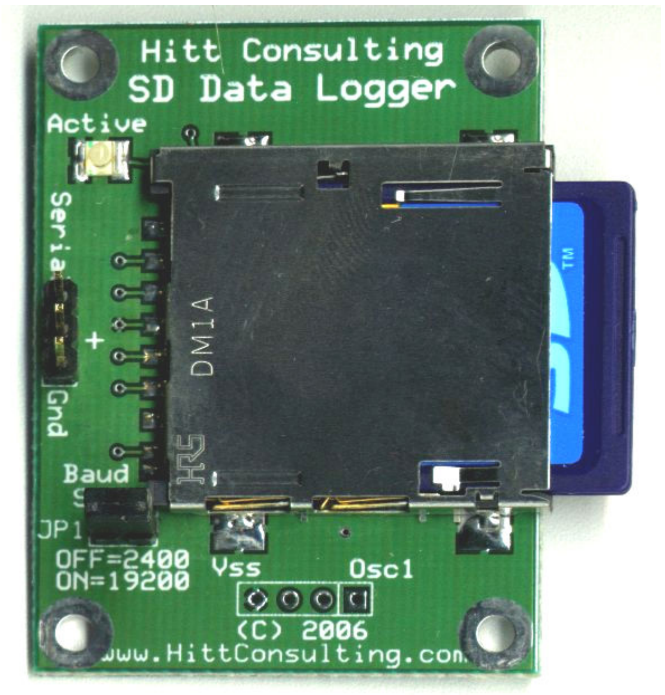


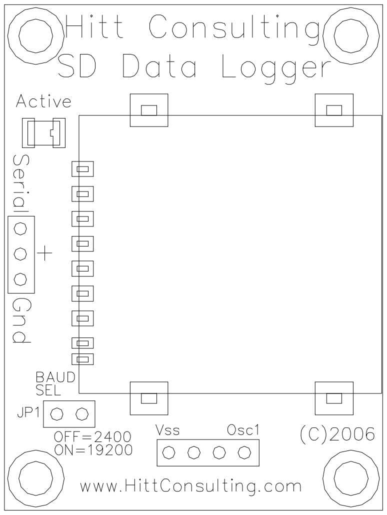
Low Power SD Data Logger

Rev A Manual Rev:B

* Reads & writes FAT16 files (8.3 filenames)
* 19,200 or 2400 baud serial (8N1)
* Store up to 32 megabytes per card
* Programmable data pacing
* Allows up to 16 files per card
* Files are directly readable from PC
* On-board regulator can use 3.6V to 7.5 VDC
* Uses 4mA + card requirements when idle (0.5mA in sleep mode)
* Uses full size SD media
* Simple 3 pin interface (Serial, Power, Ground)

Hitt Consulting
105 S Locust Street
Shiremanstown Pa 17011
info@hittconsulting.com
www.hittconsulting.com

Low Power SD Data Logger



WHAT DOES IT DO ?

Allows creating, reading, and appending data to files stored on SD media cards. The files are stored in FAT16 format and can be directly read and written using a PC with an SD card reader.

Allows strings of variable length to be read and written using a programmable string terminator character.

Programmable pacing time to allow the host to process data "character by character" without missing data sent by the data logger.

Allows read and writing to fast SRAM (16K when using media, 32K when not using media).

SPECS

Power:	3.6 to 7.5 VDC 50mA
Size:	2.0" x 1.5" (not including connectors)
Speed:	19,200 or 2400 Baud
Media:	Uses full size SD memory cards Up to 32MB directly, larger cards work as 32MB.
Files:	16 files per media. 8.3 filenames. Maximum file size is 16MB.
SRAM:	32KB (16K free when using media)

PIN FUNCTIONS

Serial	Serial In/Out connection to the SD data logger. The data logger allows either 19,200 baud or 2400 baud depending on jumper JP1. If the jumper is on, then the baud rate is 19,200. If the jumper is off, then the baud rate is 2400. This pin is used for both sending and receiving data.
+	Supply voltage input to regulator. Vin may be between 3.6 and 7.5 volts. This source must be able to supply 50 milliamps of current.
Gnd	Power supply and serial ground. This MUST also be connected to ground on the device sending the serial data to the module.

SERIAL DATA FORMAT

The serial data format is inverted with eight data bits, no parity and 1 stop bit. Characters are sent using standard ASCII values. Baud rate may be 19,200 or 2400 depending on the setting of jumper JP1 (BAUD SEL).

COMMAND FORMAT

Commands sent to the SD data logger begin with a "bang" or exclamation mark. All of the commands are two letters. For example the command to set the pacing value is "!PV" followed by the value desired. Values are sent as one byte, NOT as ascii. For example to set the pacing value to 100 (10.0 milliseconds) you would use SEROUT SPin, Baud, ["!PV", 100] for the Basic Stamp 2®. DO NOT INCLUDE THE VALUE INSIDE THE QUOTES. For example this is wrong ~~SEROUT SPin, Baud, ["!PV100"]~~

When a value greater than 255 is required, the value is sent as multiple bytes with the least significant byte first.

DATA PACING

The data logger uses programmable pacing when sending data. This is a pause between characters giving the host time to process a character without missing the next character. And without having to ask for each character one at a time.

The pacing value is set in 0.1millisecond increments, and can be from 0 (no delay) to 255 (25.5 millisecond delay).

DATA TYPE (RECORD VS STRING)

The data logger works with two different types of data, records and strings. Understanding the differences is necessary to insure that your information is stored, and can be retrieved, properly from the media card. And that the information can be successfully read by a PC.

Records: A record is a fixed length piece of information. This could be a name, or an ID number, or anything really. The main distinction is that it's length is fixed. If you determine that 20 characters are required for the longest name, then 20 characters are used for ALL names. Usually unused characters are filled with spaces, although this is up to you. Although records can be very wasteful of space, they are useful in that records can be retrieved in any order desired. If we know that each record is 20 characters, and we want to read the 100th record, we can simple use the "goto position" command to get to position 2000 (20 * 100), and read the record.

Often records contain "binary" or byte value data. So instead of storing a number as each character, it's actual value is stored instead. This saves space and is more efficient. In other words a number like 123 would be stored as one byte with a value of one hundred and twenty three, instead of 3 characters "1", "2", "3".

Strings: A string is a variable length piece of information that end with a special "string terminator" character. The data logger will read or write characters until it receives this special terminator character, then is knows this is the end of the string. Since the length of each string is not known, the only way to read a specific string is to start at the beginning and read ALL the strings up to the one you want.

The default string terminator is a linefeed(10). For greatest PC portability, it is recommended to end each string with a CR(13) then a LF(10).

SRAM MEMORY

The on-board SRAM memory can be used to store data that must be saved quickly. For example a rocket data logger might generate data quickly for a short period of time. If this data was written directly to a file the time required to access the file would limit the amount of data that could be collected. By storing the data into SRAM, then after the SRAM is full, reading the data back, and saving it in a file, will allow for much more information to be gathered. User SRAM addresses from 0 to 16383 can be used at any time. Addresses from 16384 to 32767 can be used ONLY if no media commands are used.

ERROR CODES

0 = Successful (No error)
1 = Failed (General media card failure)
2 = Bad or Missing media (Media not properly formatted)
3 = Cannot create file. (16 files max)
4 = File already open. (only 1 file may be opened)
5 = No open file.
6 = Media full.
7 = End of file. (A read operation has reached the EOF)
8 = Invalid Command

MEDIA CARD SETUP

There are two conditions for the media cards used in the SD Data Logger. One is that the file system MUST be FAT16 (just FAT under windows), and that the allocation unit size MUST be 512. For media cards up to and including 32 megabytes this is easily accomplished by formatting the media card in a PC. First open “My Computer” and determine the drive letter of your SD memory card. MAKE SURE YOU KNOW THE LETTER OF THE MEDIA CARD.

Now click “Start->Run” and type in CMD and press enter. A command window will appear (black with white text). Next type in:

```
FORMAT ?: /FS:FAT /A:512
```

Instead of a “?” use the drive letter of your media card.

If you are using a media card larger than 32MB you need to use the value (card size in MB) * 16 instead of 512 after the “/A:” parameter. Then you MUST put the media in the data logger and send the data logger the “!CC” convert card command. This command will force the allocation unit size to the required 512 value. The card will now appear to be 32MB in size. To restore the card to full capacity simply format it again.

ON-BOARD VOLTAGE REGULATOR

There is an on-board voltage regulator. You may connect the "+" pin to a 3.6 to 7.5 volt supply. The module requires about 50 milliamps when actively writing data to the media card.

LOW POWER OPERATION TIPS

Of course keeping the data logger in sleep mode will save power, but by far the most power is used when the device must write to the media card. So this must be minimized. The best way to minimize writing to the media is to store data in the SRAM. Then when the SRAM is full, save the data to the media card all at one time. Just remember that if power is lost, the SRAM contents are lost.

The idle power of SD cards varies greatly. Try different cards to see which ones use the least power.

Use a transistor or FET to turn-off the data logger when not needed. The data logger uses about 0.5mA + media card idle current when in sleep mode. By using a transistor or FET to actually remove power from the data logger can be worthwhile. Just remember that by removing power you will lose the contents of the SRAM.

LIABILITY WARNING

This device should be considered to be experimental. It has NOT gone through extensive testing. As such it could erase or corrupt any and all data on any and all media cards that are used in it. You must assume all responsibility for use of this device. You must agree that Hitt Consulting liability is strictly limited to the purchase price of the module only.

REGULATORY WARNING

This device is NOT FCC approved. It is not in finished product form. It is strictly intended for experimental purposes only. If you wish to use these modules in an actual product (a non-experimental capacity), the modules must first be designed into the product, then the whole product must be approved by the FCC.

HEALTH WARNING

This product contains lead, a chemical known by the state of CA to cause cancer and birth defects and other reproductive harm.

CUSTOMIZATIONS

Customizations to the SD Data Logger module is available from the developer Hitt Consulting. Please contact info@hittconsulting.com.

COMMANDS

CMD	PARAMETERS *RETURN BYTES*	DESCRIPTION
!PV	(1) Pacing value *(1) Pacing value*	Sets the pacing delay between characters sent Returns the pacing value
!LE	(none) *(1) Last error code*	Returns the last error value that occurred
!CC	(none) *(1) Error code*	Convert cards > 32MB Returns error code
!ST	(1) Termination char *(1) Termination char*	Sets the string terminator character Returns string terminator character
!VR	(none) *(2) MinorRev, MajorRev*	Returns the firmware version number. 1st byte is minor rev, 2nd byte is major rev
!SL	(none) *(none)*	Sleep mode (reduced power) Returns nothing
!WU	(none) *(none)*	Wake-up from sleep mode Returns nothing
!WM	(3+) addr_LSB, addr_MSB, Count, Data... *(none)*	Write (Count) bytes to SRAM memory Returns nothing
!RM	(3) addr_LSB, addr_MSB, Count *(Count) Data bytes*	Read from SRAM memory Returns (count) bytes
!OF	(filename) *(1) Error code*	Open file on media card WARNING: Filename MUST be uppercase Returns error code
!CF	(none) *(1) Error code*	Close file on media card Returns error code
!RR	(1) Record size *(Record size+1) Data, Error code*	Read next record data from media card file Returns (Record size) bytes + error code
!RS	(none) *(varies) String, Termination char, Error code*	Read next string data from media card file Returns characters (up to and including terminator) + error code
!AR	(1+) Record size, data... *(1) Error code*	Append record data to media card file Returns error code
!AS	("string" + string terminator) *(1) Error code*	Append string data to media card file Returns error code
!FS	(none) *(5) Filesize, Error code*	Returns file size of open file as 4 bytes + error code
!FP	(none) *(5) Filepos, Error code*	Returns current file position of open file as 4 bytes + error code
!GP	(4) file position (LSB first) *(1) Error code*	Go to given position of open file Returns error code

Basic Stamp ® Example Programs

```
' =========================================================================
'   File...... Demo Text.BS2
'   Purpose... Demo for the SD Data Logger
'   Author.... Terry Hitt
'   E-mail.... terry@hittconsulting.com
'   Started... Nov 22, 2005
'   Updated... May 18, 2006
' =========================================================================
' Stamp to Data Logger Connections
'
'  BS2       Data Logger
'  ---       -----------
'  P15 ----- Serial
'  GND ----- Gnd
'
'{$STAMP BS2}
'{$PBASIC 2.5}

#SELECT $STAMP
  #CASE BS2, BS2E, BS2PE
    T1200       CON     813
    T2400       CON     396
    T9600       CON     84
    T19K2       CON     32
    T38K4       CON     6
  #CASE BS2SX, BS2P
    T1200       CON     2063
    T2400       CON     1021
    T9600       CON     240
    T19K2       CON     110
    T38K4       CON     45
  #CASE BS2PX
    T1200       CON     3313
    T2400       CON     1646
    T9600       CON     396
    T19K2       CON     188
    T38K4       CON     84
#ENDSELECT

Inverted        CON     $4000
Open            CON     $8000

Pacing       CON     30                 ' * 100uSec = 3mSec between characters
Baud         CON     T19K2 + Inverted   ' Use fast serial mode 19.2K baud
ERROR_EOF    CON     7                  ' End of file error code is 7

SPin         PIN 15                     ' Use pin 15 for communication is data
logger

counter      VAR Word                   ' Loop counter
value        VAR Word                   ' Value received from data logger
result       VAR Byte                   ' Error result from data logger
char         VAR Byte                   ' Character from data logger

Start:
  ' Setup Serial Output Pin
  LOW SPin
  PAUSE 1000
  SEROUT SPin, Baud, ["!WU"] ' Wake-up module

  ' Send "Pacing Value" command
  SEROUT SPin, Baud, ["!PV", Pacing]
  SERIN SPin, Baud, 1000, NoResponse, [result]
  DEBUG "Pacing result=", DEC result, CR

  ' We should be the pacing value back as result
  IF result = PACING THEN

    ' Attempt to close file in case it was left open
    SEROUT SPin, Baud, ["!CF"]
    SERIN SPin, Baud, [result]
    DEBUG "CLOSE result=", DEC result, CR

    ' Attempt to open file "DATA.TXT" (Note that filename MUST be CAPS)
    SEROUT SPin, Baud, ["!OF", "DATA.TXT"]
    SERIN SPin, Baud, [result]
    DEBUG DEC result, " OPEN ATTEMPT", CR

    ' Open File result should be zero if file was opened
    IF result = 0 THEN
      ' Send "File Size" command
      SEROUT SPin, Baud, ["!FS"]
      ' Receive file size values from data logger (4 bytes)
      SERIN SPin, Baud, [counter.LOWBYTE, counter.HIGHBYTE, value.LOWBYTE,
value.HIGHBYTE, result]

      ' Show current file size in debug window
      DEBUG DEC result, " FILESIZE ATTEMPT", CR
      DEBUG "File Size = ", DEC counter, ":", DEC value, CR
      PAUSE 2000                       ' Wait 2 seconds

      ' Prepare to append 100 text strings to the file
      FOR counter = 1 TO 100
        DEBUG "Appending ", DEC counter, "  "
        ' Send "Append String" command (default string terminator is LF)
        SEROUT SPin, Baud, ["!AS", "Value=", DEC counter, CR, LF]
        SERIN SPin, Baud, [result]     ' Get append result
        DEBUG DEC result, CR           ' Show append result
      NEXT

      ' Get new file size
      SEROUT SPin, Baud, ["!FS"]
      SERIN SPin, Baud, [counter.LOWBYTE, counter.HIGHBYTE, value.LOWBYTE,
value.HIGHBYTE, result]
      DEBUG DEC result, " FILESIZE ATTEMPT", CR
      DEBUG "File Size = ", DEC counter, ":", DEC value, CR
      PAUSE 2000

      ' Send "Goto Position" command, to start of file
      SEROUT SPin, Baud, ["!GP",0,0,0,0]
      DEBUG "Waiting for GP result..."
      SERIN SPin, Baud, [result]
      DEBUG DEC result, CR
      IF result = 0 THEN
        counter = 0
        DO
          counter = counter + 1
          DEBUG "Read ", DEC counter, "  "
          ' Send "Read String" command
          SEROUT SPin, Baud, ["!RS"]
          DO
            SERIN SPin, Baud, [char]
            IF char = LF THEN EXIT
            DEBUG char
          LOOP
          ' Get result
          SERIN SPin, Baud, [result]
          DEBUG " Result = ", DEC result, CR
        LOOP UNTIL result = ERROR_EOF  ' Repeat until we get an END-OF-FILE
error code

        ' Get new "File Size"
        SEROUT SPin, Baud, ["!FS"]
        SERIN SPin, Baud, [counter.LOWBYTE, counter.HIGHBYTE, value.LOWBYTE,
value.HIGHBYTE, result]
        DEBUG DEC result, " FILESIZE ATTEMPT", CR
        DEBUG "File Size = ", DEC counter, ":", DEC value, CR
        PAUSE 2000

        ' Close File
        SEROUT SPin, Baud, ["!CF"]
        SERIN SPin, Baud, [result]
        DEBUG "CLOSE result=", DEC result, CR
      ENDIF
    ELSE
      DEBUG DEC result, " COULD NOT OPEN FILE.", CR
    ENDIF
  ELSE
    DEBUG DEC result, "  DID NOT RECEIVE PACING VALUE FROM DATA LOGGER.", CR
  ENDIF

Done:
  DEBUG "FINISHED...", CR
  GOTO Finished

NoResponse:
  DEBUG "NO RESPONSE FROM DATA LOGGER.", CR

Finished:
  GOTO Finished
```