The Sentinel: A users perspective©

Revision2 (March 2005)
By: M. A. Rehman & P. Bernardson **ProTek Analog**

1.0 Introduction:

This article is written from the viewpoint of a user, trying to grapple with the intricacies and functions of the Sentinel device. It elaborates not only on the currently available documentation of the device, but also includes the nitty - gritty of the operation and uses, not included in data sheets. The attempt is to make a new user comfortable with the device. Additional help is available in the form of demo boards and technical support from the applications engineers at PROTEK ANALOG and field applications engineers. For additional support please contact the nearest PROTEK ANALOG Sales office. Finally, this paper should be read in conjunction with the datasheet.

2.0 Basic description of the Sentinel device:

The Sentinel is a *low power, CMOS mixed signal monolithic device* which acts as an analog front end for a number of applications. Examples are, sensor interfaces, 3 – D computer peripherals, Instrumentation interface, Data acquisition and capture, data communications, electronic control and command.

The Sentinel requires a digital controller to operate, and may or may not, need additional memory depending on the application. The digital controller may be a microprocessor, microcontroller, a CPLD, a FPGA, discrete logic, a desktop PC, a laptop PC or any other kind of digital controller which may be easy to implement for a particular application.

In this sense the Sentinel is different from competing devices, in that it does not need to be tied in to a particular controller.

The Sentinel is packaged in a 16 pin package (SOIC, PDIP etc). It is also sold as a die. In addition the Sentinel and its complete system (digital controller, memory) can be delivered as a "Multichip System in a package"
(MSP™) if required.

The database of the Sentinel may be integrated (using CMOS) with additional features and functions to produce an Application Specific Standard product (ASSP).

The Sentinel: A users perspective©

Revision2 (March 2005)
By: M. A. Rehman & P. Bernardson **ProTek Analog**

The pre-designed cells (functional blocks) of the Sentinel are also available in the Chandler Chip™ configuration to implement systems on chip using the Sentinel and other peripherals as a custom design. Please call your nearest PROTEK ANALOG Sales office for more information.

The conceptual block diagram of the Sentinel is shown below.

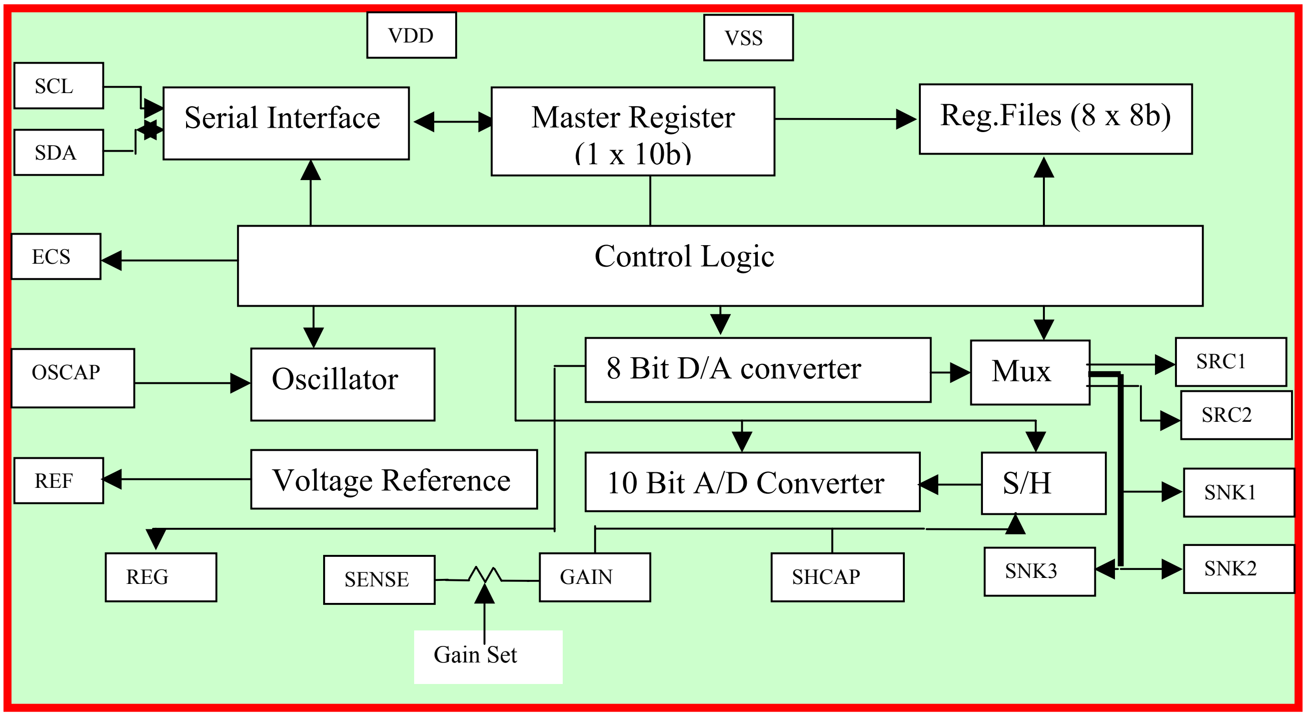


FIGURE 1. Conceptual Block Diagram of the SENTINEL

The Pin functions and descriptions are tabulated in TABLE 1.0 below.

TABLE 1. Pin descriptions

Pin no	Name	I/O	Description
1	SDA	Bi-dir	Serial Data
2	SCL	Input	Serial Clock
3	ECS	Output	EEPROM Select/Chip Select/Memory select
4	SRC1	Output	Source current from the DAC channel 1.
5	SRC0	Output	Source current from the DAC channel 2
6	REG	Output	Resistor from here to ground establishes the current drive from the DAC
7	SNK2	Output	Selectable current sink for the DAC source currents channel 1 (MUX operation)
8	VSS	Power	Ground
9	SNK1	Output	Selectable current sink for the DAC source currents channel 2 (MUX operation)
10	SNK0	Output	Selectable current sink for the DAC source currents channel 3 (MUX operation)
11	GAIN	Output	Gain setting pin and output for the internal amplifier used in the analog input channel.
12	SHCAP	Input	External capacitor for the sample and hold operation
13	SENSE	Input	Analog input pin
14	REF	Output	Reference voltage for data conversion
15	OSCAP	Input	External capacitor for the oscillator
16	VDD	Power	5.0V power

Registers.

There are eight (real) File Registers (8b each) and one Master Register (10b). The File Registers are used to store data specifying output current drivers, oscillator's frequency and a time the DAC is on (DACON). The Master Register (MR) is used to store the output of the ADC in order to transmit it to the Controller.
The communication protocol between SENTINEL and the microcontroller uses 'frames', illustrated in paragraph 3.0., starting on page 6.

By: M. A. Rehman & P. Bernardson **ProTek Analog**

The 3 Frame bits: B4=A2(msb) B5=A1 and B6=A0(lsb) have a dual function. They either specify the File Registers' addresses – when bit B3=M(=Mode) is logical '0', or they define a Command (=Instruction) – when the bit 'M' is a logical '1'. See TABLE2.

TABLE 2. Registers' Functions

Note2.1: A2, D7 = msb; A0, D0 = lsb

Note2.2: Only one of the SNKs & only one of SRCs switches are 'ON' at the same time

Note2.3: MR = Master Register (10 b); ADC=Analog to Digital Converter (10b); DAC=Digital toAnalog Converter (8b)

Note2.4: The SENTINEL's $IDAC_{OUT}$ current is used to drive sensor, etc.

B3 = M	B4 B5 B6 = A2 A1 A0	**Function**	Data Bits used
0	0 0 0	B7(=D7)→B14(=D0) specify the current out of SENTINEL = $IDAC_{OUT}$	8
0	0 0 1	”	8
0	0 1 0	”	8
0	0 1 1	”	8
0	1 0 0	”	8
0	1 0 1	”	8
0	1 1 0	data specify the Time the DAC stays on (= ”DACON”)	8
0	1 1 1	data specify the frequency of the oscillator	5 msb
Commands code		**Action**	
1	0 0 0	data in Reg. 0 set the $IDAC_{OUT}$. SRC0 & SINK0 are ON & ADC Writes to MR	
1	0 0 1	data in Reg. 1 set the $IDAC_{OUT}$. SRC1 & SINK0 are ON & ADC Writes to MR	
1	0 1 0	data in Reg. 2 set the $IDAC_{OUT}$. SRC0 & SINK1 are ON & ADC Writes to MR	
1	0 1 1	data in Reg. 3 set the $IDAC_{OUT}$. SRC1 & SINK1 are ON & ADC Writes to MR	
1	1 0 0	data in Reg. 4 set the $IDAC_{OUT}$. SRC0 & SINK2 are ON & ADC Writes to MR	
1	1 0 1	data in Reg. 5 set the $IDAC_{OUT}$. SRC1 & SINK2 are ON & ADC Writes to MR	
1	1 1 0	Not used	
1	1 1 1	Not used	

The Sentinel: A users perspective©

Revision2 (March 2005)

By: M. A. Rehman & P. Bernardson **ProTek Analog**

The following sections describe the pin functions and associated operations for the SENTINEL and should be used as a guide when using the SENTINEL. The SENTINEL is a robust device and if the communications with the device are established as per the SENTINEL protocols it will be a joy to use!!

3.0 **Functionality and Operation:**

Communication with the SENTINEL is done via 3 bit bidirectional parallel pins: Serial_Data_Addresses (SDA) – Pin1, Serial_Clock (SCL) – Pin2 and External_Chip_Select (ECS) – Pin3.

SDA carries serial data addresses and commands. Data is writen into the SENTINEL on **rising edge of SCL**. (Therefore, data cannot change just prior to the SCL positive edge.)

SCL is the clock that shifts the data and also works with SDA to provide the command signaling for the communications as described below.

External_Chip_Select (ECS) is the 'memory select/chip select' pin which enables users who want to use external selectable chips (memory or other device) to enable these when required while inhibiting the rest of the SENTINEL operations.

The communication format with the SENTINEL uses **'Frames'**. They consists of

(i) : START COMMAND
(ii) : STOP COMMAND
(iii): DATA/ADDRESS&INSTRUCTIONS

There are three *modes* of communications that use the SDA and the SCL pins. These are:

M1: SENTINEL/External memory Chip Select (ECS) MODE
M2: WRITE MODE
M3: READ MODE

By: M. A. Rehman & P. Bernardson **ProTek Analog**

M1: Select SENTINEL / ECS:

The following is the frame format for the SELECT mode. (For timing see Fig. 5.)

← *Signal flow from Controller to SENTINEL*:

B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15
START	S	W /R	M	A2 msb	A1	A0 lsb	D7 msb	D6	D5	D4	D3	D2	D1	D0 lsb	STOP
START	0	X	X	X	X	X	X	X	X	X	X	X	X	X	STOP

Where:

B0 = START signal (Fig. 2)

B1 = Select SENTINEL ('1') or External Chip Select ECS ('0')

B2 = Write (W='1') or Read (W='0') Register of Address A2 A1 A0

B3 = Mode (M):

If M=**0** => A2 A1 A0 is a Register **Address**

If M=**1** => A2 A1 A0 is a **Command (Instruction)** specified in Table2

B4 = A2 = the msb of the Register Address

B5 = A1 = Register Address bit

B6 = A0 = the lsb of the Register Address

B7 –> B14 = Register's data D7 -> D0

B15 = Stop signal (Figures 3 & 4)

FIGURE: Select ECS.

M2: Write 8b Data to SENTINEL.

(For timing see Fig. 6.)

← *Signal flow from Controller to SENTINEL*:

B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15
START	S	W /R	M	A2 msb	A1	A0 lsb	D7 msb	D6	D5	D4	D3	D2	D1	D0 lsb	STOP
START	1	1	0	A2	A1	A0	D7	D6	D5	D4	D3	D2	D1	D0	STOP

Where:

B0 = START signal (Fig. 2)

B1 = Select SENTINEL ('1')

B2 = Write (W='1') to Register of Address A2 A1 A0

B3 = Mode M=**0** => A2 A1 A0 is a File Register **Address**

B4 = A2 = the msb of the Register Address
B5 = A1 = Register Address bit
B6 = A0 = the lsb of the Register Address
B7 –> B14 = Register's data D7 -> D0
B15 = Stop signal (Figures 3 & 4)

FIGURE: Write to File Register.

M3: Reading data from SENTINEL. (Timing is on Fig.7.)

1) Reading File registers.

← *Signal flow from Controller to SENTINEL*:

B0	B1	B2	B3	B4	B5	B6	
START	S	W /R	M	A2 msb	A1	A0 lsb	Float SDA to '1' & drive SCL to '0'. Wait for response from SENTINEL (viz.: Pulling SDA to 0)
START	1	0	0	A2	A1	A0	

When ready, Sentinel responds by pulling SDA to 0 and sending (to the Controller) 8 bit Data from the Register Address A2 A1 A0 and (2 lsb) extra bits set to '1':

← *Signal flow from SENTINEL to a controller:*

		B7	B8	B9	B10	B11	B12	B13	B14	B15	B16	B17
Pull SDA to 0 & Start clocking data.	Load MR	D7	D6	D5	D4	D3	D2	D1	D0	1	1	STOP

Where:
B0 = START signal (Fig. 2)
B1= Select SENTINEL ('1')
B2 = Read (W='0') Register of Address A2 A1 A0
B3 = M=**0** => A2 A1 A0 is a Register **Address**
B4 = A2 = the msb of the Register Address
B5 = A1 = Register Address bit
B6 = A0 = the lsb of the Register Address
B7 –> B14 = Register's data D7 –> D0
B15 –> B16 = Forced '1's
B17 = Stop signal (Figures 3 & 4

FIGURE: Read File Register.

2) Reading ADC. (Timing is on Fig.7.)

When it is required to read the result of the ADC from SENTINEL, (and also to drive a current IDAC via the appropriate SRC and SINK current sources – see Table2), the procedure shall be as follows:

← *Signal flow from Controller to SENTINEL*:

B0	B1	B2	B3	B4	B5	B6	
START	S	W /R	M	A2 msb	A1	A0 lsb	Float SDA to '1' & drive SCL to '0'. Wait for response from SENTINEL (viz.: Pulling SDA to 0)
START	1	0	1	A2	A1	A0	

When ready, Sentinel responds by pulling SDA to 0 and sending (to the Controller) 10 bit Data from the Master Register:

← *Signal flow from SENTINEL to a controller:*

		B7	B8	B9	B10	B11	B12	B13	B14	B15	B16	B17
Pull SDA to 0 & Start clocking data.	Load MR	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	STOP

Where:

B0 = START signal (Fig. 2)

B1= Select SENTINEL ('1')

B2 = Read (W='0') Register of Address A2 A1 A0

B3 = M=**1** => A2 A1 A0 is a **Command (Instruction)** specified in Table2

B4 = A2 = the msb of the Register Address

B5 = A1 = Register Address bit

B6 = A0 = the lsb of the Register Address

B7 –> B16 = ADC data D9 –> D0

B17 = Stop signal (Figures 3 & 4)

FIGURE: Read ADC.

Timing.

The timing of the communications interface and operations are described below.

To begin communications the first signal that must be applied to the serial input port is the START command. The start command is defined by:

SDA goes high during the time SCL is at a high level as shown below:

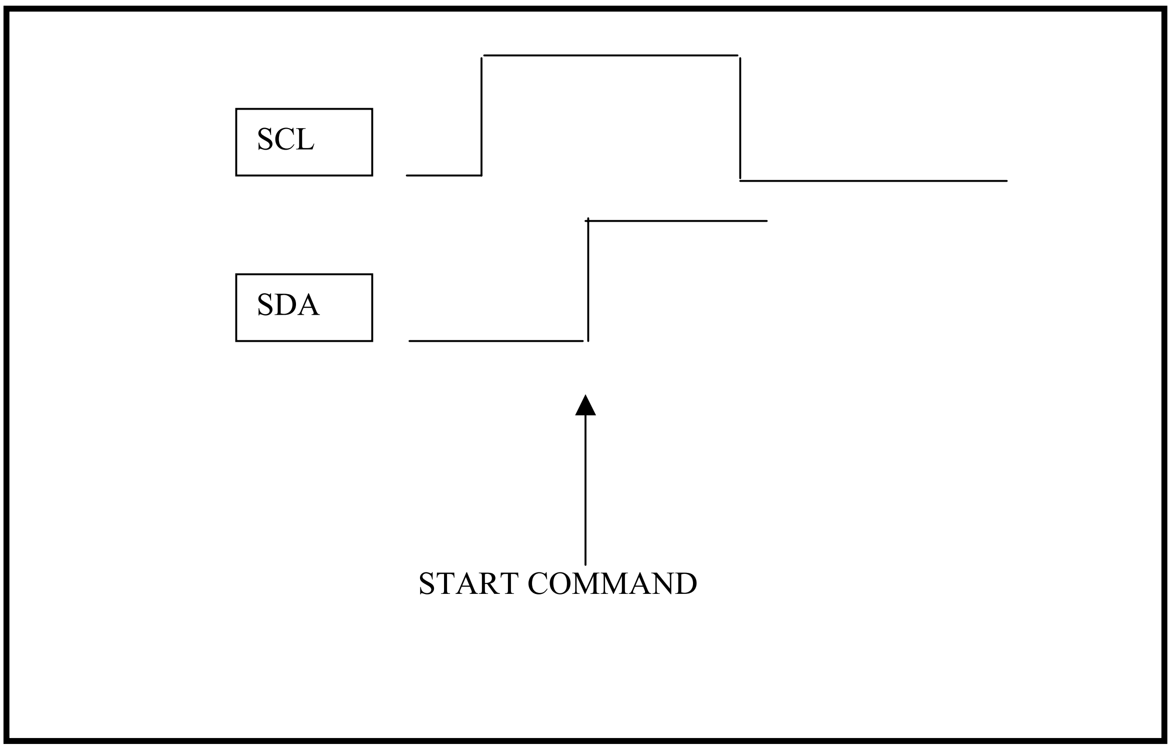


FIGURE 2: The START COMMAND

The START command can be part of continuous cycle of events or isolated. This command prepares the SENTINEL to start receiving data (if data is to be written to it) or to prepare it to send data out (when reading data from the SENTINEL).

All communications with SENTINEL start with this command.

To end the communications frame (commands, addresses and data) a STOP command is used. The STOP command resets the communications interface and some registers within the SENTINEL to prepare for the next communications frame.

The STOP command is generated when the SDA line goes from high to low during the high time of SCL as shown below.

Note that the STOP command can sometimes be generated by accident. i.e take care not to drive SDA low when SCL is high unless you actually want to stop communications.
The STOP command is shown below in FIGURE 3.0.

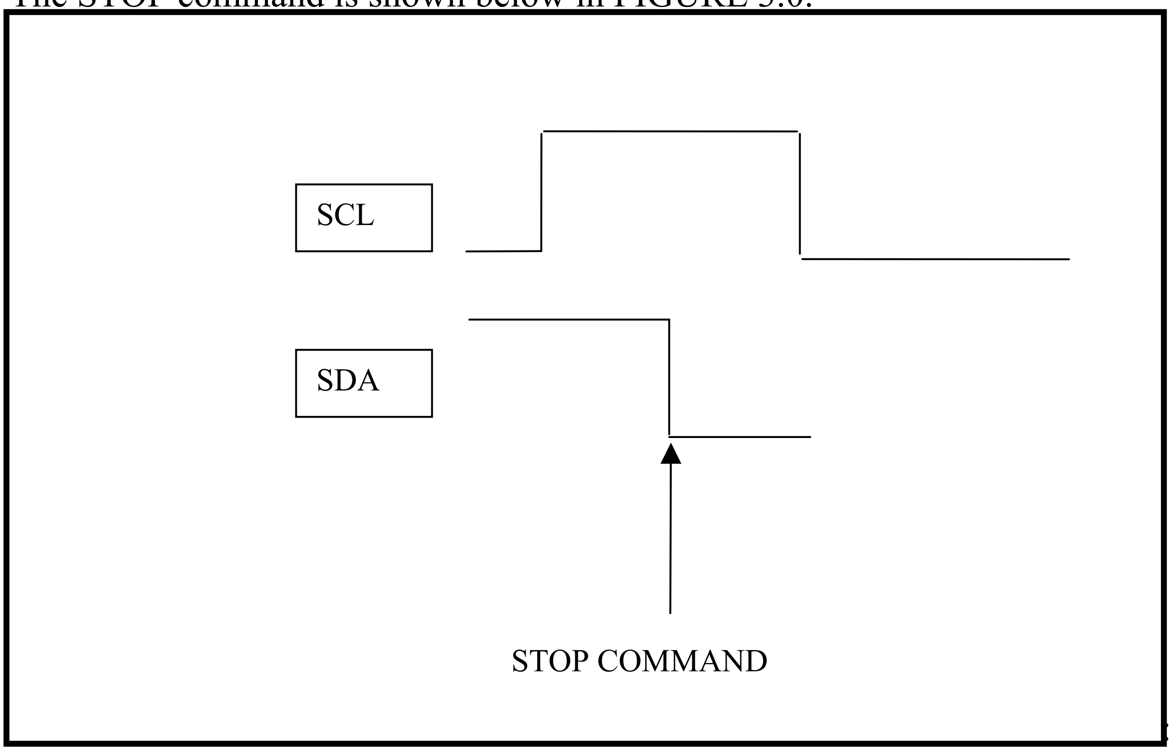


FIGURE 3: THE STOP COMMAND

Sometimes it may be necessary to first drive SDA high when SCL is high and then drive it low to generate a STOP command. This is usually the case when the previous data bit was a logic ‘0’. If this is the case then the signals are as shown below in FIGURE 4.

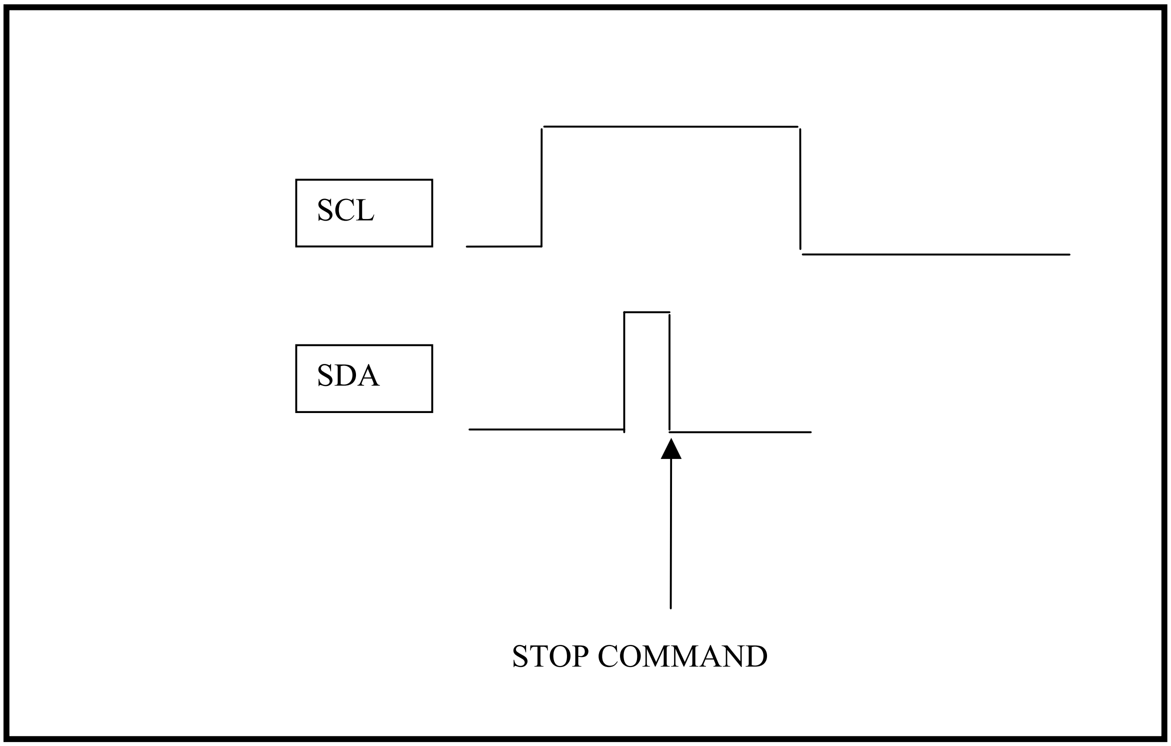


FIGURE 4: STOP Command when previous bit was a '0'.

All addresses and data are framed within the START and STOP commands. Addresses are for the registers within the register files and data is for loading and reading from the registers. The START command, the STOP command, addresses and data can be sent one at a time or in a burst.
Data is **writen into the SENTINEL** on **rising edge of SCL**.

When **reading** from the SENTINEL the address is sent after the START and select inputs. The SDA line is then allowed to float high and SCL is driven low. When the SENTINEL is ready, it pulls the SDA line low. 10 clocks are used to clock out 10 bits of A/D converted data on the negative edge of SCL.
When register data is clocked out only 8 bits are output. The last two bits are forced to '1'.

The current will be used to make a measurement, drive a sensor, etc. The response of the sensor will be converted to a digital word using the A/D and the S/H.

The modes of communication are explained and described in more detail below.

M1: ECS. Memory select/chip select mode.

- Send the START command to the SENTINEL.
- Make the first data bit after the START command = logic ‘0’.
- Rest of the bits are don’t care bits.
- Send the STOP command.

When this frame is sent to the SENTINEL it simply drives the ECS pin (Pin 3) high on the first SCL after the START command. ECS is kept high until the STOP command when it is driven low again. This is shown below. ECS is used to enable an external chip, memory etc so that it can be used.

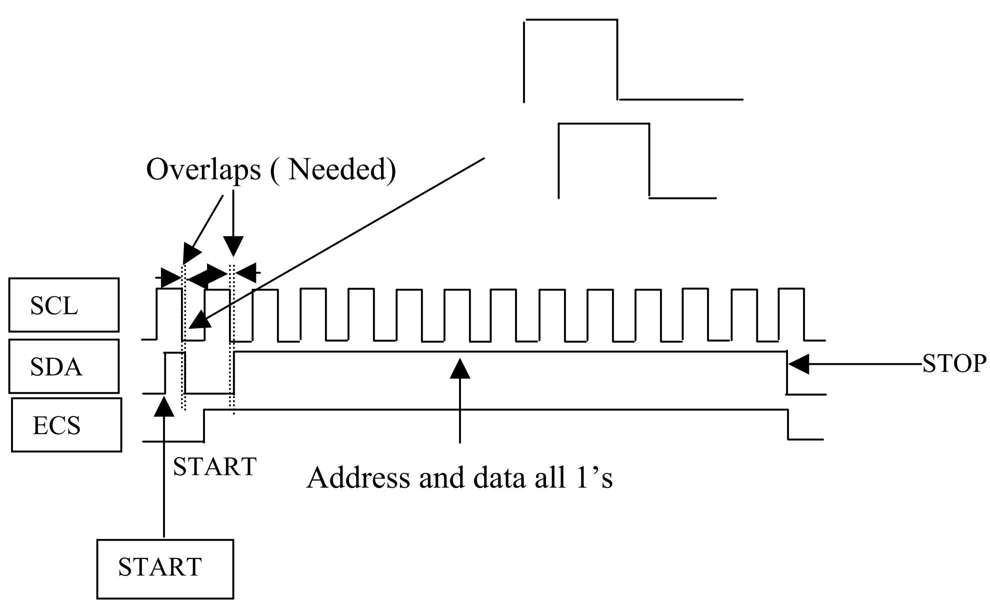


FIGURE 5: ECS turned ON/OFF

Note the overlaps as indicated on the diagram. It is **very important** that the SDA signal not go low when SCL is high unless it is a STOP. **This is a STOP condition.** Data can **only change** when SCL is **low.** Therefore the overlaps must be there. *Make* *sure that SCL has gone low before changing the SDA line from high to low or vice versa, except when sending a STOP condition.*

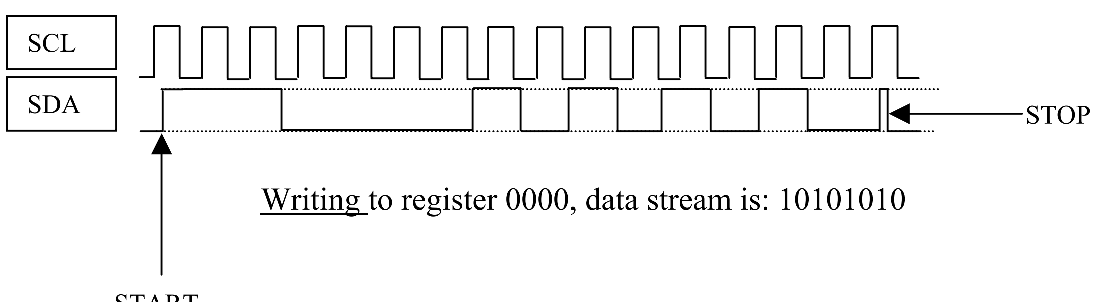
M2: WRITE Mode

During the WRITE Mode, data is written to the internal real registers of the SENTINEL. There are 6 real registers each having 8bits.

Please note that **when writing** to the real registers, the **bit M is always '0'.** (See Table2.)
Data can only change when SCL is '0'.
16 SCL cycles are necessary to write the real registers including START and STOP commands.

Note: It is important to write **at least 8 bits of data.** Internal logic loads data into the register files when the 8th bit is read in. Also SCL must make a complete cycle for the 8th bit. i.e it must read data on the rising edge and store on the falling edge.

The graphical representation of the WRITE operation is shown in FIGURE 6.



Note the overlaps and data change during low period of SCL.

FIGURE 6. WRITE Frame

All the WRITE frames have this format and sequence.

M3: READ MODE:

The READ mode sequence is slightly different. In this case a command is sent to the SENTINEL via the serial interface to read a register or, take a reading of the results of A/D conversion or, to generate DAC currents. The SENTINEL then does the task requested and responds to the host (digital controller) when it is ready. When the host receives the "data ready to read" signal it clocks out the data to be read using SCL. This operation is explained more fully below.

The READ sequence of signals is as follows. (See also FIGURE: Read FileRegister and FIGURE: Read ADC; on pages 7 & 8.)

- START command
- Bit 1 = '1'(Select SENTINEL)
- Bit 2 = '0' (Not WRITE=Read)
- B3 = Mode (M):
  If M=**0** => A2 A1 A0 is a Register **Address**
  If M=**1** => A2 A1 A0 is a **Command (Instruction)** specified in Table2
- Bit 4 = A2 (msb)
- Bit 5 = A1
- Bit 6 = A0 (lsb)
- Float SDA to '1', drive SCL to '0'
- Wait for response from the SENTINEL
- SENTINEL pulls the SDA line to '0' (SENTINEL response/acknowledge)
- Clock SCL once to load data into the Master register for o/p.(First bit emerges on the negative edge of the first clock)
- Clock SCL 10 times and obtain data from the SENTINEL on the negative edge SCL. If reading the file register (i.e.; M=0), the first 8 bits are the relevant data, next two bits are constant '1's. If taking an ADC reading (i.e. M=1), the first 10 bits are the relevant results of the conversion. (MSB first).
- STOP Command.

This sequence is shown graphically below.

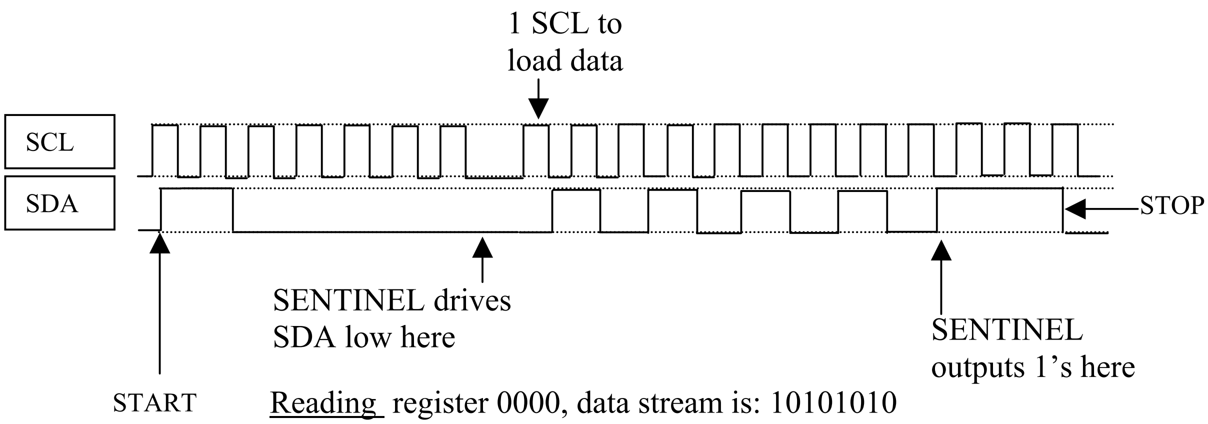


FIGURE 7. Read file register 000.

Other operations.
In the foregoing the focus was on WRITING and READING operations using the SENTINEL serial interface. In what follows, more functional modes and functional blocks of the SENTINEL are explored.

The built-in oscillator:
The SENTINEL contains a built – in RC oscillator which needs a capacitor at Pin 15 (to ground) to work.. This external capacitor and the internal circuitry of the SENTINEL set the frequency of oscillation. The oscillator is also trim-able using the serial interface and codes sent to the SENTINEL via SDA and SCL. A maximum frequency of 1.0 Mhz has been used successfully.

The address assigned to the oscillator is 111. (See Table2.) The oscillator control sequence is written to the oscillator control register in the same way as the data written to a real register. The first 5 bits of data following the oscillator address changes the frequency of the oscillator by switching resistors in and out of the oscillator feedback circuit thereby changing the frequency. The last three bits are ignored when programming the oscillator. The internal resistors which set the frequency are 1k, 2k, 4k, 8k, 16k, 32k.. The tolerance on these resistors is ±20% so the time constant can change by at least that much from sample to sample of the device, if the oscillator capacitance is kept constant.

The oscillator frequency change characteristic is hyperbolic in shape with rapid changes for the first few codes and slow for the higher codes. Once the oscillator is programmed with a code, that code remains resident until power down. After this it has to be re-programmed.

It is possible to apply an external oscillator signal to the OSCAP pin (Pin 15) if the internal oscillator is not accurate enough, thereby forcing the frequency.

The internal oscillator is used as a timer for DAC operations, A/D operations, MUX operations and DACON operations (explained below).

The internal oscillator signal appears at the OSCAP pin as a sawtooth with a DC bias as shown below. The actual frequency inside the SENTINEL is a divide – by – two of this frequency so that a 50% duty cycle waveform can be used internally.

If the internal oscillator needs to be used for external timing operations the OSCAP signal should be buffered and sliced to generate a rail to rail signal.

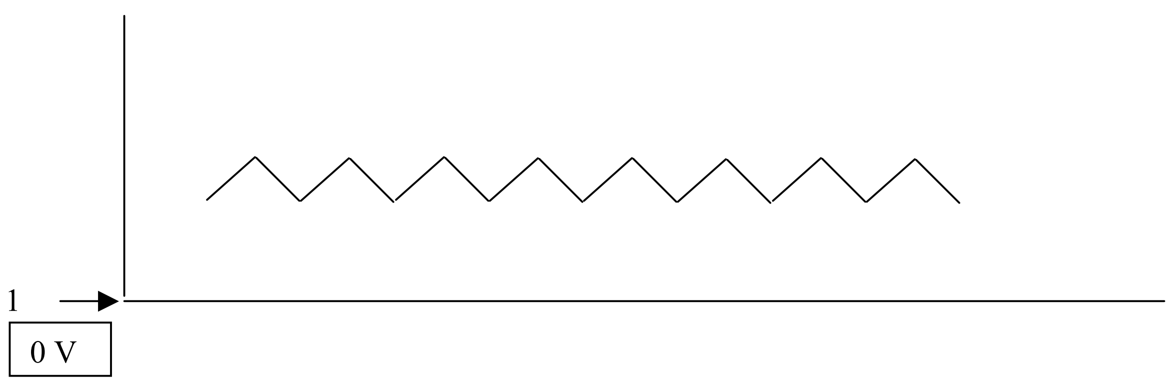


FIGURE 8: Signal at OSCAP pin. Frequency = 2X internal frequency

Oscillator programming sequence:

- Bit 0 = START command
- Bit 1 = 1 (Select SENTINEL)
- Bit 2 = 1 (Write)
- Bit 3 = 0
- Bit 4 = 1
- Bit 5 = 1
- Bit 6 = 1
- Bit 7 = D4
- Bit 8 = D3
- Bit 9 = D2
- Bit 10 = D1
- Bit 11 = D0
- Bit 12 = X
- Bit 13 = X
- Bit 14 = X
- Bit 15 = STOP Command

Graphically the waveforms are the same as shown for writing the real registers.

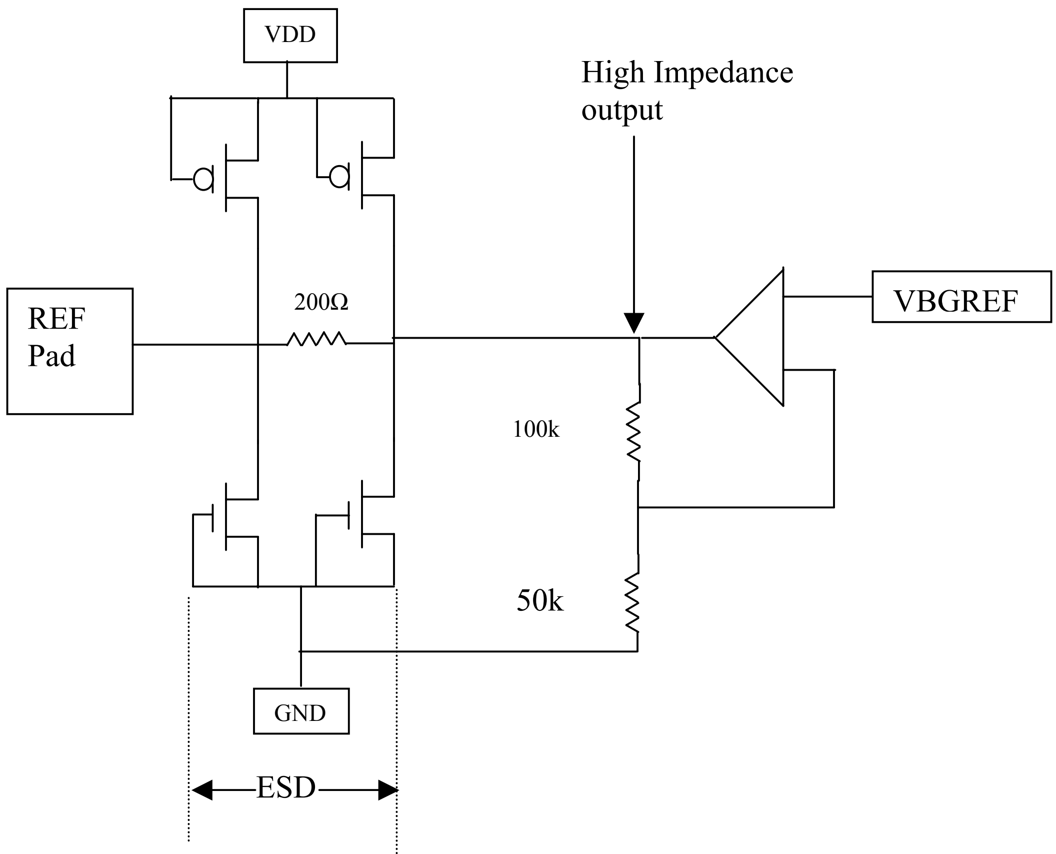
The voltage reference:

There is an internal bandgap reference voltage reference available. The output of the bandgap reference is further regulated by a simple series voltage regulator and connected to the REF pin, (Pin 14). A capacitor at the REF pin stabilizes and cleans up the reference. The voltage reference is used in the DAC and the A/D. If necessary an external voltage reference may be used at the REF pin. The load that this voltage will see is as shown below. The bandgap is not trimmed and can vary significantly. VBGREF is a 1.2V (nominal). This leads to a 3.6V (nominal) reference.

The Sentinel: A users perspective©

Revision2 (March 2005)

By: M. A. Rehman & P. Bernardson **ProTek Analog**



Analog input circuits:

The SENSE pin is the inverting input of an opamp. The non-inverting input is connected to ground. The output of the opamp is the GAIN pin. The input signal is typically a current. This current is converted to a voltage using a resistor connected from GAIN to SENSE. The output of the opamp is also connected to the S/H circuit. The S/H uses an external capacitor as its hold capacitor for simplicity and to avoid clock feedthrough and allow the user flexibility in usage.

The analog input is the SENSE pin. However, the S/H input pin may also be used as the analog input if the input can be held long enough.

The D/A converter, DACON and Current Sink operation:

The DAC is 8 Bit. The output of the DAC drives two current sources which are proportional to the 8 bit data stored in the 6 File Registers when generating the currents. Therefore 256 levels of current can be generated. The value of the current is dependent on the resistor connected from the REG pin (Pin 6) to ground. This current is multiplied by a 1000 to generate the output current. The voltage reference is used to generate the DAC levels.

An important point to note is that the DAC does not stay on continuously. *In order to provide low power operation the DAC actually stays on for an amount of time depending on the value stored in the DACON register.* The maximum time that the DAC can stay on is 256 internal clock cycles and the minimum one is 12 cycles. The DACON control register address is 110. Only the 5 msb bits of the data word are used. The rest of the 8 bit data word is ignored.

Please also note that every time the DAC is used, the DACON register must be programmed for how long it is going to be on.

The DACON also programs how long the current sinks will stay on.

The current sinks are used with the DAC to energize and measure at least three different channels. Please refer to the FIGURE 8 below.

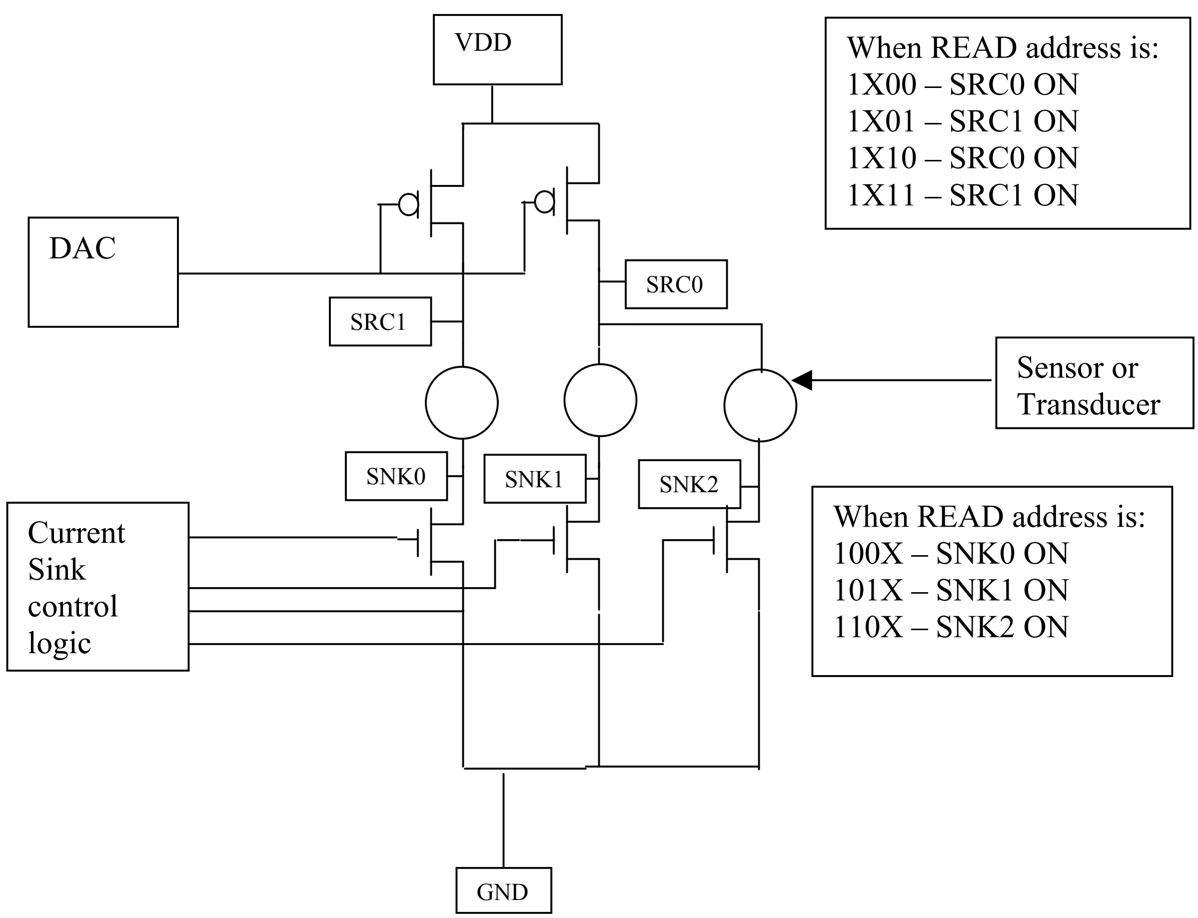


FIGURE 8: DAC source and sink circuit

When the DAC and the sinks are used in this way, the time the DAC is on is first set by writing to the DACON register. This time depends on the device or the sensor that is to be energized. Then the INSTRUCTION command (M=1) is issued. (See Table 2 and a FIGURE: Read ADC on page 8.)

For example, the Controller sends the following Frame to SENTINEL:

START,1,0,1, 0, 0, 0   =
START,S,W,M,A2,A1,A0

The W=0 (=>Read), M=1 and the address field A2 A1 A0 = 000. According to Table 2, the value in register 000 will be used to generate the DAC current I_{DAC}, which will be sourced by SRC0 and sunked by current sink SNK0. If the sensor output needs to be measured, then it will be connected to the SENSE pin. The sensor output will be captured by the S/H during the DACON time and converted to a digital value by the 10 Bit A/D converter and read out using the 'Read_ADC' command as described above.

It is not necessary to use a sensor at all. Only the DAC may be used. However, in all cases that the DAC is used, the DACON value should be set *before it is used* and then a 'Read_ADC' command issued.
The DAC current output transistors need some overhead voltage to operate correctly. Typically they will operate up to 3.5V for maximum output currents. (50 mA each) Please see FIGURE 9 for a graphical representation of the DAC, SRCi and SNKi timing.

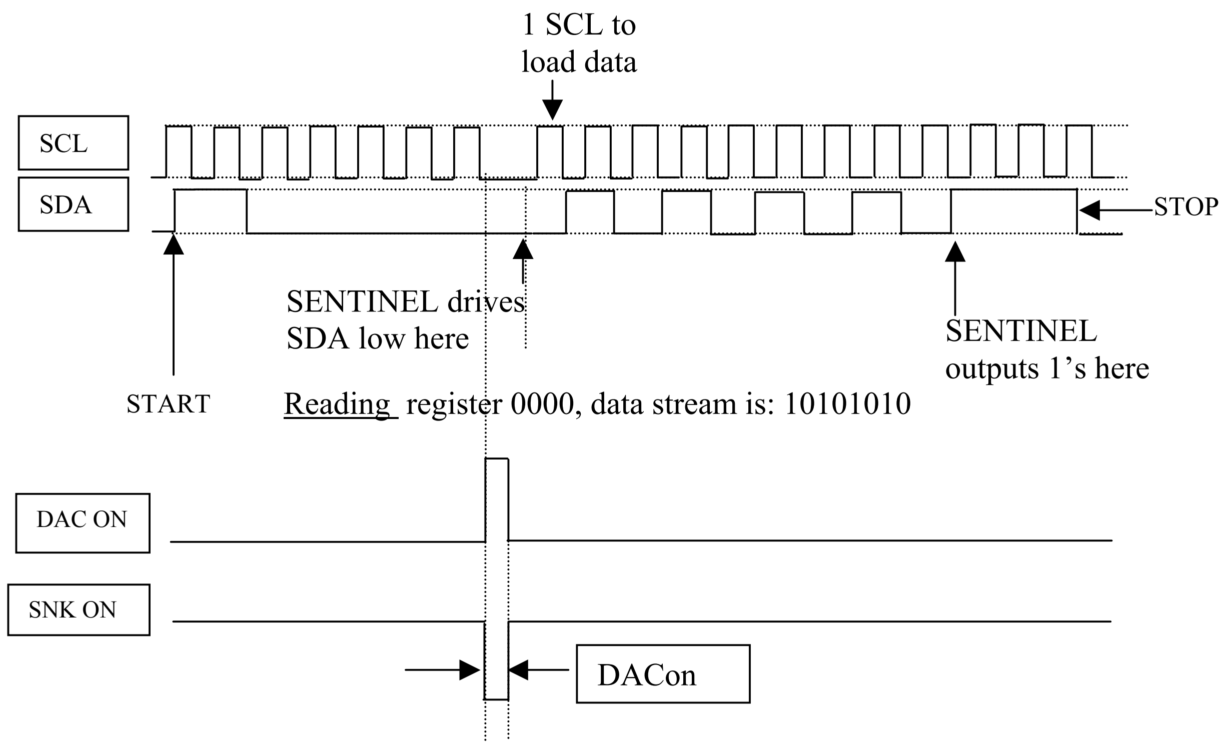


FIGURE 9: DAC , SRC and SNK during TAKE READING Operation

1 Typical configuration for using the SENTINEL.

Resistors:

RREG:	Pin 6 to Ground	100k, 1/4W, 1%
RRFB	SENSE to GAIN	100k, 1/4W, 1%
RSDA	SDA to VDD	10k, 1/4W, 5%

Capacitors:

CREF	REF to Ground	10nF, 100V, 10%
CSHCAP	SHCAP to Ground	10nF, 100V, 10%
COSCAP	OSCAP to Ground	1nF, 100V, 1%

2 Conclusions and discussions:

The SENTINEL is a compact low power, medium speed analog front end designed for multiple applications in the sensor, instrumentation, computer peripheral, control, data acquisition etc applications. It is very robust in its operation and very flexible. It can be used with a variety of digital controllers.

Its applications include operations such as simple low power A/D conversion (10 bit), complete closed loop systems such as a DAC, sensor, A/D, control systems.

It is very simple to use once the operation is understood which is the objective of this paper. The entire communications interface consists of two wires! Since these are serial signals it can be used remotely (with either a wireline or a wireless interface).

In the ultimate analysis the applications of the device and its usage is limited only by the imagination of the user. Once its various operating modes are understood it will be found to be a very useful little device indeed.

The Sentinel: A users perspective©

Revision2 (March 2005)

By: M. A. Rehman & P. Bernardson **ProTek Analog**

Notes: