cmd_response

simple command/response access via USB to Arduino's I/O capabilities

The Arduino family of microcontrollers possess a reasonable collection of input and output capabilities. While there are different versions, each with some unique capabilities, most controllers have a USB port, some digital input and output ports, a few pulse-width modulated (PWM) digital ports, and some analog input ports. This code module programs the Arduino with a simple command/response language via USB providing access to Arduino's I/O capabilities.

Arduino Interface

short introduction here

Theory of Operations

> ***Note***
>
> TODO: this section needs to be written, for now, here are some notes

basic reads and writes are very short

USB commands have this form:

```
//  USB command interface of form: baseCmd [arg1 [arg2]]
//
// char* baseCmd (lower case)
// long arg1, arg2 (no octal or hex interpreted)
```

Commands are either **get** (begins with ?) or **set** (begins with !)

All **set** commands return "Ok" or an error message.

All **get** commands return a value or an error message.

ADC signal averaging requires some more setup

averaging all signals can make unit very slow

optimize by watching only the channels to be averaged

averaging occurs after each fixed-time period

user can query and adjust the time period

the time period can only be within a fixed range, the user can query the min and max of this range

user can start or stop watching a channel for averaging

reporting of average ADC value should be an integer (so EPICS can use this as the RVAL)

user can query and adjust a scaling factor (multiplier), as in: multiplier * sum(ADC)/count

the multiplier can only be within a fixed range, the user can query the min and max of this range

an ID and a version number are available to be read

Commands

?ai command

format: `?ai pin`
example: `?ai 0`
returns: `601`

Reads the analog input *pin* specified in the first argument. The returned value is between 0 and 1023 (10-bit ADC) which represents a measured voltage between 0 and 5 VDC. The pin must be within the range of `0..NUM_ANALOG_INPUTS`

?bi command

format: `?bi pin`
example: `?bi 3`
returns: `1`

Reads the digital input *pin* specified in the first argument. The returned value is either 0 or 1. The pin must be within the range of `0..NUM_DIGITAL_PINS`.

!bo command

format: `!bo pin value`
example: `!bo 6 0`
returns: `Ok`

Sets the digital output *pin* specified in the first argument to the *value* specified in the second argument. If the syntax is correct and the value is within range, returns `Ok`. The pin must be within the range of `0..NUM_DIGITAL_PINS` and must first be configured for output by calling `!pin pin 1`.

!pwm command

format: `!pwm pin value`
example: `!pwm 11 127`
returns: `Ok`

Sets the PWM-enabled digital output *pin* specified in the first argument to the *value* specified in the second argument. If the syntax is correct and the value is within range, returns `Ok`. The pin must first be configured for output by calling `!pin pin 1`.

The pin number is checked for PWM hardware-support by a call to the `digitalPinHasPWM(pin)` macro (which is defined by the Arduino SDK for each supported board type in <Arduino>/hardware/arduino/variants/*/pins_arduino.h).

!pin command

format: `!pin pin value`
example: `!pin 11 1`
returns: `Ok`

Configures the digital output *pin* specified in the first argument for output as a binary digital output and also as a PWM digital output if the hardware enables this.

value	meaning
0	use pin as input
1	use pin as output

Examples

1. Read analog input from pin 0:

```
>>> ?ai 0
41
```

2. Set digital pin 11 for PWM output:

```
>>> !pin 11 1
Ok
```

3. Set PWM output pin 11 to 128:

```
>>> !pwm 11 128
Ok
```

4. Show how a bad command (no space between baseCmd and pin) is handled:

```
>>> !pwm11 128
ERROR_UNKNOWN_COMMAND:!pwm11 128
```

Error messages

This is a list of the possible error messages and their meanings. All error messages begin with the text `ERROR_` and then some terse descriptor of the error. In most cases, the input that triggered the error message is returned. A single ":" is used as the delimiter when the input is appended.

`ERROR_AI_PIN_NOT_AVAILABLE:input`

The specified pin number is not available for analog input on this hardware.

`ERROR_AI_PIN_NOT_WATCHED:input`

The specified pin number was not set up for averaging. Need to call `!ai:mean pin 1` to enable averaging on this pin.

`ERROR_BINARY_RANGE:input`

The value must be either `0` or `1`.

`ERROR_BI_PIN_NOT_AVAILABLE:input`

The specified pin number is not available for binary (digital) input on this hardware.

`ERROR_BO_PIN_NOT_AVAILABLE:input`

The specified pin number is not available for binary (digital) output on this hardware.

`ERROR_BUFFER_OVERFLOW`

Too many characters were received before the line terminator. All characters received so far will be discarded.

Some Arduinos do not have much available RAM. The current buffer length is 40 characters.

ERROR_COMMAND_FORMAT:input

All commands must have at least one space separating the baseCmd from the pin number. This command is generated when no space is detected in the input.

ERROR_DIGITAL_PIN_NOT_AVAILABLE:input

The specified binary (digital) pin number is not available on this hardware.

ERROR_PIN_NOT_PWM:input

The specified pin is not supported for PWM on this Arduino hardware. This is determined by calling the Arduino system macro `digitalPinHasPWM(pin)` which is defined for each different type of Arduino hardware variation.

ERROR_NOT_IMPLEMENTED_YET:input

The command specified has not yet been implemented.

ERROR_PWM_RANGE:input

The PWM value must be between 0 and 255, inclusive. This error is reported for any value outside this range.

ERROR_TOO_MANY_ARGUMENTS:input

The general form for input commands is `baseCmd pin [value]` where `baseCmd` is given in the table above, `pin` is an integer appropriate for the chosen hardware interface, and `value` is only used for "!" (set) commands.

At this time, if `value` is specified for a "?" (read) command, it is ignored. In the future, this will generate an error message.

ERROR_UNKNOWN_COMMAND:input

The input was not recognized as a valid command. One reason for this might be the use of upper case. Other possibilities exist.

Tables

Table: Communications Parameters

term	value
communications rate	115200
line terminator	\n
buffer length (chars)	40
command length (chars)	16

Table: USB command interface

When present, "#" refers to the Arduino pin number used in the operation

command	value	action
?ai pin	0..1023	returns current value of numbered analog input
?bi pin	0..1	returns current value of numbered digital input
!bo pin v	0..1	sets numbered digital output to value v
!pwm pin v	0..255	sets numbered PWM digital output to value v

`!pin pin v`	0..1	sets mode of digital pin to value v (value: 1=OUTPUT, not 1=INPUT)
`?#ai`	int	returns NUM_ANALOG_INPUTS
`?#bi`	int	returns NUM_DIGITAL_PINS
`!t`	long	sets averaging time, ms
`?t`	long	returns averaging time, ms
`?t:min`	long	returns minimum allowed averaging time, ms
`?t:max`	long	returns maximum allowed averaging time, ms
`!k`	long	sets averaging factor (k)
`?k`	long	returns averaging factor (k)
`?k:min`	long	returns minimum allowed averaging factor (k)
`?k:max`	long	returns maximum allowed averaging factor (k)
`!ai:watch`	0..1	sets up ai pin for averaging
`?ai:mean`	long	returns `<ai>*k`
`?v`	long	returns version number
`?id`	0	returns identification string
`?rate`	long	returns number of updates (technically: loops) per microsecond
other		returns "ERROR_UNKNOWN_COMMAND:text"

notes:

1. must use lower case (as shown in table)
2. integers must be specified in decimal (no octal or hex interpreted)
3. pin numbers are not checked for correctness in the current version
4. "?" commands return an integer
5. "!" commands return "Ok"
6. Errors, starting with "ERROR_" will substitute for expected output