

Introduction

The Universal Pointer Device Driver (UPDD) supports the use of BUS connected pointer devices. Unlike serial attached devices, which use RS232 there is no common protocol for bus connected devices. In order to support such controllers without the need to make driver code changes a macro language is used. This allows a developer to define the low level communication with the device.

Use by third parties

This macro language is intended primarily for internal use by Touch-Base Ltd. However it is made available for use by third parties on an unsupported basis. The language is not intended to be a complete programming language, but it supports a range of facilities needed to perform the task of bus interfacing. Touch-Base may extend this language if new requirements are found, but not if the task can be reasonably accomplished with the existing syntax.

Macros defined by a third party will be incorporated into the UPDD release product subject to a suitability review. The resulting UPDD configuration will then be supported through the usual support arrangements.

Limitations

In order to use the UPDD bus connection methodology a device must confirm to the following simple criteria. It must be attached to an interface bus, which can be addressed through the x86 in and out instructions. It must have a fixed and readily identifiable IO base address. It must raise a hardware interrupt when data is available for processing. It must be able to work within the confines of the protocol defined below. The device communication should not be dependent on exact timing. This is a limitation imposed by Windows.

From this it should be clear that most, if not all bus attached devices can be supported.

Protocol

In order to produce a generic solution a common protocol has to be identified. Due to the wide range of possible devices this protocol is very "thin". Touch-Base believes that this protocol should not preclude the use of any known bus attached pointer device. The UPDD bus protocol defines a number of events that relate to operation of a controller. A macro designer will create macro statements to respond to these events and perform the appropriate communication with the hardware. The following table describes the events involved and the action the associated macro section must perform.

Driver event	Typical macro action
OnInit	Perform any necessary hardware initialisation.
OnWrite	Optional. Write a single byte of data to the controller. Only used if initialisation macros are used.
OnEnableInterrupts	Instruct the controller to generate interrupts.
OnInterrupt	Read the interrupt data from the controller. Add packet data to the processing queue. Indicate to the controller that the interrupt has been serviced.

The bus macro syntax

Macro	Description	Example
:Label	Denotes a macro entry point or a target for a Goto instruction	:OnInit Initialisation processing :LOOP Some commands Goto LOOP
Set	Assign a value to a variable	Set Var1 1 Assigns the value 1 to the variable Var1 Set Var1 Var2 Copies the value of the variable Var2 to the variable Var1
And	Performs a logical bit wise AND operation on a variable	And Var1 1 Performs a bit wise AND operation on the value in Var1 and the constant value 1 and leaves the result in Var1 And Var1 Var2 Performs a bit wise AND operation on the value in Var1 and the value in Var2 and leaves the result in Var1
Or	Perform a logical bit wise OR operation on a variable	Or Var1 1

		Performs a bit wise OR operation on the value in Var1 and the constant value 1 and leaves the result in Var1
		Or Var1 Var2 Performs a bit wise OR operation on the value in Var1 and the value in Var2 and leaves the result in Var1
Put	Adds a single byte of data to the queue for protocol processing	Put Var1
Out	Performs an Out operation on the io port	Out P01 Var1 Performs an out operation at port base address+1 of the value in Var1. Out P02 CC Performs an out operation at port base address+2 of the hex value CC.
In	Performs an In operation on the io port	In Var1 P01 Performs an out operation at port base address+1 and stores the result in Var1
Equal	Tests the equality of 2 values. If the values are not equal the next statement is not executed	Equal Var1 Var2 Goto IsEqual Goto NotEqual Note the tested values can be variables or constants.
Goto	Indicates that macro execution will continue at the specified label	:Loop . . Goto Loop
Exit	Terminates execution of the macro routine and returns control to the driver	Exit
Inc	Increments the value of a specified variable	Inc Var1 Adds 1 to the value of Var1
Use	Sets the operation mode of the macro handler. The only currently supported mode is MMIO - which instructs the macro manager to use memory mapped IO.	Use MMIO
Wait	Halts the thread for the specified number of microseconds. NB this must not be used in the interrupt handler (OnInterrupt)	Wait 5000 Waits for 5 milliseconds
Fail	Terminates the execution of the current handler with a 'bad' status. If used during the "OnWrite" handler the initialisation macro that called it will be failed.	Fail

Notes on variables

- Variables are single byte registers.
- The names are case sensitive.
- The name must start with a letter.
- The name must be composed entirely of alphanumeric characters.
- It must not be a valid hex number (eg CC).
- It must not be a valid port identifier.
- The values have global scope, so - for example - a value is set in an initialisation is readable in the interrupt routine.

Notes on port identifiers

The port base address is specified during installation of the driver software – or through the hardware page of the DCU. Operations that act on a port (ie In and Out) use port identifiers to specify the exact address. A port identifier takes the form Pnnn, where nnn is a hex number (of 1 - 4 digits) specifying the offset from the base address.

If the base address specified during installation is 3f8 the operation will output the value 1 to port address 3f9.

General notes

- The macro is translated to a machine readable form to ensure speed of execution.
- Currently macros are limited to 255 lines and must not contain tab characters.
- BUS macros can be created in the TBArchitect (Lite) version that ships with the UPDD sdk.
- Debug information is output by the driver software at load time. By using a kernel debug message capture utility (eg DbgMon for NT or DbgTerm for 9x) a macro developer can see this output.
- You can force the re-execution of the initialisation portion of a bus macro by changing any setting in the DCU and hitting apply.

- All names and operations are case sensitive.

Contact

For further information or technical assistance please email the technical support team at technical@touch-base.com