APPLICATION NOTE

2M FLASH MEMORY

Yes. You can replace Intel 2M flash very easily with Macronix' s flash.

Intel Part Number	Replace with	Configuration, Package
28F200-T	MX28F2100T	2Mx16 48TSOP, 44 SOP
28F200-B	MX28F2100B	2Mx16 48TSOP, 44 SOP
28F002-T	MX28F002-T	2Mx8 40TSOP
28F002-B	MX28F002-B	2Mx8 40TSOP
28F020	MX28F2000P	2Mx8 bulk erase, 32 PDIP, 32 PLCC

Table of Contents

2M FLASH MEMORY MAPS V.S. Intel

1. Memory Map Cross Reference

1-1 MX28F2100 MEMORY MAPS V.S. Intel 28F200

MX28F2100 MEMORY MAP IS FULLY COMPATIBLE WITH INTEL 28F200

x16, word address		Intel 28F200-T	Intel 28F200-B
from	to	MX28F2100T	MX28F2100B
1E000H	1FFFFH	16KB boot	128KB main
1D000H	1DFFFH	8KB parameter	
1C000H	1CFFFH	8KB parameter	
10000H	1BFFFH	96KB main	
04000H	0FFFFH	128KB main	96KB main
03000H	03FFFH		8KB parameter
02000H	02FFFH		8KB parameter
00000H	01FFFH		16KB boot

1-2 MX28F002 MEMORY MAPS V.S. Intel 28F002

MX28F002 MEMORY MAP IS FULLY COMPATIBLE WITH INTEL 28F002

x8, byte address		Intel 28F002-T	Intel 28F002-B
from	to	MX28F002-T	MX28F002-B
3C000H	3FFFFH	16KB boot	128KB main
3A000H	3BFFFH	8KB parameter	
38000H	39FFFH	8KB parameter	
20000H	37FFFH	96KB main	
08000H	1FFFFH	128KB main	96KB main
06000H	07FFFH		8KB parameter
04000H	05FFFH		8KB parameter
00000H	03FFFH		16KB boot

1-3 MX28F2000P MEMORY MAPS V.S. Intel 28F020 , AMD 28F020 AND , AMD 28F020A

MX28F2000P MEMORY MAP IS THE SUPER SET OF INTEL 28F020, AMD 28F020 AND AMD 28F020A (BULK ERASE).

MX28F2000P CAN BE USED AS BULK ERASE, BLOCK ERASE OR MULTIPLE BLOCK ERASE.

MX28F2000P CAN ALSO BE USED FOR BOTH TOP OR BOTTOM BOOT BLOCK

x8, byte address		MX28F2000P	Intel 28F020 AMD 28F020A AMD 28F020
from	to		
3F000H	3FFFFH	4KB boot	256KB main
3E000H	3EFFFH	4KB boot	
3D000H	3DFFFH	4KB parameter	
3C000H	3CFFFH	4KB parameter	
38000H	3BFFFH	16KB main	
34000H	37FFFH	16KB main	
30000H	33FFFH	16KB main	
2C000H	2FFFFH	16KB main	
28000H	2BFFFH	16KB main	
24000H	27FFFH	16KB main	
20000H	23FFFH	16KB main	
1C000H	1FFFFH	16KB main	
18000H	1BFFFH	16KB main	
14000H	17FFFH	16KB main	
10000H	13FFFH	16KB main	
0C000H	0FFFFH	16KB main	
08000H	0BFFFH	16KB main	
04000H	07FFFH	16KB main	
03000H	03FFFH	4KB parameter	
02000H	02FFFH	4KB parameter	
01000H	01FFFH	4KB boot	
00000H	00FFFH	4KB boot	

2M FLASH MEMORY MAPS V.S. Intel

1-4 MX28F2100 MEMORY MAPS

MX28F2100T V.S. AMD Am29F200T

x16, word address			
from	to	MX28F2100T	Am29F200T
1E000H	1FFFFH	16KB boot	16KB boot
1D000H	1DFFFH	8KB parameter	8KB parameter
1C000H	1CFFFH	8KB parameter	8KB parameter
18000H	1BFFFH	96KB main	32KB main
10000H	17FFFH		64KB main
08000H	0FFFFH	128KB main	64KB main
00000H	07FFFH		64KB main

MX28F2100B V.S. AMD Am29F200B

x16, word address			
from	to	MX28F2100B	Am29F200B
18000H	1FFFFH	128KB main	64KB main
10000H	17FFFH		64KB main
08000H	0FFFFH	96KB main	64KB main
04000H	07FFFH		32KB main
03000H	03FFFH	8KB parameter	8KB parameter
02000H	02FFFH	8KB parameter	8KB parameter
00000H	01FFFH	16KB boot	16KB boot

2. Engineering quick reference for feature and boot block handling differences

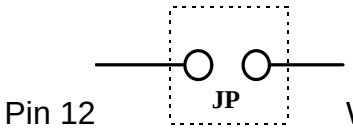
2-1 Intel 28F002 V.S. MX28F002 (Feature)

	H/W			S/W
	Pin 10 RP#	Pin 12***	16k boot block	multi-block erase
Intel 28F002BV-T/B	Yes	WP#	Yes(WP#,RP#)	No
Intel 28F002BX-T/B	Yes	don't use (DU)	Yes (RP#)	No
Intel 28F002BL-T/B	Yes	don't use (DU)	Yes (RP#)	No
MXIC 28F002T/B	Yes**	WP#	Yes(WP#,RP#)	Yes*

* Successive block load cycle must < 30us

** When RP# = Low in reset module no deep power down current

*** Pin 12



Intel 28F002BX : JP open
MXIC: JP close

1. Icc current during deep power-down mode is 0.20 uA typical for Intel. Icc current during stand by mode is 50 uA typical for Macronix.

2. Pin 12 : is the only different pin configuration between Intel 28F002BX and MXIC 28F002

2-2 Intel 28F002 V.S. MXIC 28F002BX (Boot Block Handling)

	RP#		WP#	
Intel 28F002BV	2.0 ~ 6.5V	- standard read mode	2.0 ~ 6.5V	- boot block unlock
	11.4 ~ 12.6	- unlock	N.A.	N.A.
	-0.5 ~ 0.8V	- output High-Z	-0.5 ~ 0.8V	- boot block locked
Intel 28F002BX	2.0 ~ 6.5V	- boot block locked	N.A.	N.A.
	11.5 ~ 13V	- boot block unlock	N.A.	N.A.
	-0.5 ~ 0.8V	- deep power down - boot block locked	N.A.	N.A.
Intel 28F002BL	2.0 ~ 4.1V	- boot block locked	N.A.	N.A.
	11.5 ~13.0V	- boot block unlock	N.A.	N.A.
	-0.5 ~ 0.8V	- deep power down - boot block locked	N.A.	N.A.
MXIC 28F002	0 ~ 6V	- boot block locked	2.0 ~ 5.0V	boot block unlock
	11.4 ~ 12.6V	- boot block unlock	N.A.	N.A.
	-0.3 ~ 0.8V	- READ : output High-Z - ERASE/PROGRAM : (abort command)	-0.3 ~ 0.8V	- boot block locked

2-3 Intel 28F200 V.S. MXIC 28F2100 (Feature)

	H/W			S/W
	44SOP Pin 44 RP# deep power down	44SOP Pin 2 Function	16k boot block	multi-block erase
Intel 28F200BV-T/B	Yes	WP#	Yes (RP#)	No
Intel 28F200BX-T/B	Yes	don‘ t use (DU)**	Yes (RP#)	No
Intel 28F200BL-T/B	Yes	don‘ t use (DU)**	Yes (RP#)	No
MXIC 28F2100T/B	No(reset when low)	NC	Yes (RP#)	Yes*

* Successive block load cycle must < 30us
** Pin 2 should not be connected to anything

2-4 Intel 28F200 V.S. MXIC 28F2100 (Boot Block Handling)

	RP#		WP#	
Intel 28F200BV	2.0 ~ 6.5V	- standard read mode	2.0 ~ 6.5V	- boot block unlock
	11.4 ~ 12.6	- unlock	N.A.	N.A.
	-0.5 ~ 0.8V	- output High-Z	-0.5 ~ 0.8V	- boot block locked
Intel 28F200BX	2.0 ~ 6.5V	- boot block locked	N.A.	N.A.
	11.5 ~ 13V	- boot block unlock	N.A.	N.A.
	-0.5 ~ 0.8V	- deep power down - boot block locked	N.A.	N.A.
Intel 28F200BL	2.0 ~ 4.1V	- boot block locked	N.A.	N.A.
	11.5 ~13.0V	- boot block unlock	N.A.	N.A.
	-0.5 ~ 0.8V	- deep power down - boot block locked	N.A.	N.A.
MXIC 28F2100T/B	2.0 ~ 12.6V	- not reset - normal operation	2.0 ~ 5.0V	- boot block unlock
	-0.3 ~ 0.8V	- reset - READ : output High-Z - ERASE/PROGRAM : (abort command)	-0.3 ~ 0.8V	- boot block locked

3. Read ID Sequence comparison table

part	cycle 1			cycle 2			cycle 3			cycle 4		
Intel	op	addr	data	op	addr	data	op	addr	data (word)	op	addr	data (word)
1. 28F200BX-T	write	xxxx	90H	read	00	89H	read	01	74H (2274)			
2. 28F200BX-B	write	xxxx	90H	read	00	89H	read	01	75H (2275)			
3. 28F002BX-T	write	xxxx	90H	read	00	89H	read	01	7CH			
4. 28F002BX-B	write	xxxx	90H	read	00	89H	read	01	7DH			
MXIC	op	addr	data	op	addr	data	op	addr	data (word)	op	addr	data (word)
1. 28F2100T	write	xxxx	90H	read	00	C2H	write	xxxx	90H	read	01	2CH (002CH)
2. 28F2100B	write	xxxx	90H	read	00	C2H	write	xxxx	90H	read	01	2BH (002BH)
3. 28F002T	write	xxxx	90H	read	00	C2H	write	xxxx	90H	read	01	2DH
4. 28F002B	write	xxxx	90H	read	00	C2H	write	xxxx	90H	read	01	2EH

Note :

1. For all Intel Parts , after ID code is read , a RESET or Read Array Command must be issued to switch to Read Array Mode .
2. For all MXIC parts , each time after ID code is read , the devices will switch automatically to Read Array Mode .

4. Source Code Lising

```
//---------------------------------------------------------------------------
//    MX28F2100T/B & MX28F002T/B
//    Source code
//    MACRONIX COPYRIGHT © 1997 5-30-1997
//---------------------------------------------------------------------------
//The following source code is for MX28F2100T/B & MX28F002T/B.
//---------------------------------------------------------------------------
// command code
//---------------------------------------------------------------------------
#define F21M_READ_ID_CMD                0x90
#define F21M_PROGRAM_CMD                0x40
#define F21M_CHIP_ERASE_CMD1            0x30
#define F21M_CHIP_ERASE_CMD2            0x30
#define F21M_BLOCK_ERASE_CMD1           0x20
#define F21M_BLOCK_ERASE_CMD2           0xD0
#define F21M_RESET_CMD                  0xFF
#define F21M_ERASE_SUSPEND_CMD          0xB0
#define F21M_ERASE_RESUME_CMD           0xD0
#define F21M_READ_STATUS_CMD            0x70
#define F21M_CLEAR_STATUS_CMD           0x50
#define F21M_SUSPEND_CMD 0xB0
#define F21M_RESUME_CMD                 0xD0
//---------------------------------------------------------------------------
// ID code
//---------------------------------------------------------------------------
#define ID_MX28F2100B          0x2B
#define ID_MX28F2100T          0x2C
#define ID_MX28F002T           0x2D
#define ID_MX28F002B           0x2E
//---------------------------------------------------------------------------
// Function: Read manufacture ID and device ID
//Description : Read manufacture ID when A0=0, device ID when
//                    A0=1.
// Input : none
// Output: none
//---------------------------------------------------------------------------
f21m_Read_ID()
{
 unsigned char    devID,mfrID;

//write read ID command 90H, address is don't care.
 fmemptr[0]=F21M_READ_ID_CMD;
// set A0=1, read manufacture ID code
 mfrID=fmemptr[0];

//write read ID command 90H, address is don't care.
 fmemptr[0]=F21M_READ_ID_CMD;
// set A0=1 , get device ID code ( in our eval. Kit, [2] be read)
 devID=fmemptr[2];                    // read [1], device code
}
//---------------------------------------------------------------------------
// Function: Chip erase
// Description : Erase whole chip, no further verify needed.
// Input: none
// Output: message
//---------------------------------------------------------------------------
char *f21m_Chip_Erase()
{
   fmemptr[0]=F21M_CHIP_ERASE_CMD1;     // write xxxx,30h
   fmemptr[0]=F21M_CHIP_ERASE_CMD2;     // write xxxx,30h

   f21m_DQ7_Polling();                  // loop until sr.7=1
 if( fmemptr[0] & 0x20)
// DQ5 polling, DQ5=0 succeed
    {
      sprintf(message,"Chip Erase Failed");
      // Erase fail, clear S.R. at first
      f21m_Clear_Status();
     return message;
    }
 sprintf(message,"Chip Erase Successfully Completed");
 return message;
}
//---------------------------------------------------------------------------
// Function: Polling data bus bit 7
// Description : When program or erase command is executed, the
//                    bit7 of data bus indicates whether the chip is busy
//                    ready for next operation.
// Input: none
// Output: none
//---------------------------------------------------------------------------
f21m_DQ7_Polling(void)
{
 unsigned char data;
```

```
 do
   {
//the state machine of flash will stay at read status mode
 //when the chip is programmed or erase. You can read
 // status register in any address.
     data=fmemptr[0];                          // read status
     } while(!( data & 0x80 ));
 return(0);
}
//---------------------------------------------------------------------------
// Function: polling erase
// Description : while SR.7=1 pr user press any key will stop loop
//                This function can test erase/suspend operation
// Input: none
// Output: 0    = normal return
//          -2    = suspend return
// Comment : If you don't want to suspend the erase operation, you
//                can use the "f21m_DQ7_Polling()" function.
//---------------------------------------------------------------------------
f21m_DQ7_Polling_erase(void)
{
 unsigned char data;
 do
   {
    /the state machine of flash will stay at read status mode
    //when the chip is programmed or erase. You can read
    // status register in any address.
    data=fmemptr[0];
     //In our system, any keyboard press presents suspend command
     //when the erase operation is on going, Your can redefine you
way
      //in your system.
    if( kbhit() ) {getch(); return -2;}     // if key press go to suspend
flow
    } while(!( data & 0x80 ));
 return(0);
}
//---------------------------------------------------------------------------
// Function: suspend
// Description : The erasing process can be stopped by this command
// Input: none
// Output: none
//---------------------------------------------------------------------------
f21m_Suspend()
{
 //write suspend command B0H, the address is don't care.
 fmemptr[0]=F21M_SUSPEND_CMD;
 //loop until SR.7 =1
 while(!(f21m_Read_Status()&0x80));
}
//---------------------------------------------
// Function : Resume the erase command, when the erase operation is
//            suspend
// Input : none
// Output : none
//---------------------------------------------
f21m_Resume()
{
  //write resume command D0H, address is don't care
  fmemptr[0]=F21M_RESUME_CMD;
}
//---------------------------------------------------------------------------
// Function: Block erase
// Input: block_rease_address indicate which block will be erased
// Output : 0 = success
//          -1 = block erase error
//          -2 = suspend/resume error
//Comment : In this function, multiple block erase is allowed. The
//          flow of this command should be as follows:
//          Write [xxxx], 60H
//          Write [first_erase_block_address], 60H
//                        |
//                        |
//          Write [Last_erase_block_address], 60H
//           time_out_100us()
//           polling();
//---------------------------------------------------------------------------
f21m_Block_Erase (long Block_Erase_Address)
{
  //write block erase set up command 60H, the address is don't care
  fmemptr[0]=F21M_BLOCK_ERASE_CMD1;
  fmemptr[Block_Erase_Address]=F21M_BLOCK_ERASE_CMD2;
  //delay 100us, this function is dependent on system.
  delay_100us(1);                  // delay 100us
  //when f21m_DQ7_Polling_erase return -2, then go to suspend flow
    if(f21m_DQ7_Polling_erase()==-2)
    {
    printf("Erase Suspend ..... press any key to Resume");
    f21m_Suspend();                          // suspend command
     while(!kbhit());                        // loop until key
```

```
press
      // if SR.6=1, resume erase
     if( f21m_Read_Status()&0x40)
       {
        printf("Erase Resume.");
        f21m_Resume();                          // resume command
       }
      else
     // if SR.6 =0, erase completed, check SR3,4,5, to see whether
     //erase successfully
      {
       f21m_Read_Status();
      if(st.status.d5==1) sprintf(message,"Block Erase Fail !!");
      else if(st.status.d4==1) sprintf(message,"failed to program !!");
      else if(st.status.d3==1) sprintf(message,"Operation Abort !!(Vpp
Low)");
     }
     Read_Status();
     show_status(woutput);
     return -2;
     }
  if ( fmemptr[Block_Erase_Address] & 0x20 )     // check erase result
     {
      sprintf(message,"Block Erase Fail !!");
      return(-1);
     }
  f21m_Clear_Status();
  sprintf(message,"Block Erase Successfully Completed !!");
  return(0);
}
//-------------------------------------------------------------------------------
// Function: reset the state machine of flash
// Input: none
// Output: none
//-------------------------------------------------------------------------------
f21m_ResetSTM(void)
{
  fmemptr[0]=F21M_RESET_CMD;
  fmemptr[0]=F21M_RESET_CMD;
  return(0);
}
//-------------------------------------------------------------------------------
// Function: read byte from array
// Input: byte_address
// Output: byte data
//-------------------------------------------------
unsigned char f21m_Read_Byte (long byte_address)
{

  // reset the state machine to make sure it is at the normal mode(read
  // mode)
  f21m_ResetSTM();
  return(fmemptr[byte_address]);                  // return byte data
}
//-------------------------------------------------------------------------------
// Function: Suspend the chip erase or block erase operation
// Description : If this command were issued as the erase operation is
on
//                going, the erase operation will stop until resume
//                command be issued.
// Input: none
// Output: none
//-------------------------------------------------------------------------------
f21m_Erase_Suspend()
{
 //write command B0H, address is don't care
 fmemptr[0]=F21M_ERASE_SUSPEND_CMD;
 return 0;
}
//-------------------------------------------------------------------------------
// Function: Resume the erase operation
// Description Continue the erase operation when the erase suspend
//                command interrupt the erase process
// Input: none
// Output: none
//-------------------------------------------------------------------------------
f21m_Erase_Resume()
{
 //write command D0H, address is don't care
 fmemptr[0]=F21M_ERASE_RESUME_CMD;
 return 0;
}
//-------------------------------------------------------------------------------
// Function: Read status register content
// Input: none
// Output: none
//-------------------------------------------------------------------------------
int f21m_Read_Status()
{
  // write command 70H, the address is don't care
```

```
 fmemptr[0]=F21M_READ_STATUS_CMD;
 return (fmemptr[0]);
}
//------------------------------------------------------------------------------
// Function: Clear the status register content
//Description : This function is used to clear the status register.
//                When the program or erase operations fail, the
//                state machine will stop at the read status mode,
//                this command must be issued to return to the
//                normal mode.
// Input: none
// Output: none
//------------------------------------------------------------------------------
f21m_Clear_Status()
{
 //write command 50H, address is don't care
 fmemptr[0]=F21M_CLEAR_STATUS_CMD;
 return 0;
}
```