MX29F1610 16Mb FLASH MEMORY

Since the erase/program time for flash memory are depended on the operation temperature, number of cycle, Vcc, and other factors. This application note is trying to optimize the erase/program performance by using the polling technology to check the status register in flash. In addition to the polling, a time out timer is recommended in order to prevent the dead lock in software. Two examples of software routines are written in "C" in order to demonstrate erase and program the MX29F1610 16Mb flash memory.

Please also refer to the Macronix's MX29F1610 1998 March data sheets for the flow charts, timing diagrams, erase/programming performance table and completed command sets.

```
//*********************************************************************
//
//                          MX29F1610 16M bit flash subroutines
//
// These code are copied from Macronix Flash evaluation kid work in PC
// Flow charts, timing, commands, please refer to Macronix MX29F1610 data sheets.
// By Joshua Jou Mar. 11, 1998, copyright 1998 Macronix International Co., Ltd.
//
//    Main routines:
//    1. MX29F1610 16M bit flash page programming subroutine
//    2. MX29F1610 16M bit flash chip erase subroutine
//
//*********************************************************************

int f16m_Page_Program (int read_length,long ProgAddr)
char *f16m_Chip_Erase (void);
int f16m_Clear_Status (void);
int f16m_Read_Status (void);
int f16m_DQ7_Polling(void)

#define ControlPort 0x7E0       // control byte/word mode, /WP, /WE, /PWD
#define WindowPort 0x7F0        // dram buffer, bank switch
#define fmemptr 0xd0000         // dram buffer address in evaluation kit

// setup command and address
#define F16MSetupAddr1 0xaaaa
#define F16MSetupAddr2 0x5554
#define F16MSetupAddr3 0xaaaa
#define F16MSetupCmd1 0xaa
#define F16MSetupCmd2 0x55

// commands: erase, program
#define F16M_PROGRAM_CMD 0xA0
#define F16M_CHIP_ERASE_CMD1 0x80
#define F16M_CHIP_ERASE_CMD2 0x10
#define F16M_BLOCK_ERASE_CMD1 0x80
#define F16M_BLOCK_ERASE_CMD2 0x30
#define F16M_CLEAR_STATUS_CMD 0x50
#define F16M_READ_STATUS_CMD 0x70

// Control signal definition
#define cdefault    0x1f
#define wp          0x1b
#define cbyte       0x1e
#define m16_wp_byte    cdefault & wp & cbyte        // binary 1011


//*********************************************************************
//    Function: MX29F1610 16M bit flash page programming subroutine
//    command:Write [5555],    AAH
//                Write [2AAA], 55H
//                Write [5555],    A0H
//                Page Write [ProgAddr],data_buffer[i]
//    input: ProgAddr -       page address of flash to be written
//           data_buffer[i] - data to be programmed stored in DRAM buffer
//           read_length -    maximum of 128 bytes/ 128 words of data may
//                            be written into each page programming flow
//    output: 0, if successful; -1, if failed
```

```
//     note: Flash internal program timer will time out itself in around 150 ms
//*********************************************************************
f16m_Page_Program (int read_length,long ProgAddr)
{
int     temp;     // temperary storage

// 3 cycles page program command
fmemptr[F16MSetupAddr1]=F16MSetupCmd1;
fmemptr[F16MSetupAddr2]=F16MSetupCmd2;
fmemptr[F16MSetupAddr3]=F16M_PROGRAM_CMD;

// write the data from DRAM buffer into flash internal data latch
for ( temp=0 ;     i<read_length ; temp++)
       fmemptr[ProgAddr+i]=data_buffer[temp];

delay_100us(1);     // wait for 100 us, flash will begin the programming
automatically

// flash internal program fail timer is about 150 ms
// the flash is bad if we had waited for more than 200 ms
temp=f16m_DQ7_Polling();     // wait until the flash is ready (done or time out)

// check to see if the programming is successful
if     (temp || ( fmemptr[0] & 0x10))     // time out ?;     DQ4 =1, if failed
    {
    sprintf(message,"Programming fail");
    f16m_Clear_Status();
    return -1;
    }
else return 0;     // programming successful

}       // end of f16m_Page_Program



//*********************************************************************
//     Function: MX1610 16M bit flash chip erase subroutine
//     command: Write [5555],     AAH
//                         Write [2AAA], 55H
//                         Write [5555],     80H
//                         Write [5555],     AAH
//                         Write [2AAA], 55H
//                         Write [5555],     10H
//     input: none
//     output: string of result for erasing
//               "Chip Erase Failed (time out)"
//               "Chip Erase Failed"
//               "Chip Erase Successfully Completed"
//     note: Flash internal chip/sector erase time out timer is around 2 seconds
//*********************************************************************
char *f16m_Chip_Erase()
{
  unsigned char data;     // temperary storage
  unsigned int check_counter=0;          // how many time do we check, erase fail
timer

  // assign dram buffer address in evaluation kit (memory map)
  fmemptr=(unsigned char far *)MK_FP(Decode_Seg,0);

  // command to set 1.byte mode and 2.unprotected
  outportb(ControlPort,m16_wp_byte);       // output to control port with data 0x1a

  // 6 cycles chip erase command
  fmemptr[F16MSetupAddr1]=F16MSetupCmd1;
  fmemptr[F16MSetupAddr2]=F16MSetupCmd2;
  fmemptr[F16MSetupAddr3]=F16M_CHIP_ERASE_CMD1;
  fmemptr[F16MSetupAddr1]=F16MSetupCmd1;
  fmemptr[F16MSetupAddr2]=F16MSetupCmd2;
  fmemptr[F16MSetupAddr3]=F16M_CHIP_ERASE_CMD2;

  // wait until flash begins the erase command (busy)
  while (fmemptr[0] &0x80);     // check DQ7: DQ7=1, if flash is ready; DQ7=0 if
busy

  // wait until the erase is completed or time out
  // flash internal erase fail timer is about 2 seconds
  // the flash is bad if we had waited for more than 3 seconds
  do {
        data=fmemptr[0];
        check_counter ++;
        } while(!( data & 0x80 ) && check_counter <60000);
        // jump out if ready or time out

  // show the error message if flash time out
  if(check_counter >=60000)
      {
      sprintf(message,"Chip Erase Failed (time out)");
      return message;
      }

  // show the error message if flash erase failed
```

```
  if( ((fmemptr[0])& 0x20))                          // DQ5=1 if erase failed
      {
        sprintf(message,"Chip Erase Failed");
        f16m_Clear_Status();
        return message;
      }

  sprintf(message,"Chip Erase Successfully Completed");
  f16m_Clear_Status();
  return message;
}                    // end of char *f16m_Chip_Erase()


//****************************************
// MX1610 16M bit flash clear status subroutine
// input: none
// output: always 0
//****************************************
int f16m_Clear_Status()
{
  fmemptr=(unsigned char far *)MK_FP(Decode_Seg,0);
  f16m_Read_Status();
  fmemptr[F16MSetupAddr1]=F16MSetupCmd1;
  fmemptr[F16MSetupAddr2]=F16MSetupCmd2;
  fmemptr[F16MSetupAddr3]=F16M_CLEAR_STATUS_CMD;
  return 0;
}       // end of f16m_Clear_Status()


//****************************************
// MX1610 16M bit flash read status subroutine
// input: none
// output: flash status
//****************************************
int f16m_Read_Status()
{
  fmemptr=(unsigned char far *)MK_FP(Decode_Seg,0);
  fmemptr[F16MSetupAddr1]=F16MSetupCmd1;
  fmemptr[F16MSetupAddr2]=F16MSetupCmd2;
  fmemptr[F16MSetupAddr3]=F16M_READ_STATUS_CMD;
  st.value=fmemptr[0];
  return (fmemptr[0]);
}       //// end of f16m_Read_Status()
```

```
//****************************************
// MX1610 16M bit flash read DQ7 in status register
// input: none
// output:     0 if flash is ready
//                -1 if time out error after 200 ms
//****************************************
int f16m_DQ7_Polling(void)
{
  long timeout;

  f16m_Read_Status();     // assign the fmemptr
  timeout=0L;

// wait until flash if ready or time out
  while(!( fmemptr[0] & 0x80 ) && timeout < 20)
      {
        timeout++;
        delay(10);        // delay 10ms
        }
  if (timeout>=20) return -1;
  return(0);
}     // end of f16m_DQ7_Polling(void)
```