCEDURE:      Check_Data_Polling                                             */
/*                                                                                */
/* During the internal program cycle, any attempt to read DQ7 of the              */
/* last byte loaded during the page/byte-load cycle will receive the              */
/* complement of the true data.  Once the program cycle is completed,             */
/* DQ7 will show true data.                                                       */
/*                                                                                */
/* Input:                                                                         */
/*             Dst          must already be set-up by the caller                  */
/*             TrueData          this is the original (true) data                        */
/*                                                                                */
/* Output:                                                                        */
/*             TRUE    Data polling success                                       */
/*             FALSE  Time out                                                    */
/**********************************************************************/ int
Check_Data_Polling (Uint32 Dst, WORD TrueData)
{

     WORD CurrData;
     unsigned long int TimeOut = 0;

     TrueData = TrueData & 0x0080;           // read D7
     while (TimeOut < MAX_TIMEOUT)           // MAX_TIMEOUT=0x07FFFFFF
     {
            CurrData = *sysAddress(Dst);
            CurrData = CurrData & 0x0080;        // read DQ7
            if (TrueData == CurrData)
           {
                return TRUE;
           }
             TimeOut++;
     }
     return FALSE;
}
```

## 2. Result

In this article, an overview a flash momory and related IC has introduced.

**Acknowledgement**

Here it announce, for this paper no fund has been get and writing this article is only based on person interests.

**References**

1. Novotný, R., Kadlec, J., & Kuchta, R. (2015). Nand flash memory organization and operations. *Journal of Information*

*Technology & Software Engineering, 5*(1), 8.

2. Pavan, P., Bez, R., Olivo, P., & Zanoni, E. (1997). Flash memory cells-an overview. *Proceedings of the IEEE, 85*(8), 1248-1271.
3. Mohan, V., Gurumurthi, S., & Stan, M. R. (2010). Flash Power: A detailed power model for NAND flash memory. In *2010 Design, Automation & Test in Europe Conference & Exhibition* (DATE 2010) (pp. 502-507). IEEE.
4. Heer, T. S. (2019). Nand flash memory characterization (Doctoral dissertation, UC Santa Cruz).