

How to Use DSO150 Library

Applicable library version: 15003-052 or newer

1. DSO150 Capture Engine Model

Fig. 1 shows the structure of the DSO150 capture engine.

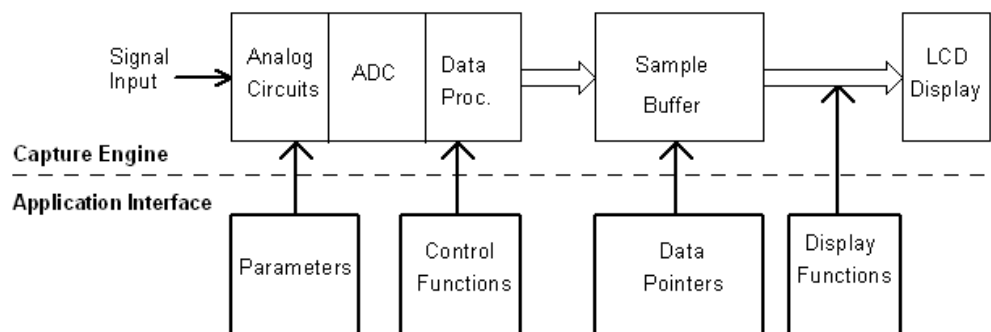


Fig. 1

Input signal is first conditioned by the analogue circuits. It is then fed to analogue-to-digital converter (ADC) where it is sampled and converted to digital values (also called as samples). These digital values are stored to a specific memory (Sample Buffer) and, after certain processing, are displayed on LCD as waveforms.

The data capturing is controlled by a set of parameters including sensitivity, couple, time base, etc. These parameters can be changed with a group of functions. The captured data which is stored in the Sample Buffer is accessible through pre-defined data pointers. The parameters, control functions, data pointers, and display functions constitute the application interface (API) of the capture engine.

2. DSO150 Library

DSO150 library functions are categorized into three major groups, Parameter Access Functions, Capture Control Functions, and Display Functions. The library also provides two data pointers and a function for accesses to the sample buffer. These functions and pointers are explained below one by one.

1) Parameter Access Functions

Accessible parameters of DSO150 include:

- 2 Sensitivity (VSen)
- 2 Couple (Cpl)
- 2 Vertical Position (VPos)
- 2 Time base (TimeBase)
- 2 Horizontal Position (HPos)
- 2 Trigger Mode (TrigMode)
- 2 Trigger Slope (TrigEdge)

- 2 Trigger Level (TrigLvl)
- 2 Record Length (RecLen)
- 2 Vertical Position Offset (VPosOfs)

When a parameter is changed the new value takes effect at the next capture cycle (i.e. the next StartCapture() function is executed).

Syntax	S8 SetVSen(S8 vsen);
Input	<i>vsen</i> – the new sensitivity to be set. Sensitivity can be one of the following values. The numbers in braces are their coded value. VS_20V (0x02) VS_10V (0x03) VS_5V (0x04) VS_2V (0x05) VS_1V (0x06) VS_05V (0x07) VS_02V (0x08) VS_01V (0x09) VS_50mV (0x0A) VS_20mV (0x0B) VS_10mV (0x0C) VS_5mV (0x0D)
Return value	The current sensitivity setting
Remarks	The function changes the sensitivity of the analogue channel.

Syntax	S8 GetVSen(void);
Input	None
Return value	The current sensitivity
Remarks	Returns the current sensitivity internal value.

Syntax	S8 SetCpl(S8 cpl);
Input	<i>cpl</i> – the new couple to be set. Couple can be one of the following values: CP_DC (0x00) CP_AC (0x01) CP_GND (0x02)
Return value	The current internal value of couple setting
Remarks	For DSO150 this function only changes the internal value of the couple setting and has no effect on actual circuit, which is completely dependant on switch SW1 on the analogue board. Internal value will be over-written by actual value whenever position of SW1 is changed.

Syntax	S8 GetCpl(void);
Input	None
Return value	The current internal value of couple setting
Remarks	Returns the current internal value of couple setting.

Syntax	S16 SetVPos(S16 vpos);
Input	<i>vpos</i> – the new vertical position to be set.

	The value range of vertical position is -500 – 500 with 0 corresponding to the vertical centre of waveform window.
Return value	The current vertical position value
Remarks	This function sets the vertical position of waveform display.

Syntax	S16 GetVPos(void);
Input	None
Return value	The current vertical position value
Remarks	This function returns the current vertical position of waveform display.

Syntax	S8 SetTimeBase(S8 timebase);																																																
Input	<p><i>timebase</i> – the new time base to be set. Time base can be one of the following values (numbers in braces are coded value. Numbers in square brackets are corresponding sampling rates):</p> <table border="0"> <tr><td>TB_500s (0x01)</td><td>[0.05sps (20s)]</td></tr> <tr><td>TB_200s (0x02)</td><td>[0.125 sps (8s)]</td></tr> <tr><td>TB_100s (0x03)</td><td>[0.25 sps (4s)]</td></tr> <tr><td>TB_50s (0x04)</td><td>[0.5 sps (2s)]</td></tr> <tr><td>TB_20s (0x05)</td><td>[1.25 sps (0.8s)]</td></tr> <tr><td>TB_10s (0x06)</td><td>[2.5 sps (0.4s)]</td></tr> <tr><td>TB_5s, (0x07)</td><td>[5 sps (0.2s)]</td></tr> <tr><td>TB_2s (0x08)</td><td>[12.5 sps]</td></tr> <tr><td>TB_1s (0x09)</td><td>[25 sps]</td></tr> <tr><td>TB_05s (0x0A)</td><td>[50 sps]</td></tr> <tr><td>TB_02s (0x0B)</td><td>[125 sps]</td></tr> <tr><td>TB_01s (0x0C)</td><td>[250 sps]</td></tr> <tr><td>TB_50ms (0x0D)</td><td>[500 sps]</td></tr> <tr><td>TB_20ms (0x0E)</td><td>[1.25Ksps]</td></tr> <tr><td>TB_10ms (0x0F)</td><td>[2.5Ksps]</td></tr> <tr><td>TB_5ms (0x10)</td><td>[5Ksps]</td></tr> <tr><td>TB_2ms (0x11)</td><td>[12.5Ksps]</td></tr> <tr><td>TB_1ms (0x12)</td><td>[25Ksps]</td></tr> <tr><td>TB_05ms (0x13)</td><td>[50Ksps]</td></tr> <tr><td>TB_02ms (0x14)</td><td>[125Ksps]</td></tr> <tr><td>TB_01ms (0x15)</td><td>[250Ksps]</td></tr> <tr><td>TB_50us (0x16)</td><td>[500Ksps]</td></tr> <tr><td>TB_20us, (0x17)</td><td>[1.25Msps*]</td></tr> <tr><td>TB_10us (0x18)</td><td>[2.5Msps*]</td></tr> </table> <p>* These are equivalent sampling rate.</p>	TB_500s (0x01)	[0.05sps (20s)]	TB_200s (0x02)	[0.125 sps (8s)]	TB_100s (0x03)	[0.25 sps (4s)]	TB_50s (0x04)	[0.5 sps (2s)]	TB_20s (0x05)	[1.25 sps (0.8s)]	TB_10s (0x06)	[2.5 sps (0.4s)]	TB_5s, (0x07)	[5 sps (0.2s)]	TB_2s (0x08)	[12.5 sps]	TB_1s (0x09)	[25 sps]	TB_05s (0x0A)	[50 sps]	TB_02s (0x0B)	[125 sps]	TB_01s (0x0C)	[250 sps]	TB_50ms (0x0D)	[500 sps]	TB_20ms (0x0E)	[1.25Ksps]	TB_10ms (0x0F)	[2.5Ksps]	TB_5ms (0x10)	[5Ksps]	TB_2ms (0x11)	[12.5Ksps]	TB_1ms (0x12)	[25Ksps]	TB_05ms (0x13)	[50Ksps]	TB_02ms (0x14)	[125Ksps]	TB_01ms (0x15)	[250Ksps]	TB_50us (0x16)	[500Ksps]	TB_20us, (0x17)	[1.25Msps*]	TB_10us (0x18)	[2.5Msps*]
TB_500s (0x01)	[0.05sps (20s)]																																																
TB_200s (0x02)	[0.125 sps (8s)]																																																
TB_100s (0x03)	[0.25 sps (4s)]																																																
TB_50s (0x04)	[0.5 sps (2s)]																																																
TB_20s (0x05)	[1.25 sps (0.8s)]																																																
TB_10s (0x06)	[2.5 sps (0.4s)]																																																
TB_5s, (0x07)	[5 sps (0.2s)]																																																
TB_2s (0x08)	[12.5 sps]																																																
TB_1s (0x09)	[25 sps]																																																
TB_05s (0x0A)	[50 sps]																																																
TB_02s (0x0B)	[125 sps]																																																
TB_01s (0x0C)	[250 sps]																																																
TB_50ms (0x0D)	[500 sps]																																																
TB_20ms (0x0E)	[1.25Ksps]																																																
TB_10ms (0x0F)	[2.5Ksps]																																																
TB_5ms (0x10)	[5Ksps]																																																
TB_2ms (0x11)	[12.5Ksps]																																																
TB_1ms (0x12)	[25Ksps]																																																
TB_05ms (0x13)	[50Ksps]																																																
TB_02ms (0x14)	[125Ksps]																																																
TB_01ms (0x15)	[250Ksps]																																																
TB_50us (0x16)	[500Ksps]																																																
TB_20us, (0x17)	[1.25Msps*]																																																
TB_10us (0x18)	[2.5Msps*]																																																
Return value	The current time base setting																																																
Remarks	This function sets the time base setting.																																																

Syntax	S8 GetTimebase(void);
Input	None
Return value	The current timebase setting
Remarks	This function returns the current time base setting.

Syntax	S16 SetHPos(S16 hpos);
Input	<p><i>hpos</i> – the new horizontal position to be set. <i>hpos</i> represents the address for the first sample of currently on-screen data in sample buffer. The value range of hpos is from</p>

	0 to [Record Length – 300]. hpos = 0 means the left-most portion of data is displayed. hpos = [Record Length – 300] means the right-most portion of data is displayed.
Return value	The current horizontal position value
Remarks	This function sets the horizontal position of waveform display.

Syntax	S16 GetHPos(void);
Input	None
Return value	The current horizontal position value
Remarks	This function returns the current horizontal position of waveform display.

Syntax	S8 SetTrigMode(S8 <i>trigmode</i>);
Input	<i>trigmode</i> – the new trigger mode to be set. Trigger mode can be one of the following values: TM_Auto (0x00) TM_Normal (0x01) TM_Single (0x02)
Return value	The current trigger mode setting
Remarks	This function sets the trigger mode.

Syntax	S8 GetTrigMode(void);
Input	None
Return value	The current trigger mode
Remarks	This function returns the current trigger mode setting

Syntax	S8 SetTrigEdge(S8 <i>trigslope</i>);
Input	<i>trigslop</i> – the new trigger slope to be set. Trigger slope can be one of the following values: TE_Falling (0x00) TE_Rising (0x01)
Return value	The current trigger slope setting
Remarks	This function sets the trigger slope.

Syntax	S8 GetTrigEdge(void);
Input	None
Return value	The current trigger slope
Remarks	This function returns the current trigger slope setting

Syntax	S16 SetTrigLvl(S16 <i>triglvl</i>);
Input	<i>triglvl</i> – the new trigger level to be set. The value range of trigger level is from -1200 to 1200 with 0 corresponding to 0V level.
Return value	The current trigger level
Remarks	This function sets the trigger level.

Syntax	S16 GetTrigLvl(void);
Input	None
Return value	The current trigger level
Remarks	This function returns the current trigger level.

Syntax	U16 SetRecLen(U16 <i>reclen</i>);
Input	<i>reclen</i> – the new record length to be set. The value range of record length is from 512 to 1024.
Return value	The current record length
Remarks	This function sets the record length.

Syntax	U16 GetRecLen(void);
Input	None
Return value	The current record length
Remarks	This function returns the current record length.

Syntax	S16 SetVPosOfs(S16 <i>ofs</i>);
Input	<i>ofs</i> – the new vertical position offset. The value range of trigger level is from -500 to 500.
Return value	The current vertical position offset
Remarks	This function sets the vertical position offset. Vertical position offset is used to correct the possible mismatch between 0V trace and vertical position indicator.

Syntax	S16 GetVPosOfs(void);
Input	None
Return value	The current vertical position offset
Remarks	This function returns the current vertical position offset.

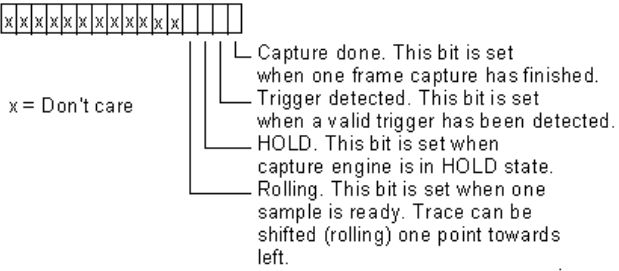
2) Capture Control Functions

The capture engine runs in two modes based on timebase and trigger settings. When time base is 20ms/div or faster, or timebase is 50ms/div or slower but trigger is in NORM or SING mode the capture engine runs in **Frame Mode**, which means waveform display is updated only after the whole capture buffer has been filled up with samples. When time base is 50ms/div or slower *and* trigger is in AUTO mode capture engine runs in **Rolling Mode**. In this mode waveform trace shifts horizontally from right to left at each new sample is captured.

Syntax	void DSO_Init(void);
Parameters	None
Return value	None
Remarks	This function performs the necessary initialization for the capture engine. It must be executed before any other capture control functions are executed and needs to run only once.

Syntax	void StartCapture(void);
Parameters	None
Return value	None
Remarks	This function starts a new capture with the current parameters. When it is called any undergoing capture will be aborted. Under Frame mode (i.e. timebase is 20ms/div or shorter) when this function is successfully finished the sample buffer will be filled with newly captured samples and the Capture Done bit in the status register will be set.

Syntax	void StopCapture(void);
Parameters	None
Return value	None
Remarks	This function stops capture engine until next StartCaptuer() is executed.

Syntax	U16 GetDsoStatus(void);
Parameters	None
Return value	<p>The current capture engine status. The meaning of capture engine status:</p>  <p>Bit 0: Capture Done Under Frame mode this bit is set after StartCapture() function is successfully executed, indicating data in sample buffer is ready for processing. Under Rolling mode this bit does not have meaning.</p> <p>Bit 1: Triggered This bit is set when trigger is detected in the capture just finished. The trigger point is at the middle of sample buffer.</p> <p>Bit 2: HOLD This bit is set when capture engine is in HOLD state.</p> <p>Bit 3: Rolling Under Rolling mode this bit is set when one sample is ready. Trace can be shifted one point towards left.</p>
Remarks	This function returns the current status of capture engine. The status bits are read-only. All status bits except the bit for HOLD are cleared at execution of function StartCapture(). The HOLD bit is changed only by SetHold() and ClrHold() functions.

Syntax	void SetHold(void);
Parameters	None
Return value	None
Remarks	This function puts the capture engine into HOLD state. Waveform refresh will be frozen until ClrHold() is performed.

Syntax	void ClrHold(void);
Parameters	None
Return value	None
Remarks	This function releases capture engine from HOLD state and restores capture.

Syntax	void UpdateTimebase(void);
Parameters	None
Return value	None
Remarks	This function puts time base (which is usually newly changed) in effect immediately instead of waiting for the execution of StartCapture() function.

3) Display Functions

Syntax	void DsoDisplay(void);
Parameters	None
Return value	None
Remarks	This function responds to the display requests (sent by function UpdateDisp()) and renders scope panel (grids), parameters, and traces accordingly. Normally it is placed in the root loop of an application in order to handle requests sent by UpdateDisp() function.

Syntax	void UpdateDisp(U16 disp);
Parameters	<i>disp</i> – display request Display request can be one of the following values or their combinations. Disp_Panel (0x0001) – Update panel display Disp_Param (0x0002) – Update parameter display Disp_Trace (0x0004) – Update trace display Disp_None (0x0008) – Clear screen
Return value	None
Remarks	This function sends various display requests to function DsoDisplay().

Syntax	void CancelDisp(U16 canceldisp);
Parameters	<i>canceldisp</i> – cancel display request. Cancel display request can be one of the following values or their combinations. Cancel_Disp_Panel (0x0001) – Cancel panel display Cancel_Disp_Param (0x0002) –Cancel parameter display Cancel_Disp_Trace (0x0004) –Cancel trace display Cancel_Disp_None (0x0008) –Cancel clear screen
Return value	None
Remarks	This function cancels pending display requests.

Syntax	void Rolling(void);
Parameters	None
Return value	None
Remarks	This function shifts waveform trace towards left by one point (one sample).

Syntax	U8 SetFocus(U8 focus);
Parameters	<i>focus</i> – item to be highlighted (focused) Focus can be one of the following values: FC_VSen (0x00) – Vertical Sensitivity

	FC_Timebase (0x01) – Timebase FC_TrigMode (0x02) – Trigger Mode FC_TrigEdge (0x03) – Trigger Slope FC_TrigLvl (0x04) – Trigger Level FC_HPos (0x05) – Horizontal Position FC_VPos (0x06) – Vertical Position
Return value	None
Remarks	This function is used to select and highlight the specified parameter so as it can be adjusted.

Syntax	U8 GetFocus(void);
Parameters	None
Return value	The currently highlighted (focused) parameter
Remarks	This function returns the currently highlighted parameter.

4) Data Access Pointers and Functions

Captured data are stored in a segment of specific memory (Sample Buffer). Each sample is half-word (16bits) in size. But only the lower 12 bits are significant. The highest 4 bits are always zeros. The size of Sample Buffer is equal to the record length setting. To access the Sample Buffer two pointers and one function can be used.

Definition	U16 *SampleBuf;
Remarks	This is a pointer that always points to the start of sample buffer. You can use this pointer to access captured data when the status bit of CaptureDone is set.

Definition	U16 *CurrentSample;
Remarks	This is a pointer that points to the last sample acquired. Please note that this pointer is only valid when capture engine is running in Rolling Mode. In Frame Mode <i>CurrentSample</i> does not contain a defined value.

Syntax	S16 GetAverage(void);
Parameters	None
Return value	The average of sample values in Sample Buffer
Remarks	This function returns the average value (DC level) of all samples that have been stored in the Sample Buffer.

3. How to Use the Library

1) Conditions must be met for the capture engine to function

In order to have the capture engine function properly the following conditions must be satisfied.

- a) Enable the clocks for TIM1, TIM2, ADC1, and DMA1.
- b) NVIC_Configuration() (supplied in the file Board.c) must be called before entering the main loop. Do not modify the lines for TIM1, ADC1, and DMA Channel 1 interrupt settings inside the function NVIC_Configuration().
- c) Do not use TIM1, TIM2, ADC1, and DMA Channel1 or change their settings. These peripherals are exclusively used by the capture engine.

- d) The function DSO_Init() must be called once prior to any other capture control function can be run.
- e) Supporting functions much be provided and linked to the library (see below).
- f) The following font array must be provided and linked with the library.

FONT_ASC8X16

Definition of the font array can be found in the file Screen.c.

2) Supporting Functions

The following supporting functions are required for the library to run properly.

- a) Display related functions. These functions are defined in the file Screen.c.
 - void ClrScreen(void);
 - void FillRect(S16 x, S16 y, S16 xsize, S16 ysize, U16 color);
 - void PutcGenic(U16 x, U16 y, U8 ch, U16 fgcolor, U16 bgcolor, FONT *font);
 - void PutsGenic(U16 x, U16 y, U8 *str, U16 fgcolor, U16 bgcolor, FONT *font);
- b) I2C related functions and associated macros. These functions and macros are defined in the file Board.c and Board.h.

Functions:

```
void I2C_Start(void);
void I2C_Stop(void);
void I2C_SendByte(U8 byte);
U8 I2C_RecvByte(void);
U8 I2C_CheckAck(void);
void I2C_Ack(void);
void I2C_Nak(void);
void I2C_ReSync(void);
void SetSDA_In(void);
void SetSDA_Out(void);
```

Macros:

```
SetSCL_H
SetSCL_L
SetSDA_H
SetSDA_L
GetSDA
```

3) Build your application with the library

The library is provided with two files, libdso150.a and libdso150.h. They are developed under Sourcery CodeBench Lite from Mentor Graphics. To build it into your application just include the header file libdso150.h into your source codes and link them with libdso150.a. Please take the DSO Shell source code package as an example.

4) Add features to the oscilloscope

You can use the DSO Shell source code package as a base and add other features you like on top of it. Just keep the conditions met as stated in 1) and you are free to modify other codes.

Revision History

Revision	Date	Summary
v01	2017.05.08	Draft