Development kit for DATAMAN 530
Programmer's Guide
Visual C++ 6.0
Version 1.01

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 - technical support
sales@dataman.com - sales
info@dataman.com - other information

Contents

Figures and tables

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.h header file contains all the DK functions and m530drv.lib contains association of these functions with the external DLL. Add both files to the project to gain access to the DK functions. The m530drvdk.dll and m530drv.dll must be present in the project directory.

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.

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.

int calibok;
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.

int res;

int res = 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).

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 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.

```
short data[8192];
for (int i = 0; i < 8192; i++)
        data[i] = i / 2
SetWaveform(&data[0], 8192, ONOFF_OFF);
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:

int __declspec(dllimport) __stdcall GetDKError();

Parameters: -

Return value:

ERROR_OK – no error occured

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:

void __declspec(dllimport) __stdcall ResetDKError();

Parameters: -

Return value: -

3.1.3. IsPowered

Returns whether the device is powered from external power source.

int __declspec(dllimport) __stdcall IsPowered();

Parameters: -

Return value:

ONOFF_ON – the device is powered
ONOFF_OFF – the device is not powered

3.2. Initialization funcitions

3.2.1. LoadDriver

Loads m530drv.dll driver.

int __declspec(dllimport) __stdcall LoadDriver();

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.

*int __declspec(dllimport) __stdcall InitHardware(int *CalibOK);*

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.

*void __declspec(dllimport) __stdcall SetWaveform(short *data, int datalength, int arbitrary);*

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.

void __declspec(dllimport) __stdcall SetLevel(double lvl);

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.

void __declspec(dllimport) __stdcall SetOutputAttenuation(int attn);

Parameters:
attn – indicates attenuator:
OUTPUT_ATTENUATION_1 – 1:1
OUTPUT_ATTENUATION_10 – 1:10

Return value: -

3.3.4. SetOutputOnOff

Turns output on/off.

void __declspec(dllimport) __stdcall SetOutputOnOff(int OnOff);

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.

void __declspec(dllimport) __stdcall SetFilter(int filter);

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.

double __declspec(dllimport) __stdcall SetFrequency(double freq);

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.

void __declspec(dllimport) __stdcall SetShift(double shift);

Parameters:

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

Return value: -

3.3.8. SetTriggerMode

Sets trigger mode.

void __declspec(dllimport) __stdcall SetTriggerMode(int mode);

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.

void __declspec(dllimport) __stdcall SetArmOnOff(int OnOff);

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.

void __declspec(dllimport) __stdcall SetArmStopLevel(int level);

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.

void __declspec(dllimport) __stdcall SetGenerationOnOff(int OnOff);

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.

void __declspec(dllimport) __stdcall 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.

void __declspec(dllimport) __stdcall SetTriggerOnOff(int OnOff);

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.

void __declspec(dllimport) __stdcall SetTriggerEdge(int edge);

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.

double __declspec(dllimport) __stdcall PrepareFrequency(double freq);

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.

void __declspec(dllimport) __stdcall SetPowerOnOff(int OnOff);

Parameters:

OnOff – turns the power from external source on/off
ONOFF_ON – turns the power on
ONOFF_OFF – turns the power off

Return value: -