Add support for a new flash device

IN THIS DOCUMENT

To support a new flash device, a configuration file must be written that describes the device characteristics, such as page size, number of pages and commands for reading, writing and erasing data. This information can be found in the datasheet for the flash device. Many devices available in the market can be described using these configuration parameters; those that cannot are unsupported.

The configuration file for the Numonyx M25P10-A[1] is shown below. The device is described as an initializer for a C structure, the values of which are described in the following sections.

[1] http://www.xmos.com/references/m25p10a

Publication Date: 2013/11/11 REV E

```
10,                     /* 1.  libflash device ID */
256,                    /* 2.  Page size */
512,                    /* 3.  Number of pages */
3,                      /* 4.  Address size */
4,                      /* 5.  Clock divider */
0x9f,                   /* 6.  RDID cmd */
0,                      /* 7.  RDID dummy bytes */
3,                      /* 8.  RDID data size in bytes */
0x202011,               /* 9.  RDID manufacturer ID */
0xD8,                   /* 10. SE cmd */
0,                      /* 11. SE full sector erase */
0x06,                   /* 12. WREN cmd */
0x04,                   /* 13. WRDI cmd */
PROT_TYPE_SR,           /* 14. Protection type */
{{0x0c,0x0},{0,0}},     /* 15. SR protect and unprotect cmds */
0x02,                   /* 16. PP cmd */
0x0b,                   /* 17. READ cmd */
1,                      /* 18. READ dummy bytes*/
SECTOR_LAYOUT_REGULAR,  /* 19. Sector layout */
{32768,{0,{0}}},        /* 20. Sector sizes */
0x05,                   /* 21. RDSR cmd*/
0x01,                   /* 22. WRSR cmd */
0x01,                   /* 23. WIP bit mask */
```

1 Libflash Device ID

```
10, /* 1. libflash device ID */
```

This value is returned by libflash on a call to the function `fl_getFlashType` so that the application can identify the connected flash device.

2 Page Size and Number of Pages

```
10,                     /* 1.  libflash device ID */
256,                    /* 2.  Page size */
```

These values specify the size of each page in bytes and the total number of pages across all available sectors. On the M25P10-A datasheet, these can be found from the following paragraph on page 6:

> *The memory is organized as 4 sectors, each containing 128 pages. Each page is 256 bytes wide. Thus, the whole memory can be viewed as consisting of 512 pages, or 131,072 bytes.*

3 Address Size

```
3,                          /* 4.  Address size */
```

This value specifies the number of bytes used to represent an address. Figure 1 reproduces the part of the M25P10-A datasheet that provides this information. In the table, all instructions that require an address take three bytes.

Instruction	Description	One-byte instruction code		Address byte	Dummy bytes	Data bytes
WREN	Write Enable	0000 0110	06h	0	0	0
WRDI	Write Disable	0000 0100	04h	0	0	0
RDID	Read	1001 1111	9Fh	0	0	1 to 3
RDSR	Read Status Register	0000 0101	05h	0	0	1 to ∞
WRSR	Write Status Register	0000 0001	01h	0	0	1
READ	Read Data Bytes	0000 0011	03h	3	0	1 to ∞
FAST_READ	Read Data Bytes at Higher Speed	0000 1011	0Bh	3	1	1 to ∞
PP	Page Program	0000 0010	02h	3	0	1 to 256
SE	Sector Erase	1101 1000	D8h	3	0	0
BE	Bulk Erase	1100 0111	C7h	0	0	0
DP	Deep Power-down	1011 1001	B9h	0	0	0
RES	Release from Deep Power-down, and Read Electronic Signature	1010 1011	ABh	0	3	1 to ∞
	Release from Deep Power-down			0	0	0

Figure 1: Table 4 on page 17 of M25P10-A datasheet

4 Clock Rate

```
4,                          /* 5.  Clock divider */
```

This value is used to determine the clock rate for interfacing with the SPI device. For a value of *n*, the SPI clock rate used is 100/2*n* MHz. libflash supports a maximum of 12.5MHz.

Figure 2 reproduces the part of the M25P10-A datasheet that provides this information. The AC characteristics table shows that all instructions used in the configuration file, as discussed throughout this document, can operate at up to 25MHz. This is faster than libflash can support, so the value 4 is provided to generate a 12.5MHz clock.

In general, if the SPI device supports different clock rates for different commands used by libflash, the lowest value must be specified.

Symbol	Alt.	Parameter	Min	Typ	Max	Unit
f_C	f_C	Clock frequency for the following instructions: FAST_READ, PP, SE, BE, DP, RES, WREN, WRDI, RDSR, WRSR	D.C.		25	MHz
f_R		Clock frequency for READ instructions	D.C.		20	MHz
t_{CH}	t_{CLH}	Clock High time	18			ns
t_{CL}	t_{CLL}	Clock Low time	18			ns

Figure 2: Table 18 on page 40 of M25P10-A datasheet (first four entries only).

5 Read Device ID

```
0x9f,                   /* 6.  RDID cmd */
0,                      /* 7.  RDID dummy bytes */
3,                      /* 8.  RDID data size in bytes */
0x202011,               /* 9.  RDID manufacturer ID */
```

Most flash devices have a hardware identifier that can be used to identify the device. This is used by libflash when one or more flash devices are supported by the application to determine which type of device is connected. The sequence for reading a device ID is typically to issue an RDID (read ID) command, wait for zero or more dummy bytes, and then read one or more bytes of data.

Figure 1 reproduces the part of the M25P10-A datasheet that provides this information. The row for the instruction RDID shows that the command value is `0x9f`, that there are no dummy bytes, and one to three data bytes. As shown in Figure 3 and Figure 4, the amount of data read depends on whether just the manufacturer ID (first byte) is required, or whether both the manufacturer ID and the device ID (second and third bytes) are required. All three bytes are needed to uniquely identify the device, so the manufacturer ID is specified as the three-byte value `0x202011`.

Manufacturer identification	Device identification	
	Memory type	Memory capacity
20h	20h	11h

Figure 3: Table 5 on page 19 of M25P10-A datasheet

In general, if there is a choice of RDID commands then the JEDEC compliant one should be preferred. Otherwise, the one returning the longest ID should be used.

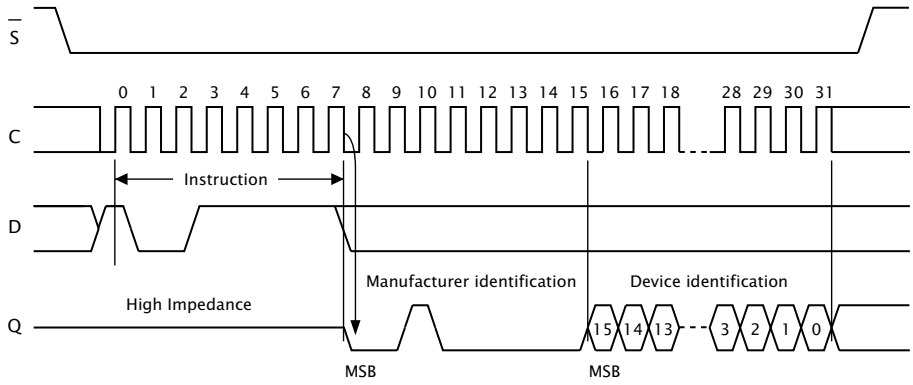

Figure 4: Figure 9 on page 19 of M25P10-A datasheet

6 Sector Erase

```
0xD8,                   /* 10. SE cmd */
0,                      /* 11. SE full sector erase */
```

Most flash devices provide an instruction to erase all or part of a sector.

Figure 1 reproduces the part of the M25P10-A datasheet that provides this information. The row for the instruction SE shows that the command value is `0xd8`. On the M25P10-A datasheet, the amount of data erased can be found from the first paragraph on page 28:

> *The Sector Erase (SE) instruction sets to '1' (FFh) all bits inside the chosen sector.*

In this example the SE command erases all of the sector, so the SE data value is set to 0. If the number of bytes erased is less than a full sector, this value should be set to the number of bytes erased.

7 Write Enable/Disable

```
0x06,                   /* 12. WREN cmd */
0x04,                   /* 13. WRDI cmd */
```

Most flash devices provide instructions to enable and disable writes to memory. Figure 1 reproduces the part of the M25P10-A datasheet that provides this information. The row for the instruction WREN shows that the command value is `0x06`, and the row for the instruction WRDI shows that the command value is `0x04`.

8 Memory Protection

```
PROT_TYPE_SR,            /* 14. Protection type */
{{0x0c,0x0},{0,0}},      /* 15. SR protect and unprotect cmds */
```

Some flash devices provide additional protection of sectors when writes are enabled. For devices that support this capability, libflash attempts to protect the flash image from being accidentally corrupted by the application. The supported values for *protection* type are:

`PROT_TYPE_NONE`
: The device does not provide protection

`PROT_TYPE_SR`
: The device provides protection by writing the status register

`PROT_TYPE_SECS`
: The device provides commands to protect individual sectors

The protection details are specified as part of a construction of the form:

```
{{a,b},{c,d}}
```

If the device does not provide protection, all values should be set to 0. If the device provides SR protection, *a* and *b* should be set to the values to write to the SR to protect and unprotect the device, and *c* and *d* to 0. Otherwise, *c* and *d* should be set to the values to write to commands to protect and unprotect the device, and *a* and *b* to 0.

Figure 5 and Figure 6 reproduce the parts of the M25P10-A datasheet that provide this information. The first table shows that BP0 and BP1 of the status register should be set to 1 to protect all sectors, and both to 0 to disable protection. The second table shows that these are bits 2 and 3 of the SR.

Figure 5: Table 2 on page 13 of M25P10-A datasheet

Status Register Content		Memory content	
BP1 bit	**BP0 bit**	**Protected area**	**Unprotected area**
0	0	none	All sectors (four sectors: 0, 1, 2 and 3)
0	1	Upper quarter (sector 3)	Lower three-quarters (three sectors: 0 to 2)
1	0	Upper half (two sectors: 2 and 3)	Lower half (sectors 0 and 1)
1	1	All sectors (four sectors: 0, 1, 2 and 3)	none

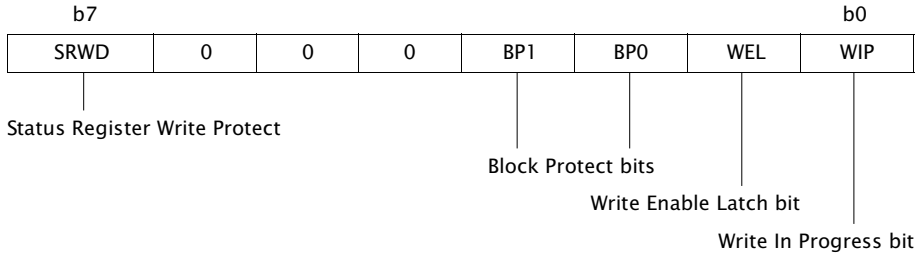

Figure 6: Table 6 on page 20 of M25P10-A datasheet

9 Programming Command

```
0x02,                    /* 16. PP cmd */
```

Devices are programmed either a page at a time or a small number of bytes at a time. If page programming is available it should be used, as it minimizes the amount of data transmitted over the SPI interface.

Figure 1 reproduces the part of the M25P10-A datasheet that provides this information. In the table, a page program command is provided and has the value 0x02.

If page programming is not supported, this value is a concatenation of three separate values. Bits 0..7 must be set to 0. Bits 8..15 should contain the program command. Bits 16..23 should contain the number of bytes per command. The libflash library requires that the first program command accepts a three byte address but subsequent program command use auto address increment (AAI).

An example of a device without a PP command is the ESMT F25L004A[2]. Figure 7 reproduces the part of the F25L004A datasheet that provides this information. In the timing diagram, the AAI command has a value 0xad, followed by a three-byte address and two bytes of data.

Figure 7: Table 7 on page 12 of F25L004A datasheet.

Symbol	Parameter	Minimum	Units
$T_{PU-READ}$	V_{DD} Min to Read Operation	10	µs
$T_{PU-WRITE}$	V_{DD} Min to Write Operation	10	µs

The corresponding entry in the specification file is:

```
0x00|(0xad<<8)|(2<<16), /* No PP, have AAI for 2 bytes */
```

[2] http://www.xmos.com/references/f25l004

10 Read Data

```
0x0b,                       /* 17. READ cmd */
1,                          /* 18. READ dummy bytes*/
```

The sequence for reading data from a device is typically to issue a READ command, wait for zero or more dummy bytes, and then read one or more bytes of data.

Figure 1 reproduces the part of the M25P10-A datasheet that provides this information. There are two commands that can be used to read data: READ and FAST_READ. The row for the instruction FAST_READ shows that the command value is 0x0b, followed by one dummy byte.

11 Sector Information

```
SECTOR_LAYOUT_REGULAR,  /* 19. Sector layout */
{32768,{0,{0}}},        /* 20. Sector sizes */
```

The first value specifies whether all sectors are the same size. The supported values are:

SECTOR_LAYOUT_REGULAR
: The sectors all have the same size

SECTOR_LAYOUT_IRREGULAR
: The sectors have different sizes

On the M25P10-A datasheet, this can be found from the following paragraph on page 15:

> *The memory is organized as:*
>
> - 131,072 bytes (8 bits each)
> - 4 sectors (256 Kbits, 32768 bytes each)
> - 512 pages (256 bytes each).

The sector sizes is specified as part of a construction: {a, {b, {c}}}. For regular sector sizes, the size is specified in *a*. The values of *b* and *c* should be 0.

For irregular sector sizes, the size number of sectors is specified in *b*. The log base 2 of the number of pages in each sector is specified in c. The value of *a* should be 0. An example of a device with irregular sectors is the AMIC A25L80P[3]. Figure 8 reproduces the part of this datasheet that provides the sector information.

[3] http://www.xmos.com/references/a25l80p

Sector	Sector Size (Kb)	Address Range	
15	64	F0000h	FFFFFh
14	64	E0000h	EFFFFh
13	64	D0000h	DFFFFh
12	64	C0000h	CFFFFh
11	64	B0000h	BFFFFh
10	64	A0000h	AFFFFh
9	64	90000h	9FFFFh
8	64	80000h	8FFFFh
7	64	70000h	7FFFFh
6	64	60000h	6FFFFh
5	64	50000h	5FFFFh
4	64	40000h	4FFFFh
3	64	30000h	3FFFFh
2	64	20000h	2FFFFh
1	64	10000h	1FFFFh
0-4	32	08000h	0FFFFh
0-3	16	04000h	07FFFh
0-2	8	02000h	03FFFh
0-1	4	01000h	01FFFh
0-0	4	00000h	00FFFh

Figure 8: Table 2 on page 7 of A25L80P datasheet

The corresponding entry in the specification file is:

```
SECTOR_LAYOUT_IRREGULAR,
    {0,{20,{4,4,5,6,7,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8}}},
```

12 Status Register Bits

```
0x05,                   /* 21. RDSR cmd*/
0x01,                   /* 22. WRSR cmd */
0x01,                   /* 23. WIP bit mask */
```

Most flash devices provide instructions to read and write a status register, including a write-in-progress bit mask.

Figure 1 reproduces the part of the M25P10-A datasheet that documents the RDSR and WRSR commands. The diagram in Figure 6 shows that the WIP bit is in bit position 0 of the SR, resulting in a bit mask of `0x01`.

13 Add Support to xTimeComposer

A configuration file can be used with libflash or xflash. The example program below uses libflash to connect to a M25P10-A device, the configuration parameters which are specified in `m25p10a`.

```
#include "platform.h"
#include "flash.h"
#include "flashlib.h"
#include "stdio.h"
#include "stdlib.h"

fl_PortHolderStruct SPI = {PORT_SPI_MISO,
                           PORT_SPI_SS,
                           PORT_SPI_CLK,
                           PORT_SPI_MOSI,
                           XS1_CLKBLK_1};

fl_DeviceSpec myFlashDevices[] = {
 {
   #include "m25p10a"
 }
};

int flash_access() {
  if (fl_connectToDevice(SPI, myFlashDevices,
      sizeof(myFlashDevices)/sizeof(fl_DeviceSpec)) != 0) {
    printf("No supported flash devices found.\n"); exit(1);
  } else {
  printf("Found custom flash device m25p10a.\n"); exit(0);
  }
  return 0;
}

int main() {
  // multicore main is required for xscope
  par {
    on stdcore[0] : flash_access();
  }
}
```

The custom flash device must be specified in the XN file as follows:

```
<ExternalDevices>
  <Device NodeId="0" Tile="0" Name="bootFlash"
          Class="SPIFlash" Type="M25P10A">
    <Attribute Name="PORT_SPI_MISO" Value="PORT_SPI_MISO" />
    <Attribute Name="PORT_SPI_SS" Value="PORT_SPI_SS" />
    <Attribute Name="PORT_SPI_CLK" Value="PORT_SPI_CLK" />
    <Attribute Name="PORT_SPI_MOSI" Value="PORT_SPI_MOSI" />
  </Device>
</ExternalDevices>
```

To compile an image file that links to the lib flash library, start the command-line tools (see XM-000950-PC) and enter the following command:

```
xcc main.xc -o prog.xe -target=target_with_custom_flash.xn -lflash
```

To generate an image file in the xCORE flash format, which can be subsequently programmed into the above flash device, enter the following command:

```
xflash prog.xe -o imgfile --spi-spec m25p10a
```

XFLASH generates an image for the custom flash device, which it writes to the specified image file.

14 Select a Flash Device

When selecting a flash device for use with an xCORE device, the following guidelines are recommended:

- If access to the data partition is required, select a device with fine-grained erase granularity, as this will minimize the gaps between the factory and upgrade images, and will also minimize the amount of data that libflash needs to buffer when writing data.
- Select a device with sector protection if possible, to ensure that the bootloader and factory image are protected from accidental corruption post-deployment.
- Select a flash speed grade suitable for the application. Boot times are minimal even at low speeds.