

DSKController V2.00 Module

Borland C++ Builder V4 Module

Release dated 22nd March 2001 - Version 2.0.0.76

Written while studying at Dept. Physics, Heriot-Watt University, Edinburgh, UK.

Keith James Symington

kjsymington@iee.org

<http://www.symington.org/>

Files

DSKCtrl.cpp, DSKCtrl.h, DSKCtrl.dfm, Ports.cpp, Ports.h, dskc.ico, DSKCtrl.pdf.

Specification

This module loads DSK and COFF files to multiple TMS 320C5X digital starter kits via the selected port(s) and runs or resets them as specified. It is essentially a Windows version of the supplied HI5LIB.

Advantages

The module provides a high degree of user configurability along with logging for up to four channels simultaneously. What makes it especially attractive is its ability to calibrate communication delays for a specific computer. Using the Sleep() function when transmitting to a DSK is slow and inefficient. Calibration decreases the delay associated with information transfer by an order of magnitude, approaching optimal transmission speeds for the com port.

Requirements

Free com port, Windows 95/98 or ME. This module has not been tested under Windows NT/2000 but it will probably not run as direct port access is required. The Form program requires MS Serif and Courier New fonts to be installed on the host system. Obviously, Borland C++ Builder is required to link the module into a project.

Channels and Comm. Ports

This document refers to both "channels" and "com ports". There is a clear distinction between these definitions. A channel has an unspecified com port but always exists from channel 1 up. If your computer is using two channels it can communicate with two starter kits. Each channel is assigned a specific com port. The reason for the difference is that sometimes a com port is taken up by another device - most usually a mouse. This channel designation prevents the first channel being unavailable by assigning the first channel to the next available com port.

DSK Communication

On startup the module searches for all available com ports using Windows API functions. This prevents any clashes with drivers etc. The available com ports can be selected using the user interface tabs.

The PC transmits and receives through its on-board UART. The DSK does not have a UART and therefore must simulate one. The DSK UART is simulated using two hardware pins known as the External Flag output (XF) and the Branch control Input

(BIO). The XF line is connected to the PC comport's RX pin and the BIO pin is connected to comports's TX line. The PC comport's DTR line is connected to the DSP's reset pin and remains high unless resetting the DSP.

This results in the following connections:

PC TX (XMIT) to DSP BIO using RS232 pin 3 (DB9) or pin 2 (DB25).

PC RX (RCV) to DSP XF using RS232 pin 2 (DB9) or pin 3 (DB25).

PC DTR (/RS) to DSP /RS using RS232 pin 4 (DB9) or pin 20 (DB25).

PC signal GND to DSP signal GND using RS232 pin 5 (DB9) or pin 7 (DB25).

All DSP transmit and receive data is at TTL levels, whereas RS-232 levels are 12v. The on-board RS-232 buffers take care of the voltage conversions.

Limitations

This program was written and tested on a PIII 450MHz with 128MB RAM. This is not however the minimum system. There are some known issues with shared interrupts on com ports so be careful if creating multiple threads. With an add-on com port card, com1/com3 use IRQ 4 and com2/com4 use IRQ 3. Therefore resetting both com2 and com4 simultaneously may result in the system locking up. Data transfer is performed by direct port access so this may be done simultaneously.

Legal Stuff

This software and all accompanying files are provided "as is" and without any warranties expressed or implied including but not limited to implied warranties of merchantability and fitness for a particular purpose.

In no event shall any liability be accepted for any damages whatsoever (including without limitation, damages for loss of business profits, business interruption, loss of business information, or any other pecuniary loss) arising out of the use or inability to use this product.

User Interface

Main Tab

The main tab can be seen overleaf. This controls general parameters and contains information about the linked module version and limitations. Each of the controls in this window and their function is now explained.

Use Channels: Select the number of DSKs to be used simultaneously. The number of available channels corresponds directly to the number of available com ports as one com port is required per channel. Just because a com port is installed does not mean that it is available since drivers such as mice may be using them. Channels can be individually configured using the appropriate tab. The maximum number of channels available is selected by default. Altering the number of available channels hides or shows the appropriate tabs.

Heriot-Watt University Logo: Clicking on this will take you to the Physics Department web site using your default browser.

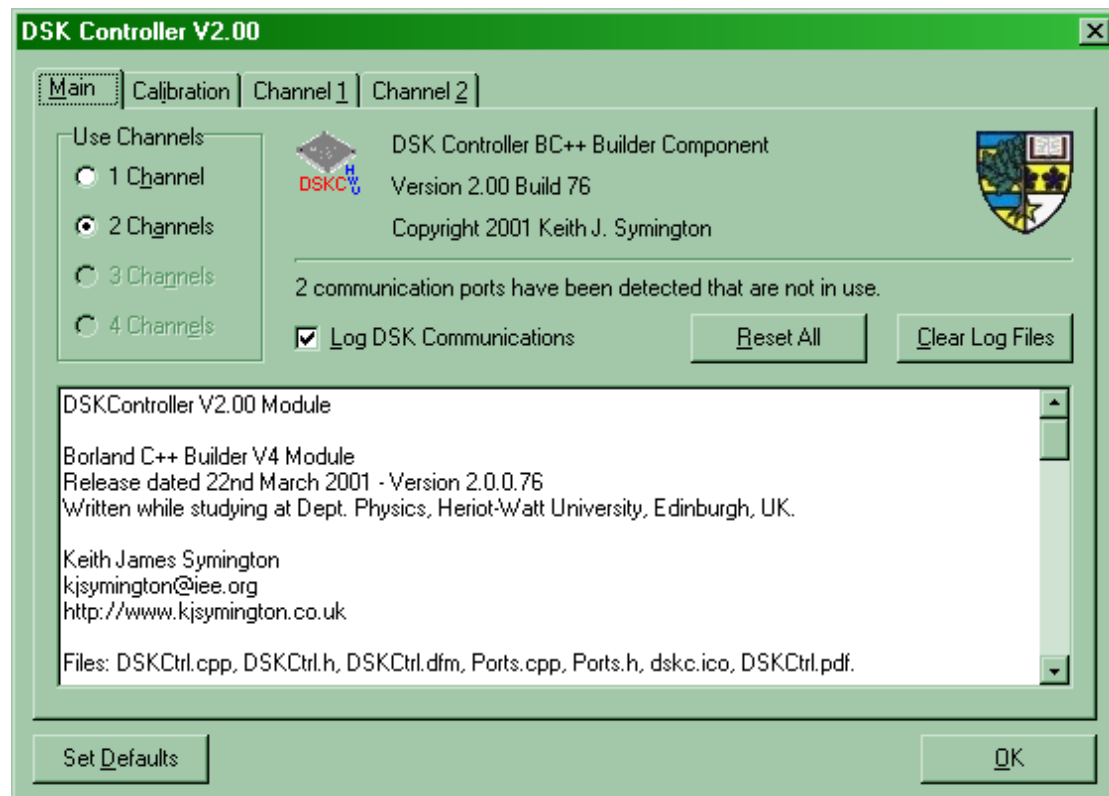
Copyright 2001 Keith J. Symington: Clicking on this will start composition of an e-mail to me.

Log DSK Communications: Check this box to log commands sent to the DSK on all channels. Logs are created in the appropriate window.

Reset All: Reset all active DSK channels.

Clear Log Files: Clear the log files for all DSK channels.

Text Window: Program information taken from the first chapter of this file.



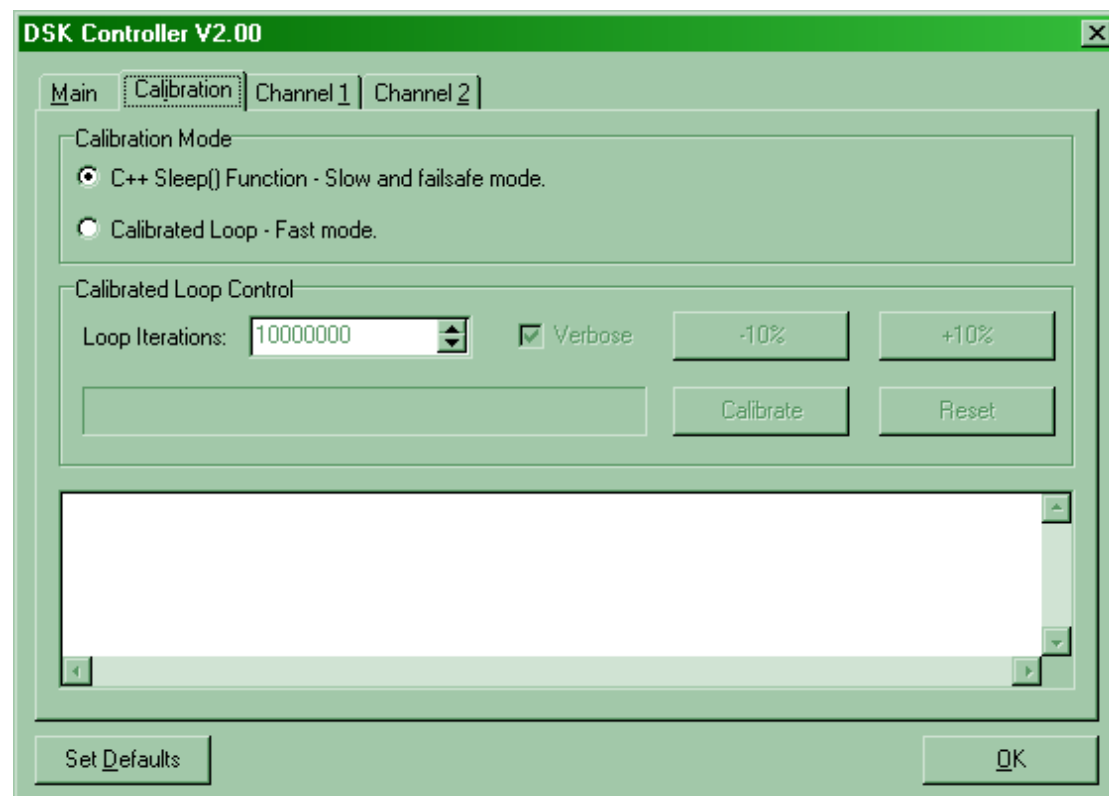
Set Defaults: Resets all tabs to the default settings. Use with care!!!

OK: Accept changes and close window.

Calibration Tab

Each of the controls in the calibration tab and their function is now explained.

Using Sleep() to communicate with the DSK is slow and inefficient. By calibrating a



delay loop to the machine, fast transmission rates can be achieved with a reduction in information transfer times of about 10 times. Unfortunately this requires a custom piece of code to optimise the delay loop. Calibration normally takes about 20 seconds. To calibrate, a working DSK needs to be attached to channel 1.

Calibration Mode: C++ Sleep() Function: This is the slow and failsafe mode for communication. Use if calibration proves to be a problem.

Calibration Mode: Calibrated Loop: Uses a delay loop to determine the transmission interval. This should work on most machines. Fluctuating processor load will cause problems in this mode. Ensure that "Calibrate" is clicked to optimise transmission before using this mode.

Loop Iterations: Number of loops to perform instead of using Sleep(). This value will be automatically determined on calibration. Can be adjusted manually although not recommended.

Verbose: If checked, error message dialog boxes will be generated by the application if an error occurs. Uncheck to handle an error in software.

-10%: Reduces the number of loop iterations by 10%.

+10%: Increases the number of loop iterations by 10%.

Calibrate: Starts calibration procedure. Usually takes about 20 seconds.

Reset: Sets default calibration values and clears any error messages.

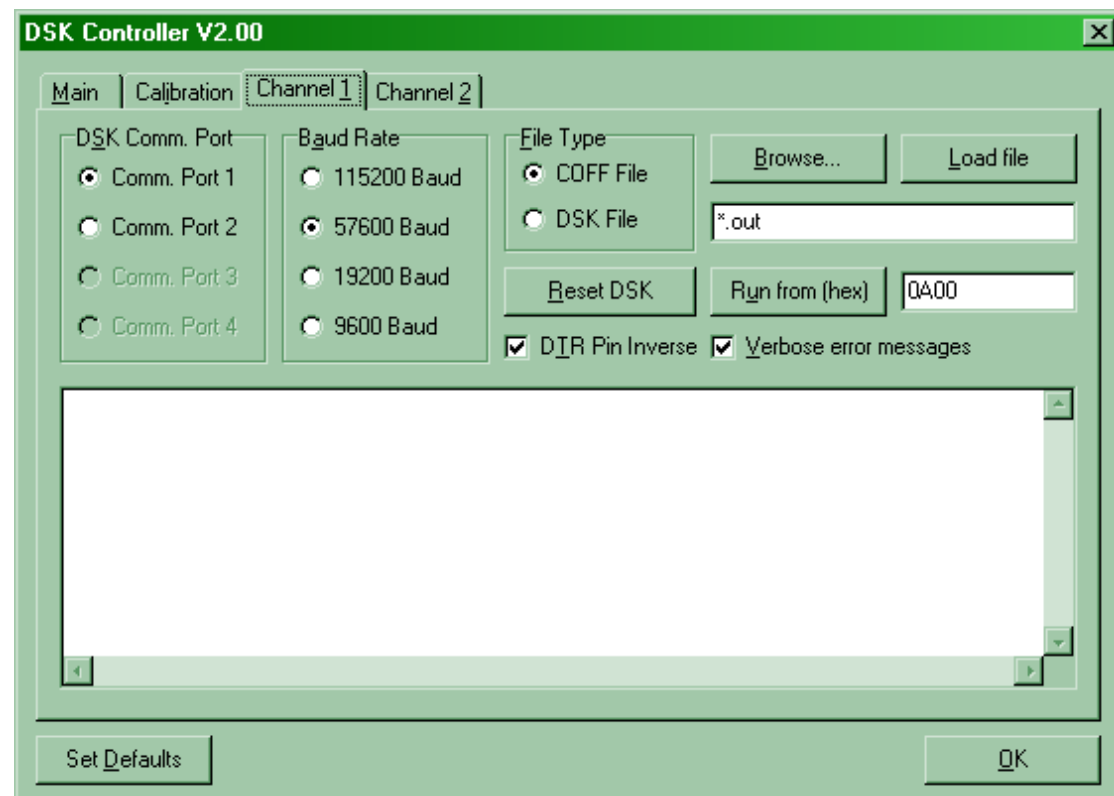
Text Window: Displays progress during calibration. Browse to find any errors.

Channel X

Each of the controls in a channel window are only relevant to that channel. There function is now explained.

DSK Comm. Port: These radio buttons allow the user to select the com port to which this channel's DSK is connected. Any ports that are greyed out are either not installed or currently in use by another program or driver.

Baud Rate: Used to select the communications speed with the DSK. This defaults to



57600 Baud as it seems to work most consistently. If there are problems with the DSK not receiving the correct data then try adjusting the baud rate.

File Type: This program supports both COFF and DSK file formats.

Browse...: Select the file to be loaded by browsing. The name then appears in the file name text box below.

Load File: Loads the file names in the file name text box below. Typing in the name and pressing enter has the same effect.

Reset DSK: Manual reset for the DSK if it has stopped responding.

Run From Hex: Runs the DSK from the memory address specified in this box. The code to be run must first be loaded. Note that this is a mask edit box so will only allow certain input.

DTR Pin Inverse: Allows the reset output (DTR line) to be inverted in case there is extra hardware between PC and DSK. If the DSK does not respond and it is correctly plugged in, toggle this box to see if there is any hardware inversion.

Verbose error messages: If checked, error message dialog boxes will be generated by the application if an error occurs. Uncheck to handle an error in software.

Log Window: Logs activity between PC and DSK.

Software Interface

Introduction

As this is a module it is designed too be integrated with other programs. To this end, there are a variety of procedures that can be used to control the DSK(s). This section describes the available functions and their associated parameters. Other procedures are available but it is recommended that only those documented here are actually used. Calling others may have an undetermined effect on program flow.

Installation

To incorporate this module into your program you need to copy 5 files to your project directory (DSKCtrl.cpp, DSKCtrl.h, DSKCtrl.dfm, Ports.cpp and Ports.h). Add both DSKCtrl.cpp and Ports.cpp to your new project. To use any procedure described here in your program simply include "DSKCtrl.h".

You can also display the form modally by calling:

```
DSKCtrlForm->ShowModal();
```

Notes on Calibration

There is not an interface procedure to calibrate the delay loop. This is because delay loop calibration is best overseen manually by the user. Therefore, calibration uses the default setting on startup of C++ Sleep() function.

If it is necessary to automate the calibration process then access the module as if you were performing the actions manually. To activate the calibrated loop set:

```
DSKCtrlForm->CalibrateRadioGroup->ItemIndex=1;
```

And to calibrate call:

```
DSKCtrlForm->CalibrateButtonClick(this);
```

Remember that this can take about 20 seconds.

To switch back into safe mode call:

```
DSKCtrlForm->CalibrateRadioGroup->ItemIndex=0;
```

portCount

```
int DSKCtrlForm->portCount
```

This variable indicates the number of com ports available to the program. It will lie within the range 0 to 4. Although a com port is installed, it need not necessarily be available as another program or driver may be using it.

errors

```
int DSKCtrlForm->errors
```

A useful variable that indicates the number of errors that occurred during the last operation. If this number is anything other than zero then an error has occurred. This variable does not however describe the error. Note that this variable may be set if the user sends commands manually - it is not specifically used by command automation of the module.

ChannelInit

```
bool DSKCtrlForm->ChannelInit(int channel, int commPort, int baud, bool DTRPin, bool verbose)
```

Initialises the specified channel with a given set of parameters. Use to initialise a DSK. This procedure alters settings in the related channel dialog box tab. The procedure takes five parameters.

channel: Channel to initialise. Can be CHANNEL1, CHANNEL2, CHANNEL3 or CHANNEL4.

commPort: Port on which channel should communicate. Can be either COM1, COM2, COM3 or COM4.

baud: Baud rate at which com port should communicate. Can be BAUD115200, BAUD57600, BAUD19200 or BAUD9600.

DTRPin: Should the DTR pin inverse be set or not. Can be true or false.

verbose: Should the form show an error dialog if problems occur. Can be true or false.

This procedure returns true if no errors occur or false otherwise. Running it will affect the value of errors.

LoadFile

```
bool DSKCtrlForm->LoadFile(int channel, AnsiString file, int fileType, bool verbose)
```

Loads the file specified after issuing a reset command to the DSK. The channel must have been initialised beforehand. This procedure alters settings in the related channel dialog box tab. It takes four parameters.

channel: Channel to load file to. Can be CHANNEL1, CHANNEL2, CHANNEL3 or CHANNEL4.

file: Path to the file that will be loaded. Long filenames are valid.

fileType: Is the file a COFF file (COFF_FILE) or is it a DSK file (DSK_FILE).

verbose: Should the form show an error dialog if problems occur. Can be true or false.

This procedure returns true if no errors occur or false otherwise. Running it will affect the value of errors.

RunDSK

```
bool DSKCtrl->RunDSK(int channel, AnsiString address, bool verbose)
```

Runs the DSK from the four digit hex address specified. This procedure alters settings in the related channel dialog box tab. The procedure takes three parameters:
channel: Channel to run DSK. Can be CHANNEL1, CHANNEL2, CHANNEL3 or CHANNEL4.

address: Address to run DSK from. Must be in string format and 4 characters long. For example, "0A00".

verbose: Should the form show an error dialog if problems occur. Can be true or false.

This procedure returns true if no errors occur or false otherwise. Running it will affect the value of errors.

EnableLogFiles

```
void DSKCtrlForm->EnableLogFiles(bool setting)
```

Alters status of logging on all channels. The procedure takes one parameter:

setting: If true then channel activity will be logged. If false, no logging will occur.

This procedure has no returns and does not affect the value of errors.

Receive

```
void DSKCtrlForm->Receive(int channel, unsigned int location,  
unsigned int *writeto, int numwords, bool stopDSK, bool  
startDSK, unsigned int runLocation)
```

Uploads information from DSK memory at a specified location. It works by stopping any running program, uploading the appropriate memory location(s) and restarting the DSK from a specified address if required. It takes seven parameters:

channel: Channel to receive information from. Can be CHANNEL1, CHANNEL2, CHANNEL3 or CHANNEL4.

location: The hex location that reading should start from in DSK memory.

writeto: A pointer to the location in PC memory where the correct number of words should be written to. Allocate the memory beforehand!

numwords: Number of words to be read from DSK location onwards.

stopDSK: If true then any running software on the DSK will be stopped before information is read. If the DSK is running then this parameter must be true otherwise an error will occur.

startDSK: If true, a run command will be sent to the DSK once the information has been read. The parameter runLocation specifies where to run from.

runLocation: The hex location that the program should be run again from. Need only be specified if startDSK is true, otherwise specify as NULL.

The procedure writes the received data to the writeto pointer. If any errors occur during transmission then errors will be incremented. The procedure has no return value.

Transmit

```
void DSKCtrlForm->Transmit(int channel, unsigned int location,  
unsigned int *writefrom, int numwords, bool stopDSK, bool  
startDSK, unsigned int runLocation)
```

Downloads information from PC memory to a specified location in DSK memory. It works by stopping any running program, downloading the appropriate memory location(s) and restarting the DSK from a specified address if required. It takes seven parameters:

channel: Channel to transmit information on. Can be CHANNEL1, CHANNEL2, CHANNEL3 or CHANNEL4.

location: The hex location that writing should start to in DSK memory.

writeFrom: A pointer to the location in PC memory where the correct number of words should be written from. The memory must have been allocated beforehand and the correct values set!

numwords: Number of words to be written to DSK location onwards.

stopDSK: If `true` then any running software on the DSK will be stopped before information is transmitted. If the DSK is running then this parameter must be `true` otherwise an error will occur.

startDSK: If `true`, a run command will be sent to the DSK once the information has been written. The parameter `runLocation` specifies where to run from.

runLocation: The hex location that the program should be run again from. Need only be specified if `startDSK` is `true`, otherwise specify as `NULL`.

The procedure transmits consecutive words from the `writeFrom` pointer. If any errors occur during transmission then `errors` will be incremented. The procedure has no return value.