Development kit for DATAMAN 520
Programmer's Guide
Version 2.00

Thank you for choosing the Dataman 520 Series Ocilloscope with development kit. We believe it will meet your expectations.

For any further information or consultations, please contact us via phone or preferably e-mail on the following address:

Address:

Dataman Programmers Ltd
Station Road
Maiden Newton
Dorset
DT2 0AE
United Kingdom

Phone:

Sales/General information: +44 (0) 1300 320719
Technical support: +44 (0) 1300 322903

Fax:

All Enquiries: +44 (0) 1300 321012

Internet:

URL:	http://www.dataman.com/
e-mail:	support@dataman.com - technical support
	sales@dataman.com - sales
	info@dataman.com - other information

Contents

1. Basic information

1.1. Development kit contents

Development kit software package has following structure:

Directory	Contents
Examples\BorlandC	Example (Borland C++ 5.01)
Examples\Delphi	Example (Inprise Delphi 5.0)
Examples\VBasic	Example (Microsoft Visual Basic 6.0)
Include	Header files
Bin\Windows	Libraries for Win98, Win2000 and WinXP

Table 1.1.1. –DK contents

1.2. Development kit usage

This chapter contains information about setup necessary to use DK. It is important to have em52X USB port driver installed before DK usage.

1.2.1. Borland C++ and Borland C++ Builder

Add header file DevKitConsts52X.h to your project. Dynamic libraries DevKit52X.dll and M52XDrv.dll are required during runtime.

1.2.2. Inprise Delphi

Add unit DevKitConsts52X.pas to your project. Dynamic libraries DevKit52X.dll and M52XDrv.dll are required during runtime.

1.2.3. Microsoft Visual Basic

Add module DevKitConsts52X.bas to your project. Dynamic libraries VBDevKit52X.dll and M52XDrv.dll are required during runtime.

2. Controlling the device

This chapter contains information about the device initialization, setting device parameters and measurement using development kit.

2.1. Driver loading

First of all, it is necessary to load the device driver using LoadDriver function.

int LoadDriver(void);

Appropriate error code is returned, if device driver loading fails.

2.2. Device initialization

After successful device driver loading, it is time to perform device initialization.

Function InitHardware initializes device.

*int InitHardware(void *DeviceContextBuffer);*

As a parameter, pointer to the DeviceContextBuffer structure is required. After successful initialization, this structure will be filled with device dependent parameters (such as memory size, available timebases etc.)

If the InitHardware returns ERROR_OK, device is ready for communication.

Now it is possible to set the device parameters with proper functions.

2.3. Data acquisition loop

Application should enter data acquisition loop after device initialization. Data acquisition is controlled by function Data. (fig. 2.3.1. for block diagram).

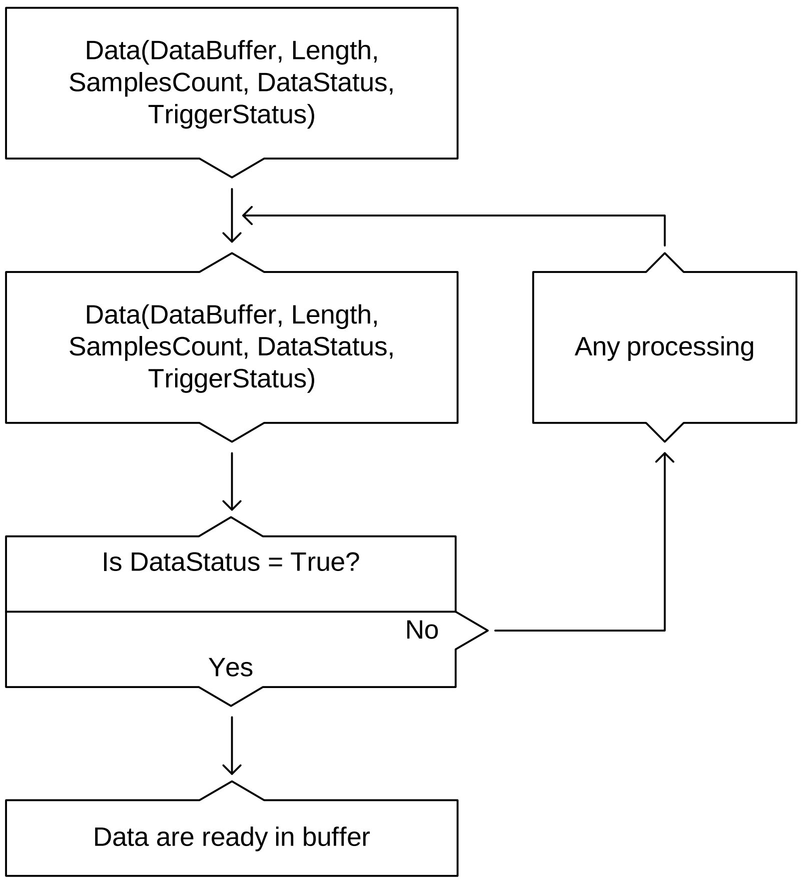
Data(DataBuffer, Length, SamplesCount, DataStatus, TriggerStatus)
Data(DataBuffer, Length, SamplesCount, DataStatus, TriggerStatus)
Any processing
Is DataStatus = True?
No
Yes
Data are ready in buffer

Figure 2.3.1. – Data acquisition loop

```
int SamplesCount, TriggerStatus;
 bool DataStatus;
unsigned short int DataBuffer[8192];
 while (1)
 {
        Data(DataBuffer, 8192, SamplesCount, DataStatus, TriggerStatus);
        if (DataStatus)
        {
                for (int i = 0; i < 8192; i++)
                {
                        fBufferA[i] = DataBuffer[i] & 0xff;
                        fBufferB[i] = (DataBuffer[i] >> 8) & 0xff;
                }
                UpdateWithNewData();
        }
  }
return 0;
```

New data are stored in the DataBuffer each time the Data function returns with successful data acquisition.

2.4. Application termination

The application should call DeInitHardware function prior its termination to deinitialize the device.

3. Reference

This chapter describes DK functions.

Initialization functions

LoadDriver
InitHardware
DeInitHardware

Trigger functions

SetTrigger
SetTriggerCount
SetTriggerLength
SetTriggerMode
SetAfterTriggerSamplesCount
SetBeforeTriggerDelay
SetPrimaryTriggerSource
SetSecondaryTriggerSource
InvertPrimaryTriggerSource
InvertSecondaryTriggerSource

Time base function

SetTimeBase

Vertical control functions

SetProbe
SetRange
SetDC
SetVert

Data acquisition functions

Data
EnableWaveformConformityDetection
SetDigitalShielding
SetShapePrediction

Other functions

CheckConnection
CompensationGenerator
GetSDKVersion
GetDeviceDriverVersion
GetUSBDriverVersion

Visual Basic functions

GetDensities
GetDeviceID
GetDeviceMemorySize
GetDevicePointsPerDivider
GetTimeBases

3.1. Initialization functions

DeInitHardware

Deinitializes the device and releases the device driver library.

Declaration:

typedef _export _stdcall void (*fDeInitHardware)(void);
TDeInitHardware = procedure; stdcall;
Public Declare Sub DeInitHardware Lib "VBDevKit52X.dll" ()

Parameters:

None

Return value:

None

InitHardware

Performs the device initialization.

Declaration:

typedef _export _stdcallint (*fInitHardware) (void*DeviceContextBuffer);
TInitHardware = function (DeviceContextBuffer: Pointer): Integer; stdcall;
Public Declare Function InitHardware Lib "VBDevKit52X.dll" () As Long

Parameters:

DeviceContextBuffer – Pointer to the device context data structure. It is filled with device dependent values after successful initialization.

Returned value:

ERROR_OK – Initialization successful
ERROR_USB_DRIVER_NOT_LOADED – USB driver loading failed
ERROR_DEVICE_CONFIGURATION_FAILED – Unable to configure FPGA
ERROR_DEVICE_CALIBRATION_BROKEN – Calibration data in device are corrupted
ERROR_DEVICE_DRIVER_NO_ENTRY_POINT – Device driver is not compatible with DK
ERROR_DEVELOPMENT_KIT_NOT_ENABLED – DK cannot be used with connected device (DK was not purchased)
ERROR_USB_FAILED – Unable to communicate with device
ERROR_UNKNOWN_DEVICE – Unknown device is connected

LoadDriver

Loads the device driver library.

Declaration:

typedef _export _stdcall int (*fLoadDriver) (void);
TLoadDriver = function : Integer; stdcall;
Public Declare Function LoadDriver Lib "VBDevKit52X.dll" () As Long

Parameters:
None

Return Value:
ERROR_OK – Device driver loading succeeded
ERROR_DRIVER_NOT_LOADED – Device driver loading failed
ERROR_DRIVER_FUNCTIONS_MISSING – M52Xdrv.dll driver is not compatible with the DK

3.2. Trigger functions

SetTrigger

Sets desired threshold voltage.

Declaration:
typedef _export _stdcall int (*fSetTrigger) (int ThresholdVoltage, int ChannelSelector);
TSetTrigger = function (ThresholdVoltage: Integer; ChannelSelector: Integer): Integer; stdcall;
Public Declare Function SetTrigger Lib "VBDevKit52X.dll" (ByVal ThresholdVoltage As Long, ByVal ChannelSelector As Long) As Long

Parameters:
ThresholdVoltage – Desired threshold voltage for channel specified in ChannelSelector. Valid values are in range <0, 255>.
ChannelSelector – Valid values are constants CHANNEL_A or CHANNEL_B

Return Value:
ERROR_OK – Function call successfuly completed
ERROR_DEVELOPMENT_KIT_NOT_ENABLED – DK can not be used with connected device (DK was not purchased)
ERROR_USB_COMMUNICATION_FAILED – USB communication failed

Note:
Use the actual range setting and waveform vertical shift setting on specified channel to calculate ThresholdVoltage.

SetTriggerCount

Sets desired count of valid trigger event occurence is set. For more information see the DATAMAN 520 oscilloscope user's guide, chapters about triggering system.

Declaration:

typedef _export _stdcall int (*fSetTriggerCount) (int InputValue, int LevelSelector);

TSetTriggerCount = function (InputValue: Integer; LevelSelector: Integer): Integer; stdcall;

Public Declare Function SetTriggerCount Lib "VBDevKit52X.dll" (ByVal InputValue As Long, ByVal LevelSelector As Long) As Long

Parameters:

InputValue – Desired count of occurence. Valid range is <0, 32767>

LevelSelector – Specifies affected trigger system level. Valid values are TRIGGER_LEVEL_PRIMARY or TRIGGER_LEVEL_SECONDARY constants.

Return Value:

ERROR_OK – Function call successfuly completed

ERROR_DEVELOPMENT_KIT_NOT_ENABLED – DK can not be used with connected device (DK was not purchased)

ERROR_USB_COMMUNICATION_FAILED – USB communication failed

SetTriggerLength

Sets desired minimal length of valid trigger event. For more information see the DATAMAN 520 oscilloscope user’s guide, chapters about triggering system filters.

Declaration:

typedef _export _stdcall int (*fSetTriggerLength)(int SamplesCount, int LevelSelector);

TSetTriggerLength = function (SamplesCount: Integer; LevelSelector: Integer): Integer; stdcall;

Public Declare Function SetTriggerLength Lib "VBDevKit52X.dll" (ByVal SamplesCount As Long, ByVal LevelSelector As Long) As Long

Parameters:

InputValue – Desired length of valid trigger event (samples count). Valid values are 0, 8 and multiples of 4 from range <12, 131068>.

LevelSelector – Specifies which trigger system level is affected. Valid values are TRIGGER_LEVEL_PRIMARY or TRIGGER_LEVEL_SECONDARY constants.

Return Value:

ERROR_OK – Function call successfuly completed

ERROR_DEVELOPMENT_KIT_NOT_ENABLED – DK can not be used with connected device (DK was not purchased)

ERROR_USB_COMMUNICATION_FAILED – USB communication failed

SetTriggerMode

Sets trigger mode.

Declaration:

typedef _export _stdcall int (*fSetTriggerMode) (int TriggerMode);
TSetTriggerMode = function (TriggerMode: Integer): Integer; stdcall;
Public Declare Function SetTriggerMode Lib "VBDevKit52X.dll" (ByVal TriggerMode As Long) As Long

Parameters:

TriggerMode – Valid values are following constants:

TRIGGER_MODE_NORMAL – Data acquisition starts on occurrence of valid trigger event after the Data function call and
TRIGGER_MODE_AUTO – If valid trigger event does not occur, data acquisition starts immediately, otherwise on the trigger event
TRIGGER_MODE_MANUAL – Data acquisition starts immediately after the Data function call

Return Value:

ERROR_OK – Function call successfuly completed
ERROR_DEVELOPMENT_KIT_NOT_ENABLED – DK can not be used with connected device (DK was not purchased)
ERROR_USB_COMMUNICATION_FAILED – USB communication failed

Note:

Other trigger system settings are ignored, if the TRIGGER_MODE_MANUAL mode is set.

SetAfterTriggerSamplesCount

Sets desired samples count after trigger event.

Declaration:

typedef _export _stdcall int (*fSetAfterTriggerSamplesCount)(int InputValue);
TSetAfterTriggerSamplesCount = function (InputValue: Integer): Integer; stdcall;
Public Declare Function SetAfterTriggerSamplesCount Lib "VBDevKit52X.dll" (ByVal InputValue As Long) As Long

Parameters:

InputValue – Desired samples count. Valid range is <0, 63457>

Return Value:

ERROR_OK – Function call successfuly completed
ERROR_DEVELOPMENT_KIT_NOT_ENABLED – DK can not be used with connected device (DK was not purchased)
ERROR_USB_COMMUNICATION_FAILED – USB communication failed

Note:

Use actual sampling frequency to calculate after trigger acquisition length.

SetBeforeTriggerDelay

Sets desired hold off.

Declaration:

typedef _export _stdcall int (*fSetBeforeTriggerDelay) (int SamplesCount);
TSetBeforeTriggerDelay = function (SamplesCount: Integer): Integer; stdcall;
Public Declare Function SetBeforeTriggerDelay Lib "VBDevKit52X.dll" (ByVal SamplesCount As Long) As Long

Parameters:

SamplesCount – Desired hold off in samples. Valid range is <0, 131072>.

Return Value:

ERROR_OK – Function call successfuly completed
ERROR_DEVELOPMENT_KIT_NOT_ENABLED – DK can not be used with connected device (DK was not purchased)
ERROR_USB_COMMUNICATION_FAILED – USB communication failed

SetPrimaryTriggerSource

Primary level trigger source selection. You can activate more than one source at simultaneously.

Declaration:

typedef _export _stdcall int (*fSetPrimaryTriggerSource)(int InputMask);
TSetPrimaryTriggerSource = function (InputMask: Integer): Integer; stdcall;
Public Declare Function SetPrimaryTriggerSource Lib "VBDevKit52X.dll" (ByVal InputMask As Long) As Long

Parameters:

InputMask – Combination of constants TRIGGER_CHANNEL_A, TRIGGER_CHANNEL_B, TRIGGER_EXTERNAL (bitmask). If the given bit is set the appropriate trigger source will be considered valid.

Return Value:

ERROR_OK – Function call successfuly completed
ERROR_DEVELOPMENT_KIT_NOT_ENABLED – DK can not be used with connected device (DK was not purchased)
ERROR_USB_COMMUNICATION_FAILED – USB communication failed

Note:

Bitmask is sum of appropriate constants.

SetSecondaryTriggerSource

Secondary level trigger source selection. You can activate more than one source at simultaneously.

Declaration:

typedef _export _stdcall int (*fSetSecondaryTriggerSource)(int InputMask);
TSetSecondaryTriggerSource = function (InputMask: Integer): Integer; stdcall;
Public Declare Function SetSecondaryTriggerSource Lib "VBDevKit52X.dll" (ByVal InputMask As Long) As Long

Parameters:
InputMask – Combination of constants TRIGGER_CHANNEL_A, TRIGGER_CHANNEL_B, TRIGGER_EXTERNAL (bitmask). If the given bit is set the appropriate trigger source will be considered valid.

Return Value:
ERROR_OK – Function call successfuly completed
ERROR_DEVELOPMENT_KIT_NOT_ENABLED – DK can not be used with connected device (DK was not purchased)
ERROR_USB_COMMUNICATION_FAILED – USB communication failed

Note:
Bitmask is sum of appropriate constants.

InvertPrimaryTriggerSource

Selects whether the rising or falling edge will be considered as a valid trigger event on primary trigger level.

Declaration:

typedef _export _stdcall int (*fInvertPrimaryTriggerSource)(int InputMask);
TInvertPrimaryTriggerSource = function (InputMask: Integer): Integer; stdcall;
Public Declare Function InvertPrimaryTriggerSource Lib "VBDevKit52X.dll" (ByVal InputMask As Long) As Long

Parameters:
InputMask – Combination of constants TRIGGER_CHANNEL_A, TRIGGER_CHANNEL_B, TRIGGER_EXTERNAL (bitmask). If the given bit is set the appropriate trigger source will be considered valid.

Return Value:
ERROR_OK – Function call successfuly completed
ERROR_DEVELOPMENT_KIT_NOT_ENABLED – DK can not be used with connected device (DK was not purchased)
ERROR_USB_COMMUNICATION_FAILED – USB communication failed

Note:
Bitmask is sum of appropriate constants.

InvertSecondaryTriggerSource

Selects whether the rising or falling edge will be considered as a valid trigger event on secondary trigger level.

Declaration:

typedef _export _stdcall int (*fInvertSecondaryTriggerSource)(int InputMask);
TInvertSecondaryTriggerSource = function (InputMask: Integer): Integer; stdcall;
Public Declare Function InvertSecondaryTriggerSource Lib "VBDevKit52X.dll" (ByVal InputMask As Long) As Long

Parameters:
InputMask – Combination of constants TRIGGER_CHANNEL_A, TRIGGER_CHANNEL_B, TRIGGER_EXTERNAL (bitmask). If the given bit is set the appropriate trigger source will be considered valid.

Return Value:
ERROR_OK – Function call successfuly completed
ERROR_DEVELOPMENT_KIT_NOT_ENABLED – DK can not be used with connected device (DK was not purchased)
ERROR_USB_COMMUNICATION_FAILED – USB communication failed

Note:
Bitmask is sum of appropriate constants.

3.3. Time base functions

SetTimeBase

Sets desired time base.

Declaration:
typedef _export _stdcall int (*fSetTimeBase) (int InputValue, int &Retval);
TSetTimeBase = function (InputValue: Integer; var Retval: Integer): Integer; stdcall;
Public Declare Function SetTimeBase Lib "VBDevKit52X.dll" (ByVal InputValue As Long, ByRef RetVal As Long) As Long

Parameters:
InputValue – Desired TimeBase value in nanoseconds
Retval – Indicates, whether the measurement will be performed in sampling or in real mode. Valid values are DEVICE_TIME_MODE_SAMPLING or DEVICE_TIME_MODE_NORMAL

Return Value:
ERROR_OK – Function call successfuly completed
ERROR_DEVELOPMENT_KIT_NOT_ENABLED – DK can not be used with connected device (DK was not purchased)
ERROR_USB_COMMUNICATION_FAILED – USB communication failed

Note:

You should use only timebase values obtained from the InitHardware function via DeviceContex structure or constants from the table. It is recommended to set valid trigger length to zero (call function SetTriggerLength (0, TRIGGER_LEVEL_PRIMARY)) before entering sampling mode.

3.4. Vertical control functions

SetProbe

Sets desired probe attenuation 1:1, 1:10 or 1:100 on channel selected by parameter ChannelSelector.

Declaration:

typedef _export _stdcall int (*fSetProbe)(int ProbeType, int ChannelSelector);

TSetProbe = function(ProbeType: Integer; ChannelSelector: Integer): Integer; stdcall;

Public Declare Function SetProbe Lib "VBDevKit52X.dll" (ByVal ProbeType As Long, ByVal ChannelSelector As Long) As Long

Parameters:

ProbeType – Valid values are constants PROBE_TYPE_1, PROBE_TYPE_10, PROBE_TYPE_100

ChannelSelector – Valid values are constants CHANNEL_A or CHANNEL_B

Return Value:

ERROR_OK – Function call successfuly completed

ERROR_DEVELOPMENT_KIT_NOT_ENABLED – DK can not be used with connected device (DK was not purchased)

ERROR_USB_COMMUNICATION_FAILED – USB communication failed

SetRange

Sets the desired range on channel specified by parameter ChannelSelector.

Declaration:

typedef _export _stdcall int (*fSetRange)(int DensityValue, int ChannelSelector, int &Retval);

TSetRange = function (DensityValue: Integer; ChannelSelector: Integer; var Retval: Integer): Integer; stdcall;

Public Declare Function SetRange Lib "VBDevKit52X.dll" (ByVal DensityValue As Long, ByVal ChannelSelector As Long, ByRef RetVal As Long) As Long

Parameters:

DensityValue – The range value in millivolts to be set on channel selected by the ChannelSelector.

ChannelSelector – Valid values are constants CHANNEL_A or CHANNEL_B.

Retval – The new vertical shift value for given channel and given range is returned through Retval variable.

Return Value:

ERROR_OK – Function call successfuly completed

ERROR_DEVELOPMENT_KIT_NOT_ENABLED – DK can not be used with connected device (DK was not purchased)

ERROR_USB_COMMUNICATION_FAILED – USB communication failed

Note:

When changing the range, the waveform vertical position will also slightly change. Use the Retval value in SetVert function if you want to keep the waveform vertical position unchanged after range change.

SetDC

This function switches between AC or DC coupling on channel specified by ChannelSelector.

Declaration:

typedef _export _stdcall int (*fSetDC) (bool InputValue, int ChannelSelector);

TSetDC = function (InputValue: Boolean; ChannelSelector: Integer): Integer; stdcall;

Public Declare Function SetDC Lib "VBDevKit52X.dll" (ByVal InputValue As Boolean, ByVal ChannelSelector As Long) As Long

Parameters:

InputValue – *True* for DC coupling

ChannelSelector – Valid values are constants CHANNEL_A or CHANNEL_B

Return Value:

ERROR_OK – Function call successfuly completed

ERROR_DEVELOPMENT_KIT_NOT_ENABLED – DK can not be used with connected device (DK was not purchased)

ERROR_USB_COMMUNICATION_FAILED – USB communication failed

SetVert

Sets vertical shift.

Declaration:

typedef _export _stdcall int (*fSetVert) (int VerticalShiftValue, int ChannelSelector, int &Retval);

TSetVert = function (VerticalShiftValue: Integer; ChannelSelector: Integer; var Retval: Integer): Integer; stdcall;

Public Declare Function SetVert Lib "VBDevKit52X.dll" (ByVal VerticalShiftValue As Long, ByVal ChannelSelector As Long, ByRef RetVal As Long) As Long

Parameters:

VerticalShiftValue – Desired value of vertical shift on channel specified by ChannelSelector. Valid range is <0, 4095>

ChannelSelector – Valid values are constants CHANNEL_A or CHANNEL_B

Retval – The position of zero (GND) is returned via this variable.

Return Value:

ERROR_OK – Function call successfuly completed

ERROR_DEVELOPMENT_KIT_NOT_ENABLED – DK can not be used with connected device (DK was not purchased)

ERROR_USB_COMMUNICATION_FAILED – USB communication failed

3.5. Data acquisition function

Data

The data acquisition process is controlled by this function.

Declaration:

_export _stdcall int (*fData) (void *DataBuffer, int BufferLength, int &SamplesCount, bool &DataStatus, int &TriggerStatus);

TData = function (DataBuffer: Pointer; BufferLength: Integer; var SamplesCount; var DataStatus: Boolean; var TriggerStatus: Integer): Integer; stdcall;

Public Declare Function Data Lib "VBDevKit52X.dll" (Data As Integer, ByVal BufferLength As Long, ByRef SamplesCount As Long, ByRef DataStatus As Boolean, ByRef TriggerStatus As Long) As Long

Parameters:

TriggerStatus - This variable contains information on current data acquisition status and the sampling mode status. Decode it as follows:

Byte	Meaning
0	Data acquisition / trigger status
1	Sampling mode status for channel A
2	Sampling mode status for channel B

Sampling mode status value can be one of the following:
TRIGGER_SAMPLING_STATUS_LESS_THAN_HALF - less than 50% of data were measured
TRIGGER_SAMPLING_STATUS_MORE_THAN_HALF - more than 50% but less then 100% of data were measured
TRIGGER_SAMPLING_STATUS_MORE_THAN_ENOUGH - all data were measured

Data acquisition / trigger status value can be one of the following:
TRIGGER_STATUS_READY - measurement is completed

TRIGGER_STATUS_NOT_READY - measurement is in progress
TRIGGER_STATUS_NOT_TRIGGER - measurement started, waiting for trigger

SamplesCount – Amount of acquired samples
DataStatus – *True* indicates successful data acquisition
BufferLength – DataBuffer length (samples count), must be greater or equal to device memory size
DataBuffer – When measurement is finished, acquired data are placed in this buffer. The size of this the buffer must be *BufferLength* 16 bit words. Data are always located at the end of the buffer (location of trigger doesn’t move if same number of after trigger samples is set). Figure 3.5.1. shows data alignment.

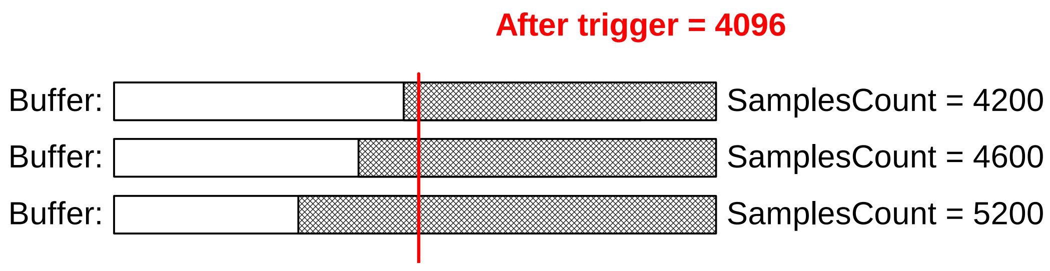

Figure 3.5.1. – Data alignment

Valid data are located from index (MemorySize – SamplesCount) to the end of the buffer.

Return Value:
ERROR_OK – Function call successfuly completed
ERROR_DEVELOPMENT_KIT_NOT_ENABLED – DK can not be used with connected device (DK was not purchased)
ERROR_USB_COMMUNICATION_FAILED – USB communication failed

EnableWaveformConformityDetection

Activates / deactivates WCD technology.

Declaration:
typedef _export _stdcall int (*fEnableWaveformConformityDetection) (bool EnableDetection, int DetectionSensitivity);
TEnableWaveformConformityDetection = function (EnableDetection: Boolean; DetectionSensitivity: Integer): Integer; stdcall;
Public Declare Function EnableWaveformConformityDetection Lib "VBDevKit52X.dll" (ByVal EnableDetection As Boolean, ByVal DetectionSensitivity As Long) As Long

Parameters:
EnableDetection – *True* enables the WCD, *false* disables it.
DetectionSensitivity – WCD sensitivity. Valid values are following constants:

WCD_VERY_HIGH_SENSITIVITY = 8
WCD_HIGH_SENSITIVITY = 12
WCD_MEDIUM_SENSITIVITY = 16
WCD_LOW_SENSITIVITY = 20

Return Value:

ERROR_OK – Function call successfuly completed
ERROR_DEVELOPMENT_KIT_NOT_ENABLED – DK can not be used with connected device (DK was not purchased)
ERROR_USB_COMMUNICATION_FAILED – USB communication failed

SetDigitalShielding

Enables digital shielding and sets its level on channel specified by ChannelSelector.

Declaration:

typedef _export _stdcall int (*fSetDigitalShielding) (bool DigitalShieldingActive, int DigitalShieldingLevel, int ChannelSelector);
TSetDigitalShielding = function (DigitalShieldingActive: Boolean; DigitalShieldingLevel: Integer;ChannelSelector: Integer): Integer; stdcall;
Public Declare Function SetDigitalShielding Lib "VBDevKit52X.dll" (ByVal DigitalShieldingActive As Boolean, ByVal DigitalShieldingLevel As Long, ByVal ChannelSelector As Long) As Long

Parameters:

DigitalShieldingActive – *True* activates digital shielding, *false* deactivates it
DigitalShieldingLevel – Desired level of digital shielding <2..64>. We recommend to set value 4 for most of measurements.
ChannelSelector – Valid values are constants CHANNEL_A or CHANNEL_B

Return Value:

ERROR_OK – Function call successfuly completed
ERROR_DEVELOPMENT_KIT_NOT_ENABLED – DK can not be used with connected device (DK was not purchased)
ERROR_USB_COMMUNICATION_FAILED – USB communication failed

Note:

The higher digital shielding level is entered, the slower oscilloscope reflects signal change.

SetShapePrediction

Activates the shape prediction for sampling mode.

Declaration:

Typedef _export _stdcall int (*fSetShapePrediction) (bool ShapePrediction);
TSetShapePrediction = function (ShapePrediction: Boolean): Integer; stdcall;
Public Declare Function SetShapePrediction Lib "VBDevKit52X.dll" (ByVal ShapePrediction As Boolean) As Long

Parameters:

ShapePrediction –

true – shape prediction active
false – shape prediction inactive

Return Value:

ERROR_OK – Function call successfuly completed
ERROR_DEVELOPMENT_KIT_NOT_ENABLED – DK can not be used with connected device (DK was not purchased)
ERROR_USB_COMMUNICATION_FAILED – USB communication failed

Note:

Shape prediction is used in the sampling mode to predict the waveform shape from measured samples until all data are collected.

DeviceContext

```
struct DensityItem {
  /* Millivolts per divider */
   int fDensity;
   /* Value to be filled in the device density registry */
   unsigned short int fDensityRegistryCombinationChannelA;
   /* Value to be filled in the device reference registry for Probe type 1 */
   unsigned short int fReferenceRegistryCombinationProbe1ChannelA;
   /* Value to be filled in the device reference registry for Probe type 10 */
   unsigned short int fReferenceRegistryCombinationProbe10ChannelA;
   /* Value to be filled in the device reference registry for uncalibrated device */
   unsigned short int fReferenceRegistryCombinationNoCalibChannelA;
   /* Value to be filled in the device density registry */
   unsigned short int fDensityRegistryCombinationChannelB;
   /* Value to be filled in the device reference registry for Probe type 1 */
   unsigned short int fReferenceRegistryCombinationProbe1ChannelB;
   /* Value to be filled in the device reference registry for Probe type 10 */
   unsigned short int fReferenceRegistryCombinationProbe10ChannelB;
   /* Value to be filled in the device reference registry for uncalibrated device */
   unsigned short int fReferenceRegistryCombinationNoCalibChannelB;
  };

  struct DeviceDensitySettings {
   /* Count of used values in arrays */
   int fItemsCount;
   DensityItem fDensityItems[11];
  };

  struct TimeBaseItem {
   /* Period per divider in nanoseconds */
   double fTimeBase;
   /* Sampling period in nanoseconds */
   double fSamplingPeriod;
```

```
    /* Flag for each item indicating if it is in sampling mode */
    bool fSamplingMode;
    /* Value to be filled in the device registry */
    unsigned short int fRegistryCombination;
  };

struct DeviceSweepSettings {
    /* Count of used values in arrays */
    int fItemsCount;
    /* Array of sweep values */
    int fSweepValue [16];
  };

  struct DeviceTimeBaseSettings {
    /* Count of used values in arrays */
    int fItemsCount;
    /* Sampling Mode level */
    int fSamplingModeLevel;
    /* Array of TTimeBaseItems */
    TimeBaseItem fTimeBaseItems[31] ;
  };

  /* Device contex structure is filled at device initialization/detection time with proper
values  for given device type */

  struct DeviceContext {
    /* Link to sweep settings instance */
    struct DeviceSweepSettings fDeviceSweepSettings;
    /* Link to timebase settings instance */
    struct DeviceTimeBaseSettings fDeviceTimeBaseSettings;
    /* Link to density settings instance */
    struct DeviceDensitySettings fDeviceDensitySettings;
    /* Count of samples displayed per screen for Sweep 1:1 */
    int fMeasurementView;
    /* Count of samples displayed per divider for Sweep 1:1 */
    int fPointsPerDivider;
    /* Device memory size */
    int fDeviceMemorySize;
    /* Device ID */
    int fDeviceID;
  };

struct ScopeDriverVersion {
          /* Driver Major version */
          unsigned char MajorVersion;
          /* Driver Minor version */
          unsigned char MinorVersion;
  };
```

3.6. Other functions

CheckConnection

Checks the communication with device.

Declaration:

typedef _export _stdcall int (*fCheckConnection)(void);
TCheckConnection = function : Integer; stdcall;
Public Declare Function CheckConnection Lib "VBDevKit52X.dll" () As Long

Parameter:

None

Return Value:

ERROR_OK – Function call successfuly completed
ERROR_USB_COMMUNICATION_FAILED – USB communication failed

CompensationGenerator

Activates / deactivates the compensation generator.

Declaration:

typedef _export _stdcall int (*fCompensationGenerator)(bool InputValue);
TCompensationGenerator = function (InputValue: Boolean): Integer; stdcall;
Public Declare Function CompensationGenerator Lib "VBDevKit52X.dll" (ByVal InputValue As Boolean) As Long

Parameters:

InputValue - Boolean value
true – compensation generator active
false – compensation generator inactive

Return Value:

ERROR_OK – Function call successfuly completed
ERROR_DEVELOPMENT_KIT_NOT_ENABLED – DK can not be used with connected device (DK was not purchased)
ERROR_USB_COMMUNICATION_FAILED – USB communication failed

GetSDKVersion

Returns the Development kit version.

Declaration:

typedef _export _stdcall int (*fGetSDKVersion)(void *Version);
TGetSDKVersion = function (Version: Pointer):integer;stdcall;

Public Declare Function GetSDKVersion lib “VBDevKit52X.dll” (DriverVersion as TDriverVersion) As Long;

Parameters:
Version – Pointer to the DeviceDriverVersion structure

Return Value:
ERROR_OK – Function executed successfully

GetDeviceDriverVersion

Returns the device driver version.

Declaration:
typedef _export _stdcall int (*fGetDeviceDriverVersion)(void *Version);
TGetDeviceDriverVersion = function (Version: Pointer): Integer; stdcall;
Public Declare Function GetDeviceDriverVersion Lib "VBDevKit52X.dll" (DriverVersion As TDriverVersion) As Long

Parameters:
Version - Pointer to the device driver version structure

Return Value:
ERROR_OK – Function call successfuly completed
ERROR_DEVELOPMENT_KIT_NOT_ENABLED – DK can not be used with connected device (DK was not purchased)

GetUSBDriverVersion

Return the USB driver (em52X) version.

Declaration:
typedef _export _stdcall int (*fGetUSBDriverVersion)(void *Version);
TGetUSBDriverVersion = function (Version: Pointer): Integer; stdcall;
Public Declare Function GetUSBDriverVersion Lib "VBDevKit52X.dll" (DriverVersion As TDriverVersion) As Long

Parameters:
Version - Pointer to the device driver version structure

Return Value:
ERROR_OK – Function call successfuly completed
ERROR_DEVELOPMENT_KIT_NOT_ENABLED – DK can not be used with connected device (DK was not purchased)

3.7. Visual Basic functions

Since DeviceContext structure mapping and members accessing would be problematic in Visual Basic following functions provide interface for those data access. Structure is accessible after successful InitHardware call.

GetDensities

Returns supported voltage ranges.

Declaration:

Public Declare Function GetDensities Lib "VBDevKit52X.dll" (DensitiesField As Long) As Long

Parameters:

DensitiesField – Voltage ranges supported by device will be filled into this field.

Return Value:

None

Example:

Dim DensityField(12) As Long
RetVal = GetDensities(DensityField(0))

GetDeviceID

Returns ID of the connected device. This value identifies device model. Use the DeviceIDConstsArray constants or following table to identify the device.

ID	Device model
1	DATAMAN 522
3	DATAMAN 524
5	DATAMAN 526

Declaration:

Public Declare Function GetDeviceID Lib "VBDevKit52X.dll" () As Long

Parameters:

None

Return Value:

DeviceID value, which can be converted to device type name.

GetDeviceMemorySize

Returns connected device memory size.

Declaration:
Public Declare Function GetDeviceMemorySize Lib "VBDevKit52X.dll" () As Long

Parameters:
None

Return Value:
Memory size in samples count is returned.

GetDevicePointsPerDivider

Supported devices have different time bases. Due to various timebases, there is different points per divider ratio. Use returned value to display data properly.

Declaration:
Public Declare Function GetDevicePointsPerDivider Lib "VBDevKit52X.dll" () As Long

Parameters:
None

Return Value:
Points per divider ratio

GetTimeBases

Returns supported time base values.

Declaration:
Public Declare Function GetTimeBases Lib "VBDevKit52X.dll" (TimeBaseField As Long) As Long

Parameters:
TimeBaseField – Field where valid time base values will be filled after device initialization.

Return Value:
None

Example:
```
Dim TimeBaseField(32) As Long
RetVal = GetTimeBases(TimeBaseField(0))
```

3.8. Constants used in the DK

Channel descriptors

CHANNEL_A	100

CHANNEL_B	200
CHANNEL_C	300
CHANNEL_D	400

Data acquistion control constants

CODE_RESET_MEASURING	40
CODE_START_MEASURING	41
CODE_STOP_MEASURING	42

Device driver return values

DEVICE_FUNCTION_SUCCEEDED	1
DEVICE_TIME_MODE_NORMAL	2
DEVICE_TIME_MODE_SAMPLING	3

Trigger system status indicators

TRIGGER_STATUS_READY	1
TRIGGER_STATUS_NOT_READY	2
TRIGGER_STATUS_NOT_TRIGGER	3

Sampling mode data acquistion status indicators

TRIGGER_SAMPLING_STATUS_LESS_THAN_HALF	1
TRIGGER_SAMPLING_STATUS_MORE_THAN_HALF	2
TRIGGER_SAMPLING_STATUS_MORE_THAN_ENOUGH	3

Trigger system level selectors

TRIGGER_LEVEL_PRIMARY	1
TRIGGER_LEVEL_SECONDARY	2

Trigger source selectors

TRIGGER_CHANNEL_A	1
TRIGGER_CHANNEL_B	2
TRIGGER_EXTERNAL	4

Trigger mode selectors

TRIGGER_MODE_AUTO	1
TRIGGER_MODE_NORMAL	2
TRIGGER_MODE_MANUAL	4

Probe attenuation selectors

PROBE_TYPE_1	1
PROBE_TYPE_10	2
PROBE_TYPE_100	3

DK interface error codes

ERROR_OK	1000
ERROR_USB_FAILED	1001
ERROR_USB_DRIVER_NOT_LOADED	1002
ERROR_DEVICE_CONFIGURATION_FAILED	1003
ERROR_DEVICE_CALIBRATION_BROKEN	1004
ERROR_DEVICE_DRIVER_NO_ENTRY_POINT	1005

ERROR_DEVICE_DRIVER_NOT_LOADED	1006
ERROR_USB_COMMUNICATION_FAILED	1007
ERROR_DEVELOPMENT_KIT_NOT_ENABLED	1008
ERROR_DRIVER_NOT_LOADED	1009
ERROR_DRIVER_FUNCTIONS_MISSING	1010
ERROR_UNKNOWN_DEVICE	1011

WCD sensitivity constants

WCD_VERY_HIGH_SENSITIVITY	8
WCD_HIGH_SENSITIVITY	12
WCD_MEDIUM_SENSITIVITY	16
WCD_LOW_SENSITIVITY	20

Timebases Table

DATAMAN 522		DATAMAN 524		DATAMAN 526	
Timebase	S.p.	Timebase	S.p..	TimeBase	S.p.
				2 ns	12.5ns
		5 ns	20 ns	5 ns	12.5 ns
10 ns	20 ns	10 ns	20 ns	10 ns	12.5 ns
20 ns	20 ns	20 ns	20 ns	20 ns	12.5 ns
50 ns	20 ns	50 ns	20 ns	50 ns	12.5 ns
100 ns	20 ns	100 ns	20 ns	100 ns	12.5 ns
200 ns	20 ns	200 ns	20 ns	200 ns	5 ns
500 ns	20 ns	500 ns	10 ns	500 ns	12.5 ns
1 us	20 ns	1 us	20 ns	1 us	25 ns
2 us	40 ns	2 us	40 ns	2 us	50 ns
5 us	100 ns	5 us	100 ns	5 us	125 ns
10 us	200 ns	10 us	200 ns	10 us	250 ns
20 us	400 ns	20 us	400 ns	20 us	500 ns
50 us	1 us	50 us	1 us	50 us	1.250 us
100 us	2 us	100 us	2 us	100 us	2.5 us
200 us	5 us	200 us	5 us	200 us	5 us
500 us	10 us	500 us	10 us	500 us	12.5 us
1 ms	20 us	1 ms	20 us	1 ms	25 us
2 ms	40 us	2 ms	40 us	2 ms	50 us
5 ms	100 us	5 ms	100 us	5 ms	125 us
10 ms	200 us	10 ms	200 us	10 ms	250 us
20 ms	400 us	20 ms	400 us	20 ms	500 us

50 ms	1 ms	50 ms	1 ms	50 ms	1.25 ms
100 ms	2 ms	100 ms	2 ms	100 ms	2.5 ms