

**Development kit for DATAMAN 530  
Programmer's Guide  
Visual Basic .NET  
Version 1.01**



Copyright © 2004-2008 Dataman Programmers Ltd

For further information please contact us via phone or preferably e-mail at 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](mailto:support@dataman.com) - technical support  
[sales@dataman.com](mailto:sales@dataman.com) - sales  
[info@dataman.com](mailto:info@dataman.com) - other information

## Contents

1. Basic information.....	5
1.1. Development kit contents .....	5
1.2. DK usage.....	5
1.3. Application deployment.....	5
2. Controlling the device.....	6
2.1. Device initialization .....	6
2.2. Error handling .....	6
2.3. Waveform generation.....	6
2.4. Utilization of the PLL pair.....	7
3. Reference .....	8
3.1. Functions which return information about device .....	9
3.1.1. GetDKError.....	9
3.1.2. ResetDKError .....	9
3.1.3. IsPowered.....	9
3.2. Initialization functions .....	11
3.2.1. LoadDriver .....	11
3.2.2. InitHardware .....	11
3.3. Functions which set the generation parameters .....	12
3.3.1. SetWaveform .....	12
3.3.2. SetLevel .....	12
3.3.3. SetOutputAttenuation .....	12
3.3.4. SetOutputOnOff .....	12
3.3.5. SetFilter.....	13
3.3.6. SetFrequency.....	13
3.3.7. SetShift.....	13
3.3.8. SetTriggerMode .....	13
3.3.9. SetARMOnOff .....	14
3.3.10. SetArmStopLevel.....	14
3.3.11. SetGenerationOnOff .....	14
3.3.12. StartGeneratoion .....	15
3.3.13. SetTriggerOnOff .....	15
3.3.14. SetTriggerEdge .....	15
3.3.15. PrepareFrequency .....	15
3.4. Other functions.....	17
3.4.1. SetPowerOnOff.....	17

## Figures and tables

Table 1.1. –Development kit (DK) contents .....5

## 1. Basic information

### 1.1. Development kit contents

All development kit (DK) parts are located in the installation directory.

Directory	Contents
Examples\C#.NET	C#.NET example
Examples\VB.NET	Visual Basic .NET example
Examples\VB	Visual Basic 6.0 example
Examples\Delphi	Delphi example
Examples\CBuilder	C++ Builder example
Examples\VC	Visual C++ example
Include\C#.NET	C#.NET header files
Include\VB.NET	Visual Basic .NET header files
Include\VB	Visual Basic 6.0 header files
Include\Delphi	Delphi header files
Include\CBuilder	C++ Builder header files
Include\VC	Visual C++ 6.0 header files
Bin	m530drvdk.dll and m530drv.dll libraries

*Table 1.1. –Development kit (DK) contents*

### 1.2. DK usage

In order for the DK to work properly, it is necessary to have the DATAMAN 530 generator drivers installed. The m530drv.vb header file contains the cm530drv class, which encapsulates all the DK functions and constants. Add this file to project to gain access to the cm530drv class. The m530drvdk.dll and m530drv.dll must be present in the same directory as the .exe file is (bin\debug).

### 1.3. Application deployment

The m530drvdk.dll and m530drv.dll libraries must be distributed together with your application. The drivers for the DATAMAN 530 generator must be installed in the system in order to communicate with the device. The application will work with every device with the DK activated.

## 2. Controlling the device

### 2.1. Device initialization

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

```
cm530drv.LoadDriver()
```

After the driver is loaded, it is possible to initialize the device using the function `InitHardware`. This function returns also the information as to whether the calibration data is correct or not.

```
Dim CalibOK As Integer  
cm530drv.InitHardware(CalibOK)
```

### 2.2. Error handling

In case the error occurs, all subsequent calls to functions will fail. Therefore it is necessary to check if the operations were successful (for example check if the initialization was successful). Use `GetDKError` to obtain the error code.

```
Dim res As integer  
res = cm530drv.GetDKError()
```

In case of an error, it is necessary to reset the error flag (to indicate to the DK, that the error has been handled). Use `ResetDKError` function to do so (otherwise no other function will be successful).

```
cm530drv.ResetDKError()
```

### 2.3. Waveform generation

The DATAMAN 531 generator generates the waveforms composed of samples with 12 bit resolution. Therefore each sample has 4096 quantization levels (from 0 to 4095 – where 2048 corresponds to the selected shift and 4095 corresponds to the selected shift + level). The generator distinguishes the two waveform types by their length:

- the standard length waveform – the length is always fixed to 8192 samples. The frequency which is set is the frequency of the whole waveform. If the frequency is higher than the capabilities of the generator, the DK automatically undersamples the waveform and converts its length to match the selected frequency.

- the arbitrary length waveform – the length of this type is arbitrary. The frequency which is set is the sampling rate of the generator. Therefore it is possible to calculate the frequency of the waveform as  $\text{Frequency/Length}$ . If the frequency is higher than the capabilities of the generator, the DK sets the highest possible frequency.

Use `SetWaveform` function to set the waveform to the device.

```
Dim data(8192) As Int16  
Dim i As Integer  
For i = 0 To 8191  
    data(i) = i / 2  
Next  
cm530drv.SetWaveform(data(0), 8192, cm530drv.ONOFF_OFF)  
cm530drv.SetFrequency(10000)
```

## **2.4. Utilization of the PLL pair**

The DATAMAN 531 generator contains two PLLs which determine the output frequency. Only one is active at a time. In case the software requires a step change of the frequency, the PLL tuning can take 500ms in worst case. However, it is possible to preset the second PLL to the new frequency. When the second PLL is preset to the new frequency a change of the frequency doesn't require PLL tuning. Use `PrepareFrequency` to preset the second PLL. It is not necessary to preset the second PLL if the step frequency change doesn't occur or the longer settle time is tolerated for the application.

### 3. Reference

Functions available in the DK can be divided into four groups:

#### **Functions, which return information about device**

GetDKError  
ResetDKError  
IsPowered

#### **Initialization functions**

LoadDriver  
InitHardware

#### **Functions, which set the generation parameters**

SetWaveform  
SetLevel  
SetOutputAttenuation  
SetOutputOnOff  
SetFilter  
SetFrequency  
SetShift  
SetTriggerMode  
SetArmOnOff  
SetArmStopLevel  
SetGenerationOnOff  
StartGeneration  
SetTriggerOnOff  
SetTriggerEdge  
PrepareFrequency

#### **Other functions**

SetPowerOnOff



### 3.1. Functions which return information about device

#### 3.1.1. GetDKError

If an error occurs during a call of any DK function the error code is stored in the DK internal variable. All subsequent calls of DK functions will fail. GetDKError returns the error code.

**Declaration:**

*Public Shared Function GetDKError() As Integer*

**Parameters: -**

**Return value:**

*ERROR\_OK* – no error occurred  
*ERROR\_DRIVER\_NOT\_LOADED* – unable to load driver/driver wasn't loaded before call  
*ERROR\_DRIVER\_INCOMPATIBLE* - driver is not compatible with DK  
*ERROR\_UNABLE\_TO\_LOAD\_EM57X* – unable to load em57x driver  
*ERROR\_INIT\_FAILED* – device initialization failed  
*ERROR\_FPGA\_CONFIG\_FAILED* – FPGA configuration failed  
*ERROR\_COMMUNICATION\_FAILED* – communication with device is broken  
*ERROR\_OPERATION\_FAILED* – unable to finish last operation  
*ERROR\_DK\_NOT\_ENABLED* – the device doesn't have DK enabled  
*ERROR\_INCORRECT\_PARAMETER* – the function was called with incorrect parameter value

#### 3.1.2. ResetDKError

If an error occurs during a call of any DK function the error code is stored in the DK internal variable. All subsequent calls of DK functions will fail. ResetDKError function resets this variable thus allowing you to call DK functions again.

**Deklarácia:**

*Public Shared Sub ResetDKError()*

**Parameters: -**

**Return value: -**

#### 3.1.3. IsPowered

Returns whether the device is powered from external power source.

*Public Shared Function IsPowered() As Integer*

**Parameters:** -

**Return value:**

ONOFF\_ON – the device is powered

ONOFF\_OFF – the device is not powered

## 3.2. Initialization functions

### 3.2.1. LoadDriver

Loads m530drv.dll driver.

*Public Shared Function LoadDriver() As Integer*

**Parameters: -**

**Return value:**

*ERROR\_OK* – driver loaded successfully

*ERROR\_DRIVER\_NOT\_LOADED* – unable to load m530drv.dll library

*ERROR\_DRIVER\_INCOMPATIBLE* – m530drv.dll isn't compatible with DK

**Remark:**

The returned value is stored in the internal DK variable as well. Use GetDKError function to access this internal variable.

### 3.2.2. InitHardware

Initializes device. After successful call of this function, the device can be used.

*Public Shared Function InitHardware(ByRef CalibOK As Integer) As Integer*

**Parameters:**

*CalibOK* – this variable will be filled with information whether the calibration data in the device is ok

1 – calibration data is ok

0 – calibration data in the device is not ok.

**Return value:**

*ERROR\_OK* – device was initialized successfully

*ERROR\_UNABLE\_TO\_LOAD\_EM53X* – unable to load em53x driver

*ERROR\_INIT\_FAILED* – device initialization failed (one of the reason can be, that the device isn't connected)

*ERROR\_FPGA\_CONFIG\_FAILED* – FPGA initialization failed

*ERROR\_DK\_NOT\_ENABLED* – the DK isn't enabled in the connected device

*ERROR\_DRIVER\_NOT\_LOADED* – the driver m530drv.dll wasn't loaded before this call (use LoadDriver function to load it)

### 3.3. Functions which set the generation parameters

#### 3.3.1. SetWaveform

Sets the shape of the waveform.

```
Public Shared Sub SetWaveform(ByRef Data As Int16, ByVal DataLength As Integer, ByVal arbitrary As Integer)
```

**Parameters:**

*data* – pointer to the data

*datalength* – length of the data in samples

*arbitrary* – turns arbitrary length mode on/off. In case this mode is off, the *datalength* must equal to 8192.

**Return value:** -

**Remark:** It is necessary to set frequency after setting the waveform (by using the SetFrequency function).

#### 3.3.2. SetLevel

Sets output level.

```
Public Shared Sub SetLevel(ByVal lvl As Double)
```

**Parametre:**

*lvl* – indicates the voltage between quantization levels 4095 and 2048 in volts

**Return value:** -

#### 3.3.3. SetOutputAttenuation

Sets output attenuation to 1:1 or 1:10.

```
Public Shared Sub SetOutputAttenuation(ByVal attn As Integer)
```

**Parameters:**

*attn* – indicates attenuator:

*OUTPUT\_ATTENUATION\_1* – 1:1

*OUTPUT\_ATTENUATION\_10* – 1:10

**Return value:** -

#### 3.3.4. SetOutputOnOff

Turns output on/off.

```
Public Shared Sub SetOutputOnOff(ByVal OnOff As Integer)
```

**Parameters:**

*OnOff* – indicates, whether the output is turned on or off  
*ONOFF\_ON* – turns output on  
*ONOFF\_OFF* – turns output off

**Return value:** -

**3.3.5. SetFilter**

Sets selected filter on the output.

*Public Shared Sub SetFilter(ByVal filter As Integer)*

**Parameters:**

*filter* – indicates filter to be connected to the output  
*FILTER\_NO* – no filter  
*FILTER\_40MHZ* – 40MHz filter on the output  
*FILTER\_20MHZ* – 20MHz filter on the output

**Return value:** -

**3.3.6. SetFrequency**

Sets the frequency on the output. If the arbitrary waveform length is selected this function sets the sampling rate. Otherwise it sets the waveform frequency.

*Public Shared Function SetFrequency(ByVal freq As Double) As Double*

**Parameters:**

*freq* – frequency (If the arbitrary waveform length is selected this parameter indicates the sampling rate. Otherwise it indicates the waveform frequency.)

**Return value:**

Frequency that was set to the device.

**3.3.7. SetShift**

Sets shift of the waveform.

*Public Shared Sub SetShift(ByVal shift As Double)*

**Parameters:**

*shift* – indicates the level of the 2048 quantization level in volts

**Return value:** -

**3.3.8. SetTriggerMode**

Sets trigger mode.

*Public Shared Sub SetTriggerMode(ByVal mode As Integer)*

**Parameters:**

*mode* – trigger mode

*TRIGGER\_MODE\_PERIOD* – generator generates the waveform without any waiting for trigger event

*TRIGGER\_MODE\_SINGLE* – generator generates always only one period after the trigger event occurs

**Return value:** -

### 3.3.9. SetARMonOff

Turns the arm on/off. If the arm is on the selected level on the “SI” input pauses the generation.

*Public Shared Sub SetArmOnOff(ByVal OnOff As Integer)*

**Parameters:**

*OnOff* – indicates, whether the arm will be turned on or off

*ONOFF\_ON* – on

*ONOFF\_OFF* – off

**Return value:** -

### 3.3.10. SetArmStopLevel

Sets the level which pauses the generation when the arm is turned on.

*Public Shared Sub SetArmStopLevel(ByVal level As Integer)*

**Parameters:**

*level* – indicates level which pauses the generation when the arm is turned on

0 – logic 0

1 – logic 1

**Return value:** -

### 3.3.11. SetGenerationOnOff

This function works in the period trigger mode only. It starts/stops the waveform generation.

*Public Shared Sub SetGenerationOnOff(ByVal OnOff As Integer)*

**Parameters:**

*OnOff* – indicates, whether the generation will be started/stopped

*ONOFF\_ON* – starts generation

*ONOFF\_OFF* – stops generation

**Return value:** -

### 3.3.12. StartGeneratoion

Starts the generation of the one period in the single trigger mode. This function has no effect in the period trigger mode.

*Public Shared Sub StartGeneration()*

**Parameters:** -

**Return value:** -

### 3.3.13. SetTriggerOnOff

Turns the triggering from external source on/off. This function works in the single trigger mode only.

*Public Shared Sub SetTriggerOnOff(ByVal OnOff As Integer)*

**Parameters:**

*OnOff* – indicates, whether the triggering from external source is turned on/off

*ONOFF\_ON* – on

*ONOFF\_OFF* – off

**Return value:** -

### 3.3.14. SetTriggerEdge

Sets the trigger sensitivity on the selected edge.

*Public Shared Sub SetTriggerEdge(ByVal edge As Integer)*

**Parameters:**

*edge* – indicates edge

*TRIGGER\_EDGE\_LEADING* – leading edge

*TRIGGER\_EDGE\_TRAILING* – trailing edge

**Return value:** -

### 3.3.15. PrepareFrequency

Sets the second phase locked loop to the selected frequency. By presetting the frequency you can achieve immediate tuning to this frequency after calling the SetFrequency function.

*Public Shared Function PrepareFrequency(ByVal freq As Double) As Double*

**Parameters:**

*freq* – frequency which is preset to the device

**Return value:**

Frequency which was preset to the device.



### 3.4. Other functions

#### 3.4.1. SetPowerOnOff

Turns the power from external source on/off.

*Public Shared Sub SetPowerOnOff(ByVal OnOff As Integer)*

**Parameters:**

*OnOff* – turns the power from external source on/off

*ONOFF\_ON* – turns the power on

*ONOFF\_OFF* – turns the power off

**Return value:** -