Serial and USB Macros

Revision 1.2 – 26th June 2012
www.touch-base.com\documentation\technical

Introduction Definition Execution Commands Examples Contact

Introduction

The UPDD driver utilises 'Macros' to communicate with the touch devices. Macros are constructed using a mnemonic language. There are many reasons why the driver may need to communicate with a controller, such as setting the controller to run in a particular mode of operation, or to enable touch functionality or to send firmware commands to set or retrieve internal controller settings. Supported requests are dictated by the controller's functionality.

Some controllers respond to the commands to indicate the status or success of the requested command, others do not. If a controller is initialised by a UPDD macro the initialisation status of a controller is shown in the UPDD Console dialog.

Controller initialisation macros are created as required when we configure support for a controller, such as this macro that is being used to query a particular controller at driver startup to see if it responds such that the status in the UPDD Console and reflect the controller is responding:

[HEX]
5562000000000000000 //Query packet timing ['b']
[ACK 1000]42[END] //Look for response ['B'] in returning data

Macro Definition

Any macros that are set up for a controller will be held in the UPDD Settings file and can be modified or extended as required. A Macro utility is available to help construct macros. Developers using the UPDD API can invoke an API to process macros.

Macro execution

Macro commands can be invoked at various times, as described below:

Controller start

Issued each time the driver controller interface starts, i.e. at driver startup. The driver is re-started when UPDD driver settings are updated via the UPDD Console, calibration is initiated, or when the re-initialisation option is selected in the UPDD Console Status dialog.

Controller stop

Issued each time the driver controller interface stops, i.e. at driver shutdown. The driver is re-started when UPDD driver settings are updated via the UPDD Console, calibration is initiated. Issued prior to terminating communication with the controller.

Driver load

Issued whenever the driver loads. This is usually when the system starts but under Windows an administrator may stop and start the driver without stopping the system.

Driver unload

Issued whenever the driver unloads. This is usually, when the system shuts down but under Windows an administrator may stop and start the driver without stopping the system.

Firmware

A special macro generated by the Firmware page for controllers with firmware support. This macro is read only. It is executed only as part of one of the above macros, if the [FIRM] directive is encountered.

Macro Commands

When the macros are processed spaces are ignored, except in ASCII mode. Characters are sent to the output port as they are read, unless they are part of an ack/nak string or a macro command. All macro commands are entered between square brackets, i.e. [command].

Command	Description
//	denotes comment line - Only supported from version 2.07 eg // this is a comment [RTS 1] // this is also // so is this
[HEX]	denotes that the characters that follow are pairs of hex characters separated by commas. This is the default mode. The mode remains in effect until the end of the string, or until [ASCII] macro command is encountered. E.g. [HEX]01,02,03
[ASCII]	denotes that the characters that follow are ASCII characters. The mode remains in effect until the end of the string, or until [HEX] macro command is encountered. E.g. [ASCII]ABC
[CR]	send a carriage return (hex 0D) to the output port.
[LF]	send a line feed (hex 0A) to the output port.

[END]	denotes the end of an ACK or NAK string.
[ACK nnnn]	the characters that follow from an ACK string. Processing of the macro halts until each byte has been received in turn. The value nnnn represents a timeout value in milliseconds. If no value is specified the default value is 1 second. If the ACK in not received within the timeout period, a lower default time of 55ms is used for other ACKs in the macro and the macro initialisation status is set to "timed out". When an ACK is received, the list of NAK strings is cleared out. E.g.[ACK 1000][HEX]01,02,03[END]. The initialisation status is reflected in the UPDD Console, Status page.
[NAK text]	the characters that follow from a NAK string. They are read and stored. If the NAK string is read whilst waiting for an ACK, the macro processing is terminated and the initialisation status is set to text. Note that due to the way the macro is parsed any NAK commands must preceed the related ACK command, and before the transmission of data that might solicit the NAK. E.g. [NAK Failed] [HEX]04,05,06[END] [NAK Power low] 08,09,0A [END] The initialisation status is reflected in the UPDD Console, Status page.
[DTR n]	where n = 0 or 1. E.g.[DTR 1] sets the DTR line active (high). [DTR 0] sets the DTR line inactive (low).
[RTS n]	where n = 0 or 1. E.g.[RTS 1] sets the RTS line active (high). [RTS 0] sets the RTS line inactive (low). CTS and DSR are input lines and thus cannot be set in a macro
[WAIT nnnn]	pause for the specified number of milliseconds.
[BREAK n]	sets the UART break line to the value of n (0 or 1).
[FIRM]	Execute the firmware macro.
[DELAY nnnn]	There is a send macro API command. The Delay macro sets the delay between bytes send during macro playback. The default is 20ms, and it is rounded to the nearest 20 ms.
[USB]	[USB type=xxxxxxx flags=0 bits=0 req=0 val=xx idx=0]. The USB macro directive initiates a USB vendor or class request. It is followed by data to send (if any), and terminated by an [END] directive. The various subfields are described below:

type	Indicates the transfer type. It may be any of the following:	
	DEVICE	Request for a USB device.
	INTERFACE	Request for an interface on a USB device.
	ENDPOINT	Request for an endpoint, in an interface, on a USB device.
	OTHER	Request for a device-defined target.
flags	Indicates the transfer direction. It can only take the following values:-	
	0	Indicates a transfer from host to device.
	1	Indicates a transfer from device to host.
bits	Specifies the (hex) value of the URB field "ReservedBits".	
req	Specifies the (hex value) of the vendor defined request code.	
val	Specifies the (hex) value of the URB field "Value".	
idx	Specifies the device-defined identifier if the request is for an endpoint, interface, or device-defined target. Otherwise, Index must be 0.	

As of updd 4.1.3 we now support writing data to USB endpoint 2 using macros.

An example macro is shown.

USB request arguments (value, index etc) are irrelevant in this context and are ignored if specified.

[USB type=EP2 len=4]12 47 00 81[END]

Currently this is only available for endpoint 2.

The hardware interface is presently only supported for Windows.

[MODE] [MODE baud,parity,databits,stopbits]. Changes com port setup. Used on devices that switch speed during initialisation. E.g.[MODE 9600,N,8,1]
baud can be any valid baud rate
parity can be N / O / E (none, odd, even)
databits can be 7 or 8
stopbits can be 1 or 2
alternatively any value can be "*" - meaning use the defined hardware setting. E.g. [MODE *,*,8,1], means get baud and parity setting from the DCU hardware settings and set databits=8, stopbits=1.

This macro only applies to serial controllers. Using this for any other type of connection will result in a "not supported" macro status.

We recommend that for macros that change controller speed, the "operating speed" of the controller be set in the hardware settings, and the alternate speed be set at the start of the macro. By finishing up with [MODE *,*,*,*] the controller operating speed is reflected in the DCU hardware dialog. Doing this the other way round could lead to confusion.

E.g. a controller initially set to 1200,N,7,1 but eventually operates at 9600,N,8,1 could have 1200,N,7,1 in the hardware settings, and use [MODE 9600,N,8,1] in the macro. We would recommend setting 9600,N,8,1 in the DCU Hardware settings, then changing to 1200,N,7,1 then the initialisation sequence and 9600,N,8,1 in the macro.

Example

Example macro initialisation strings

Example 1 - Send string 0x01, 0x02, 0x03 to controller, no response expected.

01,02,03

Example 2 - Send string 0x01, 0x02, 0x03 to controller, controller response with 0x04, no response within 1 second indicates fail.

01,02,03[ACK 1000]04

Example 3 - Pulse CTS high for 100 milliseconds to reset controller, controller reponds within 0.5 seconds with hex 00 then send init string CR LF, controller responds with 0x01 for OK, 0x02 for low battery or 0x03 for general failure.

[CTS 1][WAIT 100][CTS 0][ACK 500]00[END][NAK Low battery]02[END][NAK Fail]03[END][ACK]01[END][CR][LF]

Example 4 - The following example macro sends a CLASS_INTERFACE request to a particular controller to place the device in multi-touch mode.

[USB type=CLASS_INTERFACE flags=0 bits=22 req=9 val=0302 idx=0 len=3]02 02 00[END]

Example 5 – Send a firmware command to a controller to run a specific command which, in this case, returns 8 bytes of data in the next request

[USB type=VENDOR_DEVICE bits=0 req=5 val=0 idx=0 flags=0 len=8]66 00 00 00 00 00 00 00[END] – send 8 byte command
[USB type=VENDOR_DEVICE bits=0 req=6 val=0 idx=0 flags=1 len=8][END] – retrieve response to above command

Contact

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