INSTRUCTION MANUAL

TDRSDK Software Development Kit for TDR100

8/05

TDRSDK Table of Contents

PDF viewers note: These page numbers refer to the printed version of this document. Use the Adobe Acrobat® bookmarks tab for links to specific sections.

1. Overview 1

1.1 Purpose of TDRSDK 1
1.2 Requirements 1
1.3 Description of TDRSDK package 1

2. TDRSDK Description 2

2.1 Description of TDRCOM.DLL 2
2.2 Task complete or data available notification after command issued 2
2.2.1 Using the Call-back method 2
2.2.2 Using the State Machine method 2

3. TDR100 Command Set 2

3.1 Commands for error odes, serial port control, call-back and state machine queries 3
3.2 Commands to retrieve TDR100 waveform parameter values 5
3.3 Commands for setting TDR100 parameters 5
3.4 Commands to execute TDR100 measurements and to retrieve data 8

4. Sample Programs 15

Appendices

A. TDR100 Command Protocol A-1

B. TDR100 Response Protocol B-1

TDRSDK Software Development Kit for TDR100

1. Overview

1.1 Purpose of TDRSDK

TDR100 Software Development Kit, TDRSDK, allows users to create custom applications that communicate directly with the TDR100. This is accomplished by using a dynamically linked library (DLL). The DLL functions as a transparent interface between a 32-bit personal computer (PC) and the TDR100. The DLL allows access to the entire TDR100 command set. Applications can be written using C++, Delphi (Pascal) and Visual Basic including Microsoft Excel.

1.2 Requirements

The TDRCOM.DLL is designed to run in the Microsoft 32-bit Windows environment, which presently includes Windows97, Windows98, WindowsNT, Windows2000 and WindowsXP. The TDRCOM.DLL is designed solely for the purpose of communicating with Campbell Scientific, Inc. TDR100 Time Domain Reflectometer over direct connect RS232 serial communications at one of three baud rates, 9600bps, 19.2kbps, 57.6kbps (default).

1.3 Description of TDRSDK package

The TDRSDK software package includes the following on a compact disk:

1. Executable install program for installing TDRSDK on the computer to be used.

2. Dynamically Linked Library (DLL) file that is installed when the setup application is run. The DLL contains calls TDR100 command set functions for setting parameter, retrieving data and control functions.

3. Operating manual

4. Example programs in C++, Delphi (Pascal) and Visual Basic including Microsoft Excel.

2. TDRSDK Description

2.1 Description of TDRCOM.DLL

TDRCOM.DLL is a dynamically linked library, DLL, comprised of functions that allow control and monitoring of a TDR100 using PC serial communications. The DLL functions are called by a program written by the user, and the TDR100 communication protocol is transparent. Each DLL function is defined with a name and a number. The functions perform a range of TDR100 tasks including setting measurement parameters and collecting waveform data.

2.2 Task complete or data available notification after command issued

To enhance PC performance, query methods are used to determine if information is available from the TDR100. Many of the TDRCOM.DLL API functions only initiate commands to the TDR100 and do not continue to execute while the TDR100 is running the assigned task. The Call-back and State Machine methods are used to determine when the TDR100 has completed a task.

2.2.1 Using the Call-back method

When programming in C++ or Pascal, a call-back procedure can be used to notify the user program that data is available from the TDR100. The call-back procedure uses 2 two parameters, (1) DLL command number and (2) error code, to determine which command to execute and which part of the user program to notify when data is available from TDR100. DLL function RegCallBack is used to pass the address of the procedure in the user program that will be notified when data available.

2.2.2 Using the State Machine method

The State Machine method is used in Visual Basic (including Excel). The main part of the DLL will be implemented as a state machine, incrementally stepping through short blocks of functionality (states). At regular time intervals an Application Program Interface (API) function queries the state machine to determine if data is available from the TDR100. When the data is signaled to be available, a separate command is issued to retrieve the data from the DLL to the user program

3. TDR100 Command Set

The TDR100 command set is also called the TDR100COM Application Program Interface (API). Each command has an associated command number that is used in DLL execution.

3.1 Commands for error codes, serial port control, call-back and state machine queries

1. ReadErrorCode

Allows the calling application to get an error string by specifying the API function number, and error code. This is used to determine reason for error on failure of a DLL command.

int ReadErrorCode(int apiNumber, int errCode, char errStr, int strSiz)

Parameters: apiNumber: API function number

errCode: return non-zero value from each API function

errStr: describes Error in a string

strSiz: buffer size for errStr.

Return Codes: 0 = Success

1 = Unknown Error Code

2. SetCommPort

Allows the calling application to specify the communications port and the communication baud rate. The comm port values are limited to those available on the machine. The baud rate is limited to 9600bps, 19200bps, or 57600bps. The default baud rate value is 57600bps. Changing baud rate requires repositioning jumpers inside TDR100. See TDR100 Operating Manual.

int SetCommPort(short portNbr, int baudRate)

Parameters: portNbr: Communications port number, i.e. 1 for Com1

baudRate: Communications baud rate, i.e. 57600

Return Codes: 0 = Success

1 = Parameter out of Range

3. CloseCommPort

Allows the calling application to close the current communications port previously opened using 'SetCommPort';

int CloseCommPort

Parameters: None

Return Codes: 0 = Success

1 = Failure

4. RegCallBack

Allows the calling application to register a call-back procedure with the DLL. This call-back procedure will allow the DLL to notify the calling application when a command response is received from the TDR100. The call-back procedure will have parameters to report command type and success/failure. The procedure cannot be an object method. This method does not work with VB or VBA. See examples.

int RegCallBack (int cmdType, int errCode))

Parameters: CBfunc: Pointer to procedure that will be called (if required), upon receipt of a TDR100 command response.

cmdType: API function numbers

errCode: error code for the cmdType.

Return Codes: 0 = Success

1 = Failure

---

5. GetDLLver

Allows the calling application to obtain the DLL version.

int GetDLLver(char Ver, int iStrSize)

Parameters: Ver: Pointer to a string buffer. Upon successful return, will contain the version in the form '1.0'.

iStrSize: Integer, size of pVer string buffer

Return Codes: 0 = Success

1 = String buffer too small

---

6. QuerySM

Allows the calling application to check the current state of the state machine in the DLL. This function returns immediately. An application is required to make the request according to the current state; otherwise, an error code will be returned. See the VBA examples.

int QuerySM(int FuncNbr)

Parameters: FuncNbr: pointer to integer value, returns function number being processed or data available from.

Return Codes: 0 = Idle

n = still processing request

100 = Get data successfully

101 = CRC16 check error

102 = Wrong response

---

7. ReadSM_State

Allows the calling application to translate state number to a state string.

int ReadSM_State(int stateNbr, char stateStr, int strSiz)

Parameters: stateNbr: state numbers.

stateStr: state described in a string.

strSiz: buffer size for stateStr.

Return Codes: 0 = Success

1 = Failure

3.2 Commands to retrieve TDR100 waveform parameter values

The waveform parameter values are (1) relative propagation velocity, Vp, (2) waveform averaging, (3) number of data points in waveform, (4) distance to beginning of waveform, (5) length of waveform window, (6) probe rod length, (7) probe offset, and (8) cell constant or Kp.

When a 'request' command is issued, Callback or State Machine is used to determine if requested data is available. When Callback or State Machine is satisfied the data is written to the DLL. A 'get' command is then used to get the data from the DLL to the API.

8. ReqSettings

Allows calling application to request current TDR100 parameter settings. If no error occurred, current settings will be available from the DLL, and can be retrieved using the GetSetting (#9) API call after you receive a call back or get a state 100. Result: Settings values are sent by the TDR100 to the DLL.

int ReqSettings

Parameters: None

Return Codes: 0 = Success

1 = Failure

9. GetSetting

Allows calling application to get TDR100 parameter settings stored in DLL. This command, if successful, will return a record containing a setting. This command must be called repeatedly to get all 9 settings.

int GetSetting(int recNbr, float Fval)

Parameters: recNbr: number counting from 1 in a data structure.

Fval: Pointer to a floating point value.

Return Codes: 0 = Success

1 = Failure

3.3 Commands for setting TDR100 parameters

10. SetVpSetting

Allows calling application to set the TDR100 relative propagation velocity, Vp. When transaction is complete, the call-back procedure will be called (if required), or the state machine can be queried until non-zero state is obtained.

int SetVpSetting(float fValue)

Parameters: fValue: Float, value with which to set TDR100 Vp setting.

Return Code: 0 = Success

1 = Failure

11. SetAvgSetting

Allows calling application to set the number of waveforms to be averaged for final waveform. When transaction is complete, the call-back procedure will be called (if required), or the state machine can be queried until non-zero state is obtained.

int SetAvgSetting(float fValue)

Parameters: fValue: Float, value with which to set TDR100 'Average' setting.

Return Code 0 = Success

1 = Failure

12. SetPointsSetting

Allows calling application to set the TDR100 number of points in waveform. When transaction is complete, the call-back procedure will be called (if required), or the state machine can be queried until non-zero state is obtained.

int SetPointsSetting(float fValue)

Parameters: fValue: Float, value with which to set TDR100 'Points' setting.

Return Code: 0 = Success

1 = Failure

13. SetDistanceSetting

Allows calling application to set the TDR100 distance to beginning of waveform. When transaction is complete, the call-back procedure will be called (if required), or the state machine can be queried until non-zero state is obtained.

int SetDistanceSetting(float fValue)

Parameters: fValue: Float, value with which to set TDR100 'Distance' setting.

Return Code: 0 = Success

1 = Failure

14. SetWndLgtSetting

Allows calling application to set the TDR100 waveform window length. When transaction is complete, the call-back procedure will be called (if required), or the state machine can be queried until non-zero state is obtained.

int SetWndLgtSetting(float fValue)

Parameters: fValue: Float, value with which to set TDR100 'Window Length' setting.

Return Code: 0 = Success

1 = Failure

15. SetProbeLgtSetting

Allows calling application to set the TDR100 probe rod length. When transaction is complete, the call-back procedure will be called (if required), or the state machine can be queried until non-zero state is obtained.

int SetProbeLgtSetting(float fValue)

Parameters:fValue: Float, value with which to set TDR100 'ProbeLength' setting.

Return Code: 0 = Success

1 = Failure

16. SetProbeOffSetting

Allows calling application to set the current TDR100 'ProbeOffset' setting using the ':SPRO' type-1 command. When transaction is complete, the call-back procedure will be called (if required) with success reported.

int SetProbeOffSetting(float fValue)

Parameters: fValue: Float, value with which to set TDR100 'ProbeOffset' setting.

Return Code: 0 = Success

1 = Failure

17. SetCellConstSetting

Allows calling application to set the TDR100 Probe Cell Constant or Kp. When transaction is complete, the call-back procedure will be called (if required), or the state machine can be queried until non-zero state is obtained.

int SetCellConstSetting(float fValue)

Parameters: fValue: Float, value with which to set TDR100 'Cell Constant' setting.

Return Code: 0 = Success

1 = Failure

18. This function is not available

19. SetMuxSetting

Allows calling application to set the SDMX50 multiplexer channel. When transaction is complete, the call-back procedure will be called (if required), or the state machine can be queried until non-zero state is obtained.

The format is a two digit floating point value with the first digit the multiplexer level (1, 2 or 3) and the second digit the channel (1 – 8).

int SetMuxSetting(float fValue)

Parameters:

fValue: Float, value with which to set TDR100 'SDMX50 channel' setting.

Return Code: 0 = Success

1 = Failure

3.4 Commands to execute TDR100 measurements and to retrieve data

When a 'request' command is issued, Callback or State Machine is used to determine if requested data is available. When Callback or State Machine is satisfied the data is written to the DLL. A 'get' command is then used to get the data from the DLL to the API.

20. ReqCalcCellConst

Allows calling application to request the TDR100 to calculate the probe cell constant or Kp. When transaction is complete, the call-back procedure will be called (if required), or the state machine can be monitored to detect completion. If no error occurred, cell constant will be available from the DLL and can be retrieved using the GetCalcCellConst (#21) API call. Result: TDR100 sends calculated cell constant to the DLL.

int ReqCalcCellConst(float WaterTemp)

Parameters: WaterTemp: Floating point value, Temperature of DI water surrounding probe.

Return Code: 0 = Success

1 = Failure

21. GetCalcCellConst

Allows calling application to get TDR100 cell constant (Kp) value stored in DLL. This command, if successful, will return a floating point variable containing the value.

int GetCalcCellConst(float pValue)

Parameters: pValue: Pointer to a floating point variable containing cell const

Return Codes: 0 = Success

1 = Failure

22. ReqCalVoltage

Allows calling application to request the current TDR100 calibration voltage. When transaction is complete, the call-back procedure will be called (if required), or the state machine can be monitored to detect completion. If no error occurred, current 'Calibration Voltage' will be available from the DLL and can be retrieved using the GetCalVoltage (#23) API call. Result: TDR100 send the calibration voltage to the DLL.

int ReqCalVoltage

Parameters: None

Return Code: 0 = Success

1 = Failure

23. GetCalVoltage

Allows calling application to get TDR100 calibration voltage value stored in DLL. This command, if successful will return a floating point variable containing the value.

int GetCalVoltage(float pValue)

Parameters: pValue: Pointer to a floating point variable.

Return Codes: 0 = Success

1 = Failure

24. ReqConductivity

Allows calling application to request the TDR100 bulk electrical conductivity measurement. When transaction is complete, the call-back procedure will be called (if required), or the state machine can be monitored to detect completion. If no error occurred, current 'Conductivity' value will be available from the DLL and can be retrieved using the GetConductivity (#25) API call. Result: TDR100 sends the conductivity value to the DLL.

int ReqConductivity

Parameters: None

Return Code: 0 = Success

1 = Failure

25. GetConductivity

Allows calling application to get TDR100 measured bulk electrical conductivity value stored in DLL. This command, if successful will return a floating point variable containing the value.

int GetConductivity(float pValue)

Parameters: pValue: Pointer to floating point variable

Return Codes: 0 = Success

1 = Failure

28. ReqLastDeri

Allows calling application to request the most recently calculated derivative waveform stored in TDR100. This derivative is calculated from the most recent waveform data. When transaction is complete, the call-back procedure will be called (if required), or the state machine can be monitored to detect completion. If no error occurred, the last derivative' will be available from the DLL and can be retrieved using the GetDeriWF (#29) API call. Result: The last derivative acquired by the TDR100 will be sent to the DLL.

int ReqLastDeri

Parameters: None

Return Code: 0 = Success

1 = Failure

29. GetDeriWF

Allows calling application to get a single TDR100 derivative waveform data point from an array of values stored in DLL. The index parameter specifies which value in the derivative waveform array to retrieve. This command, if successful will return a floating point variable containing the value. The DLL must be called as many times as there are waveform data points (value set in API #12). Result: One value of the Derivative Waveform (array of values) will be passed from the DLL to the Application. Note: This command is identical to command 35, GetNewDeri.

int GetLastDeri(float pValue, int Index)

Parameters: pValue: Pointer to floating point variable

Index: Integer, index into 1-based derivative array

Return Codes: 0 = Success

1 = Failure

30. ReqLastWaveform

Allows calling application to request the waveform currently stored in TDR100. New waveform acquisition is not performed by TDR100. When transaction is complete, the call-back procedure will be called (if required), or the state machine can be monitored to detect completion. If no error occurred, current 'Last Waveform' will be written to the DLL and can be retrieved using the GetWaveform (#31) API call. Result: The TDR100 will send the stored waveform to the DLL.

int ReqLastWaveform

Parameters: None

Return Code: 0 = Success

1 = Failure

31. GetWaveform

Allows calling application to get a single TDR100 waveform value from an array of values stored in DLL. Index parameter specifies which value in the Waveform array to retrieve. This command, if successful will return a floating point variable containing the value. Result: One value of the Waveform (array of values) will be passed from the DLL to the Application.

int GetWaveform(float pValue, int Index)

Parameters: pValue: Pointer to floating point variable

Index: Integer, index into 1-based waveform array

Return Codes: 0 = Success

1 = Failure

32. ReqMoisture

This command might better be called ReqLaL instead of ReqMoisture. It allows the calling application to request the TDR100 to compute the ratio of the apparent rod length to the actual rod length and correct the ratio for probe offset. When transaction is complete, the call-back procedure will be called (if required), or the state machine can be monitored to detect completion. If no

error occurred, La/L corrected for probe offset will be written to the DLL and can be retrieved using the GetMoisture (#33) API call. Result: The TDR100 sends the moisture value to the DLL.

int ReqMoisture

Parameters: None

Return Code: 0 = Success

1 = Failure

---

33. GetMoisture

This command might better be called GetLaL instead of GetMoisture. Allows calling application to get TDR100 La/L corrected for probe offset stored in DLL. This command, if successful, will return a floating point variable containing the value. Result: The moisture value is passed from the DLL to the Application.

int GetMoisture(float pValue)

Parameters: pValue: Pointer to floating point variable

Return Codes: 0 = Success

1 = Failure

---

34. ReqNewDeri

Allows calling application to request that the TDR100 acquire a new waveform and calculate the derivative waveform and return it to the DLL. When transaction is complete, the call-back procedure will be called (if required), or the state machine can be monitored to detect completion. If no error occurred, current 'New Derivative' value will be available from the DLL and can be retrieved using the GetDeriWF (#29) API call. Result: The TDR100 will acquire a new derivative waveform and send it to the DLL.

int ReqNewDeri

Parameters: None

Return Code: 0 = Success

1 = Failure

---

35. GetNewDeri

Allows calling application to get a single TDR100 derivative waveform data point from an array of values stored in DLL. The index parameter specifies which value in the derivative waveform array to retrieve. This command, if successful will return a floating point variable containing the value Result: One value of the Derivative Waveform (array of values) will be passed from the DLL to the Application. Note: This command is identical to command 29, GetDeriWF.

int GetLastDeri(float pValue, int Index)

Parameters: pValue: Pointer to floating point variable

Index: Integer, index into 1-based derivative array

Return Codes: 0 = Success

1 = Failure

---

36. ReqWFnocal

Allows calling application to request that the TDR100 acquire a new waveform without performing a calibration first. When transaction is complete, the call-back procedure will be called (if required), or the state machine can be monitored to detect completion. If no error occurred, waveform values (not calibrated) will be available from the DLL and can be retrieved using the GetWaveform (#31) API call. Result: TDR100 will acquire a new waveform without calibration and then send the waveform to the DLL.

int ReqWFnocal

Parameters: None

Return Code: 0 = Success

1 = Failure

37. ReqWaveform

Allows calling application to request that the TDR100 acquire a new waveform and make available to the DLL. When transaction is complete, the call-back procedure will be called (if required), or the state machine can be monitored to detect completion. If no error occurred, current waveform values will be available from the DLL and can be retrieved using the GetWaveform (#31) API call. Result: Tdr100 will acquire a new waveform and then send the waveform to the DLL.

int ReqWaveform

Parameters: None

Return Code: 0 = Success

1 = Failure

Commands 38 through 41 are not currently used

42. ReqVariables

Allows calling application to request the TDR100 values derived from the algorithm for determining (1) start, (2) end, and (3) RMS value. When transaction is complete, the call-back procedure will be called (if required), or the state machine can be monitored to detect completion. If no error occurred, values will be available from the DLL and can be retrieved using the GetVariables API call. Result: TDR100 sends the three variables to the DLL.

int ReqVariables

Parameters: None

Return Code: 0 = Success

1 = Failure

43. GetVariables

Allows calling application to get TDR100 values from algorithms that are stored in DLL. This command, if successful will return a floating point variable containing the value. Result: Variables are passed form DLL to Application.

int GetVariables(float pStart, float pEnd, float pRMS)

Parameters:	pStart: Pointer to floating point variable, Start
	pEnd: Pointer to floating point variable, End
	pRMS: Pointer to floating point variable, RMS
Return Codes:	0 = Success
	1 = Failure

44. ReqVerSig

Allows calling application to request the current TDR100 versions and signatures for BootCode and Operating System'. When transaction is complete, the call-back procedure will be called (if required), or the state machine can be monitored to detect completion. If no error occurred, current version and signature values will be available from the DLL and can be retrieved using the GetVerSig API call. Result: TDR100 sends the two version values and the two signature values to the DLL.

int ReqVerSig

Parameters:	None
Return Code:	0 = Success
	1 = Failure

45. GetVerSig

Allows calling application to get TDR100 version and signature values stored in DLL. This command, if successful will return a floating point variable containing the value. Result: Version & Signature values are passed from DLL to Application

int GetVerSig(flaot pBCver, float pBCsig,float pOSver, float pOSsig)

Parameters:	pBCver: Pointer to floating point variable, Boot Code version
	pBCsig: Pointer to floating point variable, Boot Code signature
	pOSver: Pointer to floating point variable, OS version
	pOSsig: Pointer to floating point variable, OS signature
Return Codes:	0 = Success
	1 = Failure

46. AcqWFnocal

Allows calling application to initiate the acquisition of a new waveform without first doing calibration. When transaction is complete, the call-back procedure will be called (if required), or the state machine can be monitored to detect completion. If no error occurred, the waveform is not immediately available but can be retrieved using 'ReqLastWaveForm' (#30) to get the waveform sent to the DLL. Then use 'GetWaveform' (#31) to move the waveform from the DLL to the Application. Result: Waveform is acquired, without calibration, but is NOT sent to the DLL.

int AcqWFnocal

Parameters: None

Return Code: 0 = Success

1 = Failure

47. AcqWaveform

Allows calling application to initiate the acquisition of a new waveform with calibration. When transaction is complete, the call-back procedure will be called (if required), or the state machine can be monitored to detect completion. If no error occurred, the waveform is not immediately available but can be retrieved using 'ReqLastWaveForm' (#30) to get the waveform sent to the DLL. Then use 'GetWaveform' (#31) to move the waveform from the DLL to the Application. Result: Waveform is acquired but is NOT sent to the DLL.

int AcqWaveform

Parameters: None

Return Code: 0 = Success

1 = Failure

48. SetPwrOff

Allows calling application to set the analog power in the TDR100 to the off state. After this function is called the analog section of the TDR100 is in a low power state. When transaction is complete, the call-back procedure will be called (if required), or state machine status is monitored for reporting of any error code. Result: TDR100 puts its analog section in a low power state.

int AcqWFnocal

Parameters: None

Return Code: 0 = Success

1 = Failure

50. ReqCancel

Allows calling application cancel its last command. When the cancellation transaction is complete, the application can continue its next command.

int ReqCancel

Parameters: None

Return Code: 0 = Success

1 = Failure

4. Sample Programs

Directory example applications contain four subdirectories with examples in the four languages currently supported by TDRSDK. The examples are designed to help make the initial connection between the API and the TDR100. Code for typical tasks is included in the examples and can help get the API get the best measurements and save startup time.

Appendix A. TDR100 Command Protocol

:XXXX ffffffffHLcr	Type 1 commands to set a variable (space between command and value)
:XXXXHLcr	Type 2 & 3 commands

:	Start of Command
XXXX	Command ASCII Characters
ffffffff	Floating point number in ASCII characters preceded by a space. No maximum number of characters, a single floating point value.
HL	Hexadecimal representation of 8 bit Checksum which includes all characters after ':' and before the checksum. Computed by summing each character and discarding any carries.
	H = high order bits, L = low order bits.
cr	Carriage Return
:XXXX	One of the following 4 character commands.

Type 1 Commands (Set a variable)

:S_VP	n	Set Vp value
:SDIS	n	Set Distance (Cable Length)
:SMAX	n	Set maximum display value
:SMIN	n	Set minimum display value
:SMUX	n	Set SDMX50 channel
:SNAV	n	Set Number of Averages
:SPCC	n	Set Probe Cell Constant
:SPNT	n	Set Number of Points
:SPRL	n	Set Probe Length
:SPRO	n	Set Probe Offset
:SSMO	n	Set Smooth Value
:SWLN	n	Set Window Length
:CCCC	n	Calculate Cell Constant (special type-1 cmd that returns value like type-2 cmd)

Type 2 Commands (Gets Values)

CMD	HL	Results
:DUMP	36	Get Vp, Average, Points, Distance, Window Length, Probe Length, Probe Offset, Cell Constant, Smooth Factor.
:GCAL	17	Get Calibration Voltage Value
:GCON	27	Get Conductivity Value
:GDTS	32	Get Distance to Short Value
:GLDR	29	Get Last Derivative Value
:GLWF	30	Get Last Waveform (no new acquisition)
:GMOS	36	Get Moisture Value
:GNDR	28	Get New Derivative waveform
:GNWA	20	Get Waveform, no calibration
:GRLN	33	Get last calculated Cable Reference Length Value
:GTIM	31	Get Time Values (Power Delay, Acq. Waveform, Moisture Result)
:GVAR	30	Get Variables (Start, End, RMS)
:GVER	34	Get Boot Code (version, signature), Operating System(version, signature)
:GWAV	35	Acquire and return Waveform

Type 3 Commands (No values set or returned)

CMD	HL	Results
:ABRT	29	A NOP instruction which can be used to Abort the current command.
:ANWA	27	Acquire Waveform with Calibration
:AWAV	2F	Acquire Waveform
:RSET	3E	Retrieve Setup
:SOFF	2E	Set Analog Power Off
:SRLN	3F	Set Cable Reference Length - for liquid level measurements
:SSET	3F	Save Setup

Commands are aborted if the start of a new command is received before the finish of current command.

Appendix B. TDR100 Response Protocol

|S| quoted data |E|

S Start Character 0x3A : (colon)

E End Character 0x0D (carriage return)

“ Quote Character 0x22 “ (double quote)

Quoted Data includes

Unquoted Data

CRC-16

Quoted data is the same as unquoted data except that it doesn’t allow the bytes (characters) colon, carriage return or double quote. When those values are encountered in the data they are replaced with the quote character <”> followed by the 2’s complement of the data. i.e.

```
unsigned char data;

data = -data;
```

Quoted Replacements

: 0x22C6

cr 0x22F3

“ 0x22DE

Unquoted Formats

Unquoted data adheres to the following formats with a maximum of 8200 bytes followed by the CRC-16.. The first byte of unquoted data is “:”, and the last byte is “cr”. The CRC is computed on the unquoted data without the first byte , the last byte, and CRC16. CRC16 is in the Big-Endian format. Since the CRC might have quoted characters, the CRC cannot be compared until the unquoted data is retrieved.

Error Response

|!|NN|

! ASCII '!' to indicate an Error Packet

NN	**Error Numbers in ASCII**
'01'	Bad Checksum
'02'	Illegal Cmd Format, Not Defined
'03'	No Valid Letters Or Numbers
'04'	Could Not be Parsed
'05'	Command Not Identified
'06'	Command Not Recognized
`07'	Calibration Unsuccessful
'08'	Extra Period (terminal mode)
'09'	No Reference Cable Length
'10'	Value Out of Range
'11'	Timeout - Cable Short Not Found
'12'	Timeout Waiting for Data
'13'	Exponent Not Defined
'14'	No Command Defined for Output
'15'	Bad Data
'16'	Bad Moisture Calculation
'17'	Could Not Detect Liquid Level
'18'	Incorrect Mux Address or Channel
'19'	Unable to Locate Pulse
'20'	Could Not Measure Baseline
'21'	Couldn't Measure Top of Pulse
'22'	Unknown Internal Error
'69'	Command Decode Error
'70'	Unknown Error
'71'	Device Write not accepted
'72'	Unknown Error
'73'	IOPOLL: Timeout
'74'	WRITE: Data not written
'75'	WRITE: Address not accepted
'76'	WRITE: Write not accepted
'77'	Unknown Error
'78'	READ: Read not accepted
'79'	READ: Address not accepted
'80'	Unknown Error

Acknowledge Response (Response to Type 1 and Type 3 Commands)

|$|CCCC|

$ ASCII '$' to indicate Acknowledgment

C The Four ASCII command characters that followed the ':'

Value Response (Response to Type 2 Commands)

|#|CCCC|0xXXXX|data bytes up to 8192

ASCII '#' to indicate a value response

C The four ASCII command characters that followed the ':'

X data in 4 byte IEEE floating point in the Big-Endian format

Campbell Scientific Companies

Campbell Scientific, Inc. (CSI)
815 West 1800 North
Logan, Utah 84321
UNITED STATES
www.campbellsci.com
info@campbellsci.com

Campbell Scientific Africa Pty. Ltd. (CSAf)
PO Box 2450
Somerset West 7129
SOUTH AFRICA
www.csafrica.co.za
sales@csafrica.co.za

Campbell Scientific Australia Pty. Ltd. (CSA)
PO Box 444
Thuringowa Central
QLD 4812 AUSTRALIA
www.campbellsci.com.au
info@campbellsci.com.au

Campbell Scientific do Brazil Ltda. (CSB)
Rua Luisa Crapsi Orsi, 15 Butantã
CEP: 005543-000 São Paulo SP BRAZIL
www.campbellsci.com.br
suporte@campbellsci.com.br

Campbell Scientific Canada Corp. (CSC)
11564 - 149th Street NW
Edmonton, Alberta T5M 1W7
CANADA
www.campbellsci.ca
dataloggers@campbellsci.ca

Campbell Scientific Ltd. (CSL)
Campbell Park
80 Hathern Road
Shepshed, Loughborough LE12 9GX
UNITED KINGDOM
www.campbellsci.co.uk
sales@campbellsci.co.uk

Campbell Scientific Ltd. (France)
Miniparc du Verger - Bat. H
1, rue de Terre Neuve - Les Ulis
91967 COURTABOEUF CEDEX
FRANCE
www.campbellsci.fr
campbell.scientific@wanadoo.fr

Campbell Scientific Spain, S. L.
Psg. Font 14, local 8
08013 Barcelona
SPAIN
www.campbellsci.es
info@campbellsci.es

Please visit www.campbellsci.com to obtain contact information for your local US or International representative.