The FlexSPI configuration is defned by the boot object in the SPI code located at the start of the SPI flash.
For example the i.MX RT 1020 EVK has the following boot object for its IS25LP064A QSPi flash part:

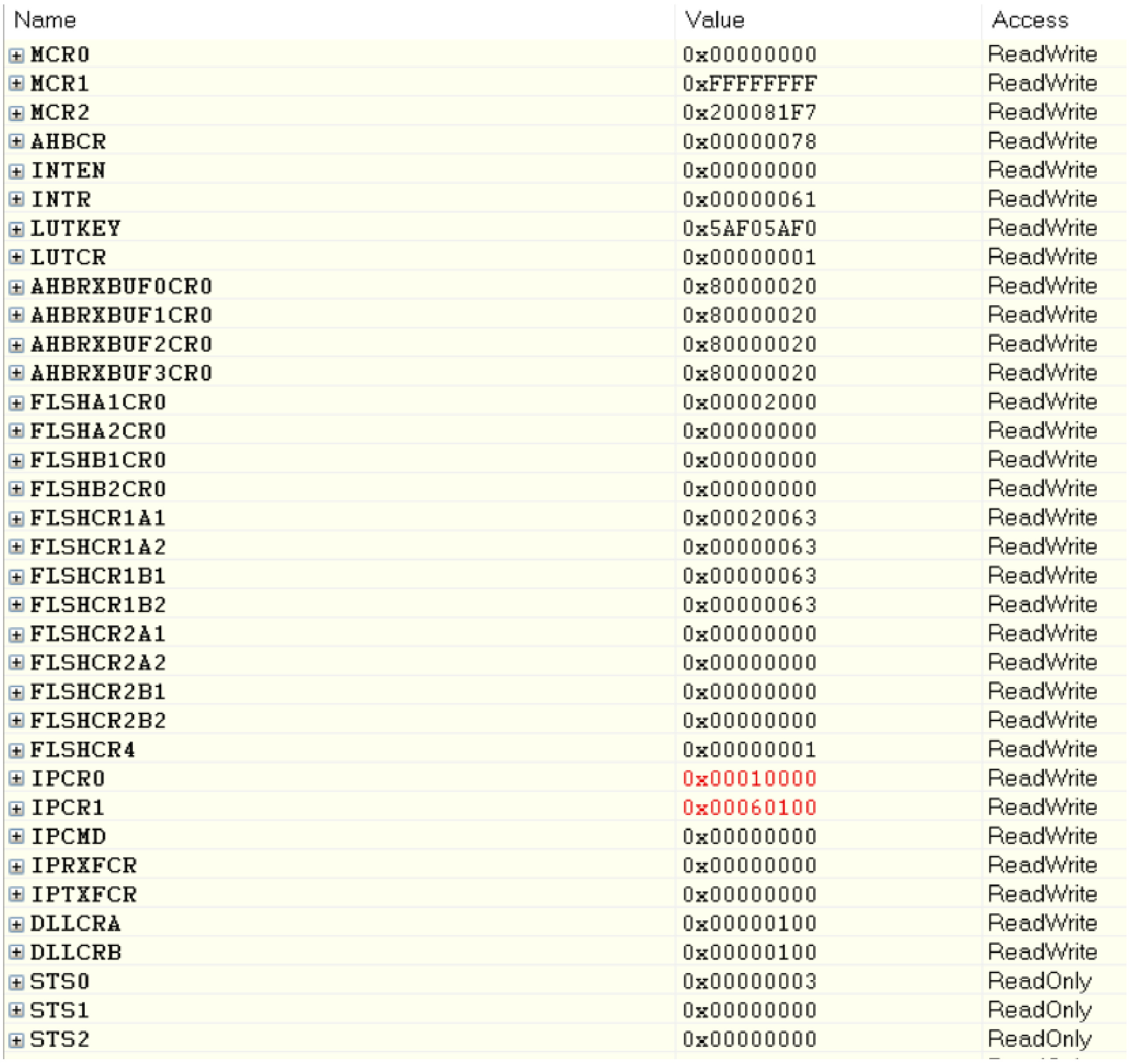
```
const FLEXSPI_NOR_BOOT_CONFIGURATION __boot_config = {
    .memConfig = {
        .tag              = FLEXSPI_CFG_BLK_TAG,
        .version          = FLEXSPI_CFG_BLK_VERSION,
        .readSampleClkSrc =
kFlexSPIReadSampleClk_LoopbackFromDqsPad,
        .csHoldTime       = 3,
        .csSetupTime      = 3,
        // Enable DDR mode, word addressable, safe configuration,
differential clock
        //
        .deviceType       = kSerialNOR,
        .sflashPadType    = kSerialFlash_4Pads,
        .serialClkFreq    = kFlexSpiSerialClk_30MHz,
        .sflashA1Size     = FLEXSPI_FLASH_SIZE,
        .lookupTable = {
            // Read LUT
            //
            FLEXSPI_LUT_1PAD_SEQ_1(FLEXSPI_Command_SDR,
FAST_READ_QUAD_I_O) |
FLEXSPI_LUT_4PAD_SEQ_2(FLEXSPI_Command_RADDR_SDR,
ADDRESS_LENGTH_24BITS),
            FLEXSPI_LUT_4PAD_SEQ_1(FLEXSPI_Command_DUMMY_SDR,
0x06) | FLEXSPI_LUT_4PAD_SEQ_2(FLEXSPI_Command_READ_SDR, 0x04),
        },
    },
    .pageSize              = SPI_FLASH_PAGE_LENGTH,
    .sectorSize            = SPI_FLASH_SUB_SECTOR_LENGTH,

    .blockSize             = SPI_FLASH_BLOCK_LENGTH,
    .isUniformBlockSize = 0,
};
```

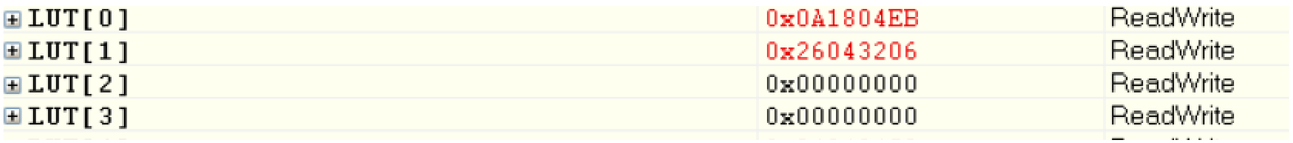
The cnfiguration is usually supplied by the SPI flash manufacturer, configuring optially for the device in hand.

The ROM loader and configures the FlexSPI (pins, registers and look-up table) based on the information read here so that it can execute code directly from the SPI flash.

The look-up table entry defines (at least) the command to beb used to read from the SPI flash, which is performed in quad mode for efficiency.
This is the setup resulting in the FlexSPI:



| Name | Value | Access |
|---|---|---|
| MCR0 | 0x00000000 | ReadWrite |
| MCR1 | 0xFFFFFFFF | ReadWrite |
| MCR2 | 0x200081F7 | ReadWrite |
| AHBCR | 0x00000078 | ReadWrite |
| INTEN | 0x00000000 | ReadWrite |
| INTR | 0x00000061 | ReadWrite |
| LUTKEY | 0x5AF05AF0 | ReadWrite |
| LUTCR | 0x00000001 | ReadWrite |
| AHBRXBUF0CR0 | 0x80000020 | ReadWrite |
| AHBRXBUF1CR0 | 0x80000020 | ReadWrite |
| AHBRXBUF2CR0 | 0x80000020 | ReadWrite |
| AHBRXBUF3CR0 | 0x80000020 | ReadWrite |
| FLSHA1CR0 | 0x00002000 | ReadWrite |
| FLSHA2CR0 | 0x00000000 | ReadWrite |
| FLSHB1CR0 | 0x00000000 | ReadWrite |
| FLSHB2CR0 | 0x00000000 | ReadWrite |
| FLSHCR1A1 | 0x00020063 | ReadWrite |
| FLSHCR1A2 | 0x00000063 | ReadWrite |
| FLSHCR1B1 | 0x00000063 | ReadWrite |
| FLSHCR1B2 | 0x00000063 | ReadWrite |
| FLSHCR2A1 | 0x00000000 | ReadWrite |
| FLSHCR2A2 | 0x00000000 | ReadWrite |
| FLSHCR2B1 | 0x00000000 | ReadWrite |
| FLSHCR2B2 | 0x00000000 | ReadWrite |
| FLSHCR4 | 0x00000001 | ReadWrite |
| IPCR0 | 0x00010000 | ReadWrite |
| IPCR1 | 0x00060100 | ReadWrite |
| IPCMD | 0x00000000 | ReadWrite |
| IPRXFCR | 0x00000000 | ReadWrite |
| IPTXFCR | 0x00000000 | ReadWrite |
| DLLCRA | 0x00000100 | ReadWrite |
| DLLCRB | 0x00000100 | ReadWrite |
| STS0 | 0x00000003 | ReadOnly |
| STS1 | 0x00000000 | ReadOnly |
| STS2 | 0x00000000 | ReadOnly |



| | | |
|---|---|---|
| LUT[0] | 0x0A1804EB | ReadWrite |
| LUT[1] | 0x26043206 | ReadWrite |
| LUT[2] | 0x00000000 | ReadWrite |
| LUT[3] | 0x00000000 | ReadWrite |

whereby also the FlexSPI QSPI mode pins are configured.

Both serial loader and application run from RAM and include SPI flash drivers. The driver will match the SPI flash chip that the board uses and allow the programs to read, write and delete data in the SPI flash using standard interface routines (these allow updaing applications in SPI fash, operating uParameter and uFileSystem or FAT in the SPI memory area).
The following explains how the Flash loaders operate, based on the example of the i.MX RT 1062 OEM board from Embedded Artists, which uses the ATXP032 EcoXip High Performance Low-Power Octal Flash:
- The “BM” loader supplies the manufacturer's boot configuration so that the SPI flash based code and operate.
- The serial loader or application then call the ATXP032 SPI Flash driver initialisation `fnCheckATXPXXX`
which enters the FlexSPI loou up table that will be required by the application's use fo the device. It requests the ID of the chip to verify that it an be correctly detected and optionalyl sets write protection on the area of the SPI flash that is used by the “B” and serial loader so that run away application code could not inadvertently cause corruption to their code source.

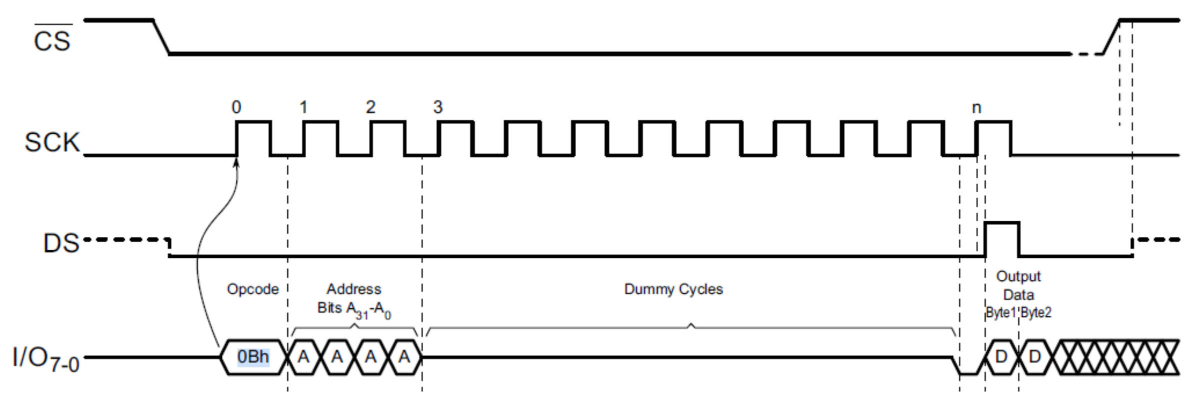
Notes: `fnInitSpiFlash() repeats the low level FlexSPI initialisaton that the ROM loader would have done and is therefore theoretically superfluose. It is however left in the code as reference or as additional means of adjusting these if ever required.`

The look-up tabel is located in the file `spi_flash_ATXP.h`, along with defines for the SPI flash's command set which are shared by the boot configuration. The command interface routine `fnSPI_command_ATXP()` is used to convert between the generic flash driver's commadns (to read, write, delete etc.) and the SPI flash's own specific command set.

An example of entries in the look up table are:

```
    {   // Fast read octal mode - SDR
        NOR_CMD_LUT_SEQ_IDX_READ_FAST_OCTAL,
        {
        FLEXSPI_LUT_8PAD_SEQ_1(FLEXSPI_Command_SDR, FAST_READ) |
FLEXSPI_LUT_8PAD_SEQ_2(FLEXSPI_Command_RADDR_DDR, ADDRESS_LENGTH_32BITS),
        FLEXSPI_LUT_8PAD_SEQ_1(FLEXSPI_Command_DUMMY_DDR, (ECOXIP_READ_NON_SPI_DUMMY_CYCLES
* 2 + 1)) | FLEXSPI_LUT_8PAD_SEQ_2(FLEXSPI_Command_READ_DDR, 0x80),
        },
    },
```

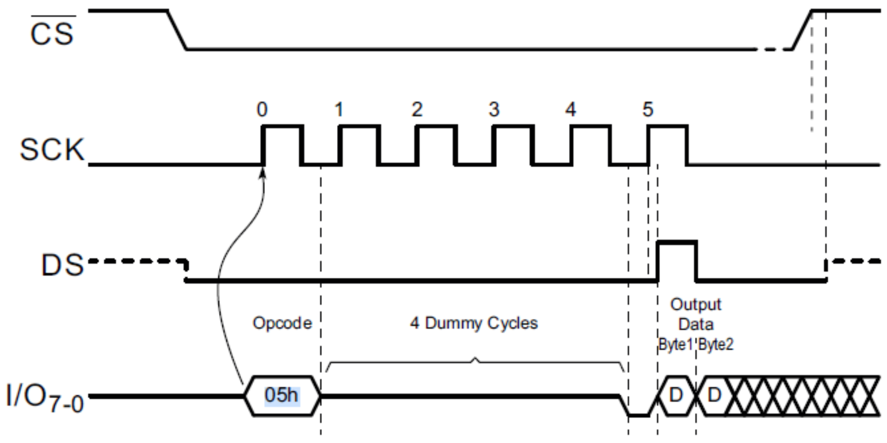
**Read Array — 0Bh Command in Octal Mode DDR**



It is seen that the lookup table entry describes the command operation, starting with sending a FAST_READ command (called Read Array by the chip's manufacturer), followed by 4 bytes to address the data to be read, and a certain number of dummy cycles (these are defined in the device's data sheet and recommended by the manufacturer's boot configuration) and finally continuous reads of the data. Lthou 128 bytes reads are specified in the look-up table the actual FlexSPI operation allows the length to be specified at each use, thus overwriting this setting. All operations are in octal mode, using all 8 data bits of the device.

```
    {   // Read status register
        NOR_CMD_LUT_SEQ_IDX_READSTATUSREG,
        {
        FLEXSPI_LUT_8PAD_SEQ_1(FLEXSPI_Command_SDR, READ_STATUS_REGISTER_1) |
FLEXSPI_LUT_8PAD_SEQ_2(FLEXSPI_Command_DUMMY_DDR, 0x08),
        FLEXSPI_LUT_8PAD_SEQ_1(FLEXSPI_Command_READ_DDR, 1) |
FLEXSPI_LUT_1PAD_SEQ_2(FLEXSPI_Command_STOP, 0x08),
        },
    },
```

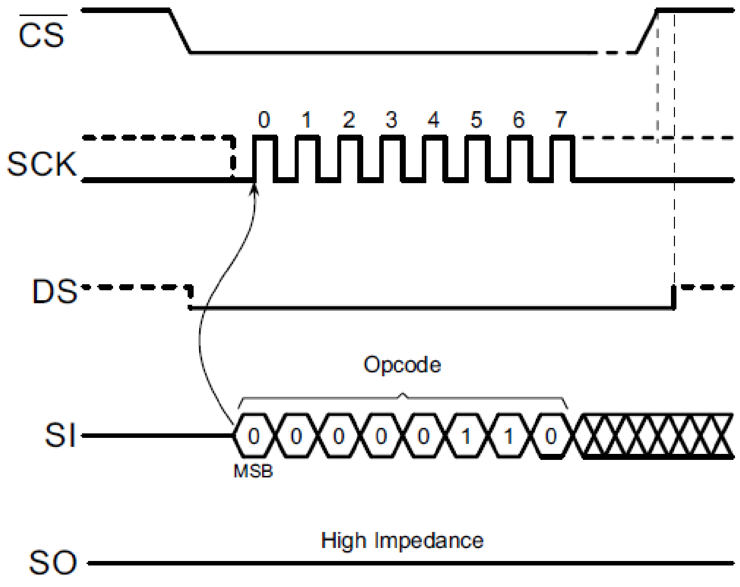
**Read Status Register Byte 1 in Octal Mode — DDR**



This look-up table entry specifies a status register read in octal mode, starting with a write of the read stats command (0x05), 4 dummy cycles, the read of a single bytes and then termination.

```
    {   // Write enable
        NOR_CMD_LUT_SEQ_IDX_WRITEENABLE,
        {
        FLEXSPI_LUT_1PAD_SEQ_1(FLEXSPI_Command_SDR, WRITE_ENABLE) |
FLEXSPI_LUT_1PAD_SEQ_2(FLEXSPI_Command_STOP, 0x00),
        0,
        },
    },
```

## Write Enable



Although the write enable command can be sent in octal mode the look-up table entry uses a single data line method. It specifies the write of the command byte 0x06 and then termination.

It can be seen from these examples that the look-up table entries can be constructed fairly easily based on the timing sequences specified in the data chip's sheet.