

FlashGenie 3.3 V SD-Card Adapter

To enable accessing SD flash memory cards with any SPI-capable microprocessor, this small circuit board provides the SD card connector, and for additional versatility, a real time clock chip and a holder for a 3 V Lithium battery is installed on the reverse side, to provide a permanent calendar when the mcu is powered down.

Communication with the SD card is done with the SPI protocol and requires four data lines:

CS : Chip Select

CLK : Data Clock

MOSI : master out - slave in

MISO : master in - slave out

For a detailed description of the SPI protocol and the command codes needed to get the SD card talking see the SanDisk reference manual at <http://www.sandisk.com/Oem/Manuals>

The SD connector is of the “push-push” type: to remove the SD card, don’t pull it out but push it in.

There are two more optional connectors for detecting card insertion and sensing the position of the write-protect switch (both have pull-up resistors and are active-low, the write-protect contact is on a separate solder jumper as shown below), and one data line that toggles two leds for giving user feedback under MCU control (a high signal lights one led, a low signal lights the other led, a floating line lights both leds dimmly).

Communication with the DSI340 clock chip is accomplished via the I2C protocol using two lines:

SDA - serial data

SCL - serial clock

See the DSI340 datasheet for the command code definitions:

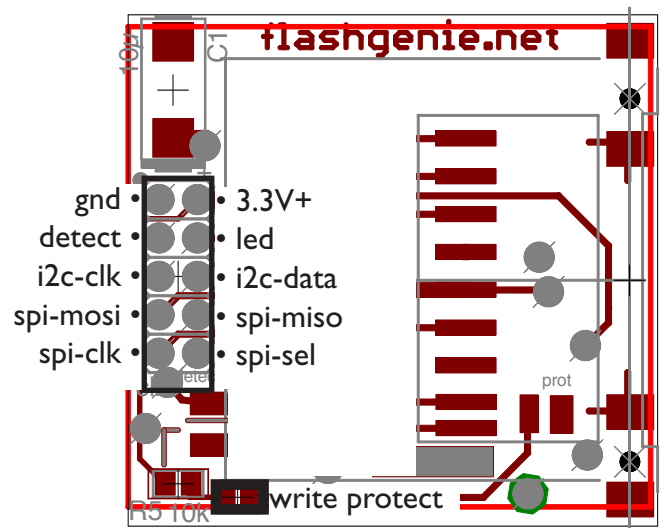
<http://pdfserv.maxim-ic.com/en/ds/DSI340.pdf>

To maintain the clock calendar when powered down, a CR2032 or CR2025 Lithium cell (3 volt) can be installed in the battery holder on the reverse side of the module (with the + side facing away from the pcb, so the stamped code and the + on the battery is visible when inserted).

© Issole Informatique
04170 Thorame-Basse
France

genie@flashgenie.net
www.flashgenie.net

top view



1) Configuring the SPI signal pins

The MMC/SD-Cards expect all signals to be active-low, i.e. at “1” during idle, and “0” when asserted.

The SPI Clock should be set to shift out the data on the **falling edge**, with idle polarity **low**. For Motorola HCS12 or DPS processors, this means the CPOL and CPHA configuration bits in the control registers should both be set to 0.

All data is transmitted MSB first.

The empty character is set to 0xFF, and empty characters should **not** be ignored.

The Slave Select pin should **not** be under SPI control but instead asserted and de-asserted under software control. It should not go on and off with each byte transmitted, but remain continuously asserted (=0) for the entire duration of each access operation.

The shift clock rate can be anything up to 25 MHz. (For MMC cards up to 400 KHz.)

Initialization code produced with Codewarrior for HCS12:

```
SPI0CR1 = 4;          /* Reset the device register */
SPI0SR;             /* Read the status register */
SPI0DR;            /* Read the device register */
/* SPI0BR: ??=0, SPPR2=0, SPPR1=0, SPPR0=0, ??=0, SPR2=0, SPR1=0, SPR0=1 */
SPI0BR = 1;        /* Set the baud rate register */
/* SPI0CR2: ??=0, ??=0, ??=0, MODFEN=0, BIDIROE=0, ??=0, SPISWAI=0, SPC0=0 */
SPI0CR2 = 0;      /* Set control register 2 */
/* SPI0CR1: SPIE=0, SPE=0, SPTIE=0, MSTR=1, CPOL=0, CPHA=0, SSOE=0, LSBFE=0 */
SPI0CR1 = 16;    /* Set control register 1 */
```

A useful method for testing the correct configuration is to output an uppercase “U” in an endless loop and observe the MOSI and CLK pins with an oscilloscope. The “U” outputs alternate 0 and 1 bits. Both signals should be high when the SPI is idle.

Sample code is available on the FlashGenie website at www.flashgenie.net

