Software Sample Code:

The sample-code below is used to describe how to design s/w for Macronix flash.

The following sample codes are suitable for AMD compatible flash:

1. **3.3V Flash: MX29LV004, MX29LV040, MX29LV400 MX29LV008, MX29LV081, MX29LV800, MX29LV800A, MX29LV800B, MX29LV017, MX29LV017A, MX29LV160, MX29LV160A, MX29LV160B, MX29LV033, MX29LV033A, MX29LV320, MX29LV320A, MX29LV065, MX29LV640, MX29LV640B, MX26LV400, MX26LV800, MX26LV160.**

2. **3.3V NBit: MX29LV64xM H/L, MX29LV640MB/T, MX29LV320MB/T, MX29LV320MH/L.**

3. **5V Flash: MX29F001, MX29F100, MX29F002(N), MX29F022(N), MX29F200, MX29F004, MX29F040, MX29F400, MX29F080, MX29F800, MX29F016.**

4. **Other devices: MX26L3220, MX26L3221, MX26L6420, MX26L6421,**

(Low voltage products are not included in the following sample code.)

We suggest customers to check ID codes first before porting algorithms. It is easy to confirm that the flash device accepts the command sets correctly.

Read_ID_Op()

```
{

   // For word mode: (Depends on system application)
        Write data AAH to address 555H
        Write data 55H to address 2AAH
        Write data 90H to address 555H
        Read Manufacture code from ADI (Address of Device Identifier, A1=0 ; A0 =0 , others don't care )
        Read Device code from ADI (Address of Device Identifier, A1 =0 ; A0 = 1, others don't care )



    // For Byte mode: (Depends on system application)
        Write data AAH to address AAAH
        Write data 55H to address 555H
        Write data 90H to address AAAH
        Read Manufacture code from ADI (Address of Device Identifier, A1=0 ; A0 =0 ; A-1 & others don't care )
        Read Device code from ADI (Address of Device Identifier, A1 =0 ; A0 = 1; A-1 & others don't care )
}
```

Chip_Erase_Op()

```
{
   loopCounter = 0L;
   TimeOutFlag = FALSE;
   eraseOkFlag = TRUE;                    // clear the erase ok flag to be TRUE
   write Chip_Erase_command ;             // write 6 commands to enter Chip Erase operation

   oldflag = read status of flash;        / * To confirm WSM has accepted the command correctly and is busy */
   newflag = read status of flash;        /* in erase operation if the Q6(toggle bit) is toggling */
   if ( ! (oldflag ^ newflag ) )
   {
      set eraseOkFlag = FLASE ;
      return(eraseOkFlag);                // the inputted command is false, try to write it again
   }

   while(TimeOutFlag )                    // set a loop to check the flash's WSM till ready or system time out
   {
      oldflag=read status of flash;
      if((oldflag == 0xff ))              // if yes, Q7 to Q0 are all '1', then erase is successful
         break;                           // exit erase operation polling loop
      else if ((oldflag & 0x20) == 0x20 ) // if Q5==1, then erase operation is time out
      {
         set eraseOkFlag = FALSE;         // set the erase operation is failed
         write reset command to reset flash; // write RESET command to clear fail status
         break;                           // break to exit the erase operation */
      }
      delay_time;                         // the delay time is adjustable and depends on the system clock
      if(TimeOut < loopCounter++)         // The number for Timeout must be set bigger than the Max. of the spec.
      {                                   // Basically, this timeout loop to prevent system hang up is not necessary,
         EraseOKFlag = FALSE;             // since the flash already provides the Q5 timeout bit for that function.
         break;
      }
   }
   return (eraseOkFlag);                  // report the final result to main program
}                                         /* end of Chip_erase_Op */
```

//

Block_Erase_Op() // MX26L3220 and MX26L6420 don't support this function

```
{
  loopCounter = 0L;
  TimeOutFlag = FALSE;
  eraseOkFlag = TRUE;                  /* clear the erase ok flag to be TRUE */
  write Block_Erase_command ;          /* write 5 commands to enter Block Erase operation */
  for(;;)
  {
    write Block-Address with 30H command;   /* 30H is a command for Block erase operation
                                               The successive block address load cycle must be done within
                                               80us from the rising edge of /WE to falling edge of /WE */
    if(last block address is True)          // end of the erasing block-address input
      break;
  }
  wait 80us ;                          // Delay 80us wait for closing Block-address load window and entering Erase
                                       // Or check Q3== 1, It means input window time-out also */
  oldflag = read status of flash;       /* To confirm WSM has accepted the command correctly and is busy */
  newflag = read status of flash;       /* in erase operation if the Q6(toggle bit) is toggling */
  if ( ! (oldflag ^ newflag ) )        // Bypass this flow if system has an interrupt service and is over 0.3sec.
  {
    set eraseOkFlag = FLASE;
    return(eraseOkFlag);               // the inputted command is false, try to write it again
  }

  while(TimeOutFlag )                  // set a loop to check the flash's WSM till ready or system time out
  {
    oldflag=read status from erased block address;
    if((oldflag == 0xff ))             /* if yes, Q7 to Q0 are all '1', then erase is successful */
      break;                           /* exit erase operation polling loop*/
    else if ((oldflag & 0x20) == 0x20 ) /* if Q5==1, then erase operation is time out*/
    {
      set eraseOkFlag = FALSE;         /* set the erase operation is failed */
      write reset command to reset flash; /* write RESET command to clear fail status */
      break;                           /* break to exit the erase operation */
    }
    delay_time;                        // the delay time is adjustable and deponed on the system clock
    if(TimeOut < loopCounter++)         // The number for Timeout must be set bigger than Max. of the spec.
    {                                  // Basically, this time out loop to prevent system hang up is not necessary,
      EraseOKFlag = FALSE;             // since the flash already provides the Q5 timeout bit for that purpose.
      break;
    }
  }
  return (eraseOkFlag);                /* report the final result to main program */
}                                      /* end of Chip_erase_Op */
```

//

Program_Op()

```
{
   loopCounter = 0L;
   TimeOutFlag = FALSE;
   ProgramOkFlag = TRUE;                  // clear the program ok flag to be TRUE
   write Program_command ;                // write 3 commands to enter program operation
   write Valid_Data to flash;             // write Valid data to flash follow the write program cmd

   oldflag = read status of flash;         /* To confirm WSM has accepted the command correctly and is busy */
   newflag = read status of flash;         /* in program operation if the Q6(toggle bit) is toggling.
   if ( ! (oldflag ^ newflag ) )          // Bypass this flow if system's read speed is slower than 3us
   { set eraseOkFlag = FLASE ;            // or interrupt service could not disabled.
     return(eraseOkFlag);                 // the inputted command is false, try to write it again
   }

   while(TimeOutFlag )                    // set a loop to check the flash's WSM till ready or system times out
   {
     read_Data = read flash at programmed address;
     if(read_Data == programmed_Data)     // if yes, write data== read out data, program is successful */
        break;                            // exit program operation polling loop*/
     else ((newflag & 0x20) == 0x20 )     // if Q5==1, then program operation is timed out*/
     {
        set ProgramOkFlag = FALSE;        // set the program operation is failed */
        write reset command to reset flash; // write RESET command to clear fail status */
        break;                            // break to exit the program operation */
     }
     delay_time;                          // the delay time is adjustable and depends on the system clock
     if(TimeOut < loopCounter++)          // The number for Timeout must be set bigger than Max. of the spec.
     {                                    // Basically, this time out loop to prevent system hang up is not necessary,
        ProgramOKFlag = FALSE;            // since the flash already provides the Q5 timeout bit for that purpose.
        break;
     }
   }
   return (ProgramOkFlag);                      /* report the final result to main program */
}                                               /* end of Program_Op */
```

//

/* To disable all system interrupts, and the sector erase function must be executed before suspend / resume flow as follows */

Suspend_Op() / Resume_Op() // MX26L3220/1, MX26L6420/1, MX26LV800. do not support this function

```
{
  int EraseOKFlag=TRUE;
  unsigned short oldflag, newflag;

  Disable all of system interrupts;
  Write Block_Erase_command;
  oldflag = read status of flash ;
  while ((oldflag & 0x80h ) ! = 0h )        // To confirm WSM has accepted the command correctly and is busy */Q7 == 0,
                                               It means busy in erasing
  While(TRUE){
    Write Erase_Suspend_command;

    While(true)
    {
      oldflag = read status of flash;        // To confirm WSM has accepted the command correctly and is busy
      newflag = read status of flash;        // if the Q6(toggle bit) is toggling.
      if ( oldflag = = newflag )             //      ^ exclusive OR
        break;
    }

    Read / Program data from/to Non-erased sectors;     // Q2 is still toggling, if the data read out from erased sector

    Write Erase_Resume_command;
    Delay 10ms ;

    oldflag=read status of flash;
    if((oldflag == 0xff ))                  // if yes, Q7 to Q0 are all '1', then erase is successful
      break;                                // exit erase operation polling loop
    else if ((oldflag & 0x20) == 0x20 )     // if Q5==1, then erase operation is time out
    {
      set eraseOkFlag = FALSE;              // set the erase operation is failed
      write reset command to reset flash;   // write RESET command to clear fail status
      break;                                // break to exit the erase operation */
    }
  }

  Enable System interrupt;
  Return (eraseOkFlag);
}
```

Sector_protect_Op(Assgin_sector Addr**)** //MX29F080/800, MX29F160, MX29LVxxx, MX26L3220/1, MX26L6420/1, MX26LV800 do not support this function

```
{
   loopCnt;
   ProtectOkFlag = FALSE;                  // clear the protect ok flag to be FALSE

   for(loopCnt=0; loopCnt<32 ; loopCnt++)  // 32 retrial times for sector protected operation
   {
      write Unlock_for_Sector_Protect/Unprotect_command ;  // write 6 command cycles to initiate protect operation
      write Assign_sectorAddr_for_7thcommand;              // write 7th cmd with assigned sector address and A6=0, A9=1 and
                                                           //data does not care except 0xf0.
      while(TRUE )                                         // set a loop to polling flash's WSM till ready
      {
         oldflag = read status of flash;          // read status from flash for checking Q6 state
         newflag = read status of flash;           .
         if ( ! (oldflag ^ newflag ) )             // if Q6 stop toggling then break ,
            break;
      }
      read_Data = read status of flash ;           // read status from assigned sector address and A1=1
      if (read_Data == 0x01)                       // if yes, read out data is equal to 0x01, then protection is successful
      {
         ProtectOKFlag = TRUE;
         break;                                    // exit protection retry loop operation
      }
   }
   write Reset_command;                           // write reset command to reset flash back normal read mode
   return (ProgramOkFlag);                        /* report the final result to main program */
}                                                 /* end of Sector_protect_Op */
```

//

Chip_unprotect_Op( ) // MX29F080/800, MX29F160, MX29LVxxx, MX26L3220/1, MX26L6420/1, MX26LV800 do not support this function

```
{
   AllSectorNo;
   loopCnt;
   UnProtectOkFlag = FALSE;                  // clear the protect ok flag to be FALSE

   for(i=0; i<AllSectorNo; i++)
   {
      if( Sector_protect_Op(sectorAddress[ i ]) == FALSE)   // call Sector_protect routine to protect all sectors
         return(protect all sectors false);                 // before chip unprotected
   }

   for(loopCnt=0; loopCnt<1000 ; loopCnt++)          // 1000 retrial times for chip unprotected operation
   {
      write Unlock_for_Sector_Protect/Unprotect_command ;  // write 6 command cycles to initiate protect operation
      write 7th_command;                                   // write 7th cmd with A6=1, A9=1 and other address don't care
                                                           // and data doesn't care except 0xf0.
      while(TRUE )                                         // set a loop for polling flash's WSM till ready
      {
         oldflag = read status of flash;          // read status from flash for checking Q6 state
         newflag = read status of flash;           .
         if ( ! (oldflag ^ newflag ) )             // if Q6 stop toggling then break ,
            break;

      }
      read_Data = read status of flash ;           // read status from A1=1 others don't care
      if (read_Data == 0x00)                       // if yes, read out data is equal to 0x00, then unprotected is successful
      {
         UnProtectOKFlag = TRUE;
         break;                                    // exit unprotected retrial loop operation
      }
   }
   write Reset_command;                           // write reset command to reset flash back normal read mode
   return (UnProgramOkFlag);                      /* report the final result to main program */
}                                                 /* end of Chip_unprotect_Op */
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

The following sample codes are suitable for all of the Macronix proprietary flash, such as MX29F1610A/B, MX29F1611, MX29F8100, MX29L1610, MX29L1611, MX29L3211, MX29L8100, MX29L8000 and so on…

Chip_Erase_Op()

```
{
   loopCounter = 0L;
   TimeOutFlag = FALSE;
   eraseOkFlag = TRUE;                    // clear the erase ok flag to be TRUE
   write Chip_Erase_command ;             // write 6 commands to enter Chip Erase operation

   while(TimeOutFlag)                     // set a loop to check the flash's WSM till ready or system time out
   {
      StatusReg = read status of flash;   // read status from flash
      if((statusReg    & 0x80) == 0x80)   // Is the Q7(ready/busy bit) ready or not ?
      {
         if((statusReg & 0x20 = = 0x00)   // It is successfully erased if Q5(erase flag) is equal to '0'
            break;                        // exit chip erase polling loop
         else
         { set EraseOKFlag = FALSE;       // if Q5 is '1', then the chip erase was failed
           write Clear_S.R._command;      // write Clear status command to clear status Reg.
           break;
         }
      }
      delay_time;                         // the delay time is adjustable and deponed on the system clock
      if(TimeOut < loopCounter++)         // The number for timeout must be set bigger than Max. of the spec.
      {                                   // Basically, this time out loop to prevent system hang up is not necessary,
         EraseOKFlag = FALSE;             // since the flash already provides the Q7 ready/busy bit for that function.
         write Clear_S.R._command;        // write Clear status command to clear status Reg.
         break;
      }
   }
   write reset command;                   // RESET command to reset flash to read operation
   return (eraseOkFlag);                  // report the final result to main program
}                                         /* end of Chip_erase_Op */
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

Page_Program_Op()

```
{
   int i;
   loopCounter = 0L;
   TimeOutFlag = FALSE;
   ProgramOkFlag = TRUE;                  // clear the program ok flag to be TRUE
   Disable system's Interrupt;            // System interrupts might cause page data downloaded flow to be improperly
                                          // completed. Disable all system interrupts before going to page program operation.
   write Program_command ;                // write 3 commands to enter program operation
   for(i=0; i<page number; i++)           // set a loop to down load whole page data into flash's buffer
   {                                      // the page number is 128 bytes or 64 words for MXIC a series flash with page
                                          // Program operation, except for MX29L3111 whose page number is 256 bytes or
                                          // 128 words.
      load Valid_Data with relative page address into flash; /* Load whole page data into flash's buffer follow the write Program
                                                                cmd and the successive data must be loaded within 30us from
                                                                the rising edge of /WE to falling edge of /WE */
   }
   Enable system Interrupt;               // to enable system interrupt
   wait 100us;                            // wait 100us for closing page data load window and entering Program operation

   while(TimeOutFlag)                     // set a loop to check the flash's WSM till ready or system time out
   {
      StatusReg = read status of flash;   // read status from flash's status Reg.
      if((statusReg    & 0x80) == 0x80)   // Is the Q7(ready/busy-bit) ready    ?
      {
         if((statusReg & 0x10 == 0x00)    // It is successful to program if Q4(Program flag) is equal to '0'
            break;                        // exit program polling loop
         esle
         { set ProgramOkFlag = FALSE;     // if Q4 is '1', then program has failed */
            write Clear_S.R._command;     // write Clear status command to clear status Reg.
            break;
         }
      }
      delay_time;                         // the delay time is adjustable and deponed on the system clock
      if(TimeOut < loopCounter++)         // The number for timeout must be set bigger than Max. of the spec.
      {                                   // Basically, this time out loop to prevent system hang up is not necessary,
         ProgramOKFlag = FALSE;           // since the flash already provides the Q7 Ready/busy bit for that function.
         write Clear_S.R._command;        // write Clear status command to clear status Reg.
         break;
      }
   }
   write reset command;                   // RESET command to reset flash to read operation
   return (ProgramOkFlag);                // report the final result to main program
}                                         /* end of Page_Program_Op */
```

The following sample codes are suitable for Macronix propreitary flash, like the MX29L160 and MX29VW160.

Chip_Erase_Op()

```
{
   loopCounter = 0L;
   TimeOutFlag = FALSE;
   eraseOkFlag = TRUE;                    // clear the erase ok flag to be TRUE
   write Chip_Erase_command ;             // write 6 commands to enter Chip Erase operation

   while(TimeOutFlag)                     // set a loop to check the flash's WSM till ready or system time out
   {
      StatusReg = read status of flash;   // read status from flash
      if((statusReg    & 0x80) != 0x80)   // Is the Q7(ready/busy bit) ready or not ?
      {                                   // If still in busy mode, check Q5 timeout flag
         if((statusReg & 0x20 = = 0x20)   // If timeout has happened, Q5(timeout flag) will be equal to '1'
         {
           set EraseOKFlag = FALSE;       // if Q5 is '1', then the chip erase has failed
           break;                         //exit chip erase polling loop
         }
      }
      delay_time;                         // the delay time is adjustable and depends on the system clock
      if(TimeOut < loopCounter++)         // The number for timeout must be set bigger than the Max. of the spec.
      {                                   // Basically, this time out loop to prevent system hang up is not necessary,
         EraseOKFlag = FALSE;             // since the flash already provides the Q7 ready/busy bit for that function.
         break;
      }
   }
   write reset command;                   // RESET command to reset flash to read operation
   return (eraseOkFlag);                  // report the final result to main program
}                                         /* end of Chip_erase_Op */
```

//

Page_Program_Op()

```
{
   int i;
   loopCounter = 0L;
   TimeOutFlag = FALSE;
   ProgramOkFlag = TRUE;                  // clear the program ok flag to be TRUE
   Disable system's Interrupt;            // System interrupts might cause page data downloaded flow to be improperly
                                          // ended. Disable all system interrupts before going to page program operation.
   write Program_command ;                // write 3 commands to enter program operation
   for(i=0; i<page number; i++)           // set a loop to down load whole page data into flash's buffer
   {                                      // the page number is 128 bytes or 64 words for Macronix flash with page
                                          // Program operation, except for MX29L3111which has page number of 256 bytes
or
                                          // 128 words.
      load Valid_Data with relative page address into flash; /* Load whole page data into flash's buffer follow the write Program
                                                                cmd and the successive data must be loaded within 30us from
                                                                the rising edge of /WE to falling edge of /WE */

   }

   Reload the last Valid_Data and address into flash;   // Reload the last data and address into flash's buffer to complete the
                                                        //    loaded procedure

   Enable system's Interrupt;             // to enable system's interrupt
   oldflag = read status of flash;        // Polling Q6 toggle bit for closing page data load window and entering program
   newflag = read status of flash;        //      operation. If Q6 start to toggle, the programming is in progress
   while (!(oldflag ^ newflag ) )         // Bypass this flow if system's read speed is slower than 3us
   {
      oldflag = newflash;
      newflag = read status of flash;
   }

   while(TimeOutFlag)                     // set a loop to check the flash's WSM till ready or system time out
   {
      CheckData = read data of flash with last address of page;          // read a byte/word data from flash with relative page
address
                                                                        //    for comparing, the programmed result is successful or not
      if((CheckData != Last valid data)   // Is the data correct ? If still in progress, check the timeout Q5 state
      {
         if((CheckData & 0x20 == 0x20)    // If timeout happened, Q5 will be equal to '1'
         {                                // If Q5 is '1', the programming was not successful,
            set ProgramOkFlag = FALSE;    //    Set the failed state for programming
            break;                        //
         }
      }
      delay_time;                         // the delay time is adjustable and depends on the system clock
      if(TimeOut < loopCounter++)         // The number for timeout must be set bigger than the Max. of the spec.
      {                                   // Basically, this time out loop to prevent system hang up is not necessary,
         ProgramOKFlag = FALSE;
         Break;
      }
   }
   write reset command;                  // RESET command to reset flash to read operation
   return (ProgramOkFlag);               // report the final result to main program
}
                                         /* end of Page_Program_Op */
```

The following sample codes (Read_ID_Op/Buffer_Program_Op) are suitable for Macronix flash, MX29LV64xM H/L, MX29LV640MB/T, MX29LV320MB/T, MX29LV320MH/L.

.

Read_ID_Op()

{

// For word mode: (Depends on system application)

Write data = 0xAAH to address = 555H
Write data = 0x55H to address = 2AAH
Write data = 0x90H to address = 555H
MfrID= read flash at address=0x00000; // read mfr. ID at Bank address + 0x00
DevID1= read flash at address=0x00001; // read Device ID1 at Bank address +0x01
DevID2= read flash at address=0x0000E; // read Device ID2 at Bank address +0x0E
DevID3= read flash at address=0x0000F; // read Device ID3 at Bank address +0x0F

Write data=0x0f0 to address = don't care; // Write RESET(0x0F0), return to normal read mode

// For Byte mode: (Depends on system application)

Write data = 0xAAH to address = AAAH
Write data = 0x55H to address = 555H
Write data = 0x90H to address = AAAH
MfrID= read flash at address=0x00000; // read mfr. ID at Bank address + 0x00
DevID1= read flash at address=0x00002; // read Device ID1 at Bank address +0x02
DevID2= read flash at address=0x0001C; // read Device ID2 at Bank address +0x1C
DevID3= read flash at address=0x0001E; // read Device ID3 at Bank address +0x1E

Write data=0x0f0 to address = don't care; // Write RESET(0x0F0), return to normal read mode

Buffer_Program_Op()

```
{
   int i;
   loopCounter = 0L;
   TimeOutFlag = FALSE;
   ProgramOkFlag = TRUE;                  // clear the program ok flag to be TRUE
   Disable system's Interrupt;            // System interrupts might cause page data downloaded flow to be improperly
                                          // ended. Disable all system interrupts before going to page program operation.
   Write data = 0xAAH to address = 555H   // Write "Write to buffer command
   Write data = 0x55H to address = 2AAH
   Write data = 0x25H to address = Sector Address
   Write data= Word counter to address = Sector Address       // Write word counter to Sector Address

   for(i=0; i<page number; i++)           // set a loop to down load whole page data into flash's buffer
   {                                      // the page number is 32 bytes or 16 words for Macronix flash with page
                                          // Program operation.
   Write data=WDi to address= WAi;        // WDi stands for write data to WA = write address

   }

   Write data = 0x29 to address = Sector Address     // Write Program buffer to flash command

   Enable system's Interrupt;             // to enable system's interrupt
   oldflag = read status of flash;         /* To confirm WSM has accepted the command correctly and is busy */
   newflag = read status of flash;         /* in program operation if the Q6(toggle bit) is toggling.
   if ( ! (oldflag ^ newflag ) )          // Bypass this flow if system's read speed is slower than 3us
   { set ProgramOkFlag = FLASE ;              // or interrupt service could not be disabled.
     return(ProgramOkFlag);                   // the inputted command is false, try to write it again
   }

   while(TimeOutFlag )                    // set a loop to check the flash's WSM till ready or system time out
   {
     read_Data = read flash at programmed address;
     if(read_Data == programmed_Data)     // if yes, write data== read out data, program is successful */
        break;                            // exit program operation polling loop*/
     else ((newflag & 0x20) == 0x20 )     // if Q5==1, then program operation is timed out*/
     {
        set ProgramOkFlag = FALSE;        // set the program operation has failed */
        write reset command to reset flash; // write RESET command to clear fail status */
        break;                            // break to exit the program operation */
     }
     delay_time;                          // the delay time is adjustable and depends on the system clock
     if(TimeOut < loopCounter++)          // The number for Timeout must be set bigger than the Max. of the spec.
     {                                    // Basically, this time out loop to prevent system hang up is not necessary,
        ProgramOKFlag = FALSE;            // since the flash already provides the Q5 timeout bit for that function.
        break;
     }
   }

   return (ProgramOkFlag);                // report the final result to main program
}                                          /* end of Page_Program_Op */
```