NAME

MESS − Multiple Encoded Switch State protocol

DESCRIPTION

The Multiple Encoding Switch State protocol is a method of encoding the state over time of a set of two-state switches (such as found in the MJS PedalBox device).

The protocol has been designed to represent the switch states both as binary bit-flags, one per switch; but to also be a printable-ASCII encoding using no unusual characters. These attributes allow the switch states to be easily interpreted, synchronously and efficiently with programming langauges supporting bit-wise operations (C, C++, Java, Korn Shell, etc) but also by text-based scripting languages (Bourne Shell, Perl, awk, Windows Scripting Host, 4DOS, et al).

The protocol is byte-size agnostic, subject to a minimum of 7 bits per byte.

SPECIFICATION

The protocol consists of a stream of "packets" each of which describes the instantaneous state of **all** switches in the source device.

Each packet consists of upto 126 payload bytes followed by an ASCII carriage-return character, in turn followed by an ASCII linefeed character.

Each payload byte represents the state of a group of four switches in the four least-significant bits, starting from the least-significant: for each switch in the group, a set bit indicates "on/closed" and a clear (zero) bit indicates "off/open/not-present". The 5th- and 6th-least-significant bit of each payload byte must be clear (zero), the 7th-least-significant bit must be set (one) and all more-significant bits of each byte must be clear (zero).

Consumers of the protocol must completely **ignore** any byte where the 7th-least-significant bit is clear (zero).

For devices with upto four switches, a single payload byte per packet is sufficient; for devices with more than four switches, each subsequent payload byte (within the same packet) encodes the state of the "next" group of four switches. Given the maximum packet-size of 128 bytes, the protocol can represent the state of a maximum of 504 switches simultaneously, per device.

EXAMPLES

In C, to determine whether **either** switch 1 **or** 3 is "on":

```
#include "mess.h"
char pkt[MESS_PKT_MAXLEN];
...
if ((pkt[0] & 0x4F) & 0x05)
```

To do the same, in UNIX Shell:

```
case "$pkt" in
[ACDEFGIKLMNO]*)    ... ;;
esac
```

In C, to determine whether **both** switches 1 **and** 3 are "on":

```
if ((pkt[0] & 0x4F) == 0x45)
```

and in the UNIX Shell:

```
case "$pkt" in
[EM]*)  ... ;;
esac
```

The packet contents for all possible states of a four-switch device are:

Switch Number				Packet Contents	
4th	3rd	2nd	1st	(hexadecimal)	(ASCII)
off	off	off	off	0x40 0x0d 0x0a	@\r\n
off	off	off	on	0x41 0x0d 0x0a	A\r\n
off	off	on	off	0x42 0x0d 0x0a	B\r\n
off	off	on	on	0x43 0x0d 0x0a	C\r\n
off	on	off	off	0x44 0x0d 0x0a	D\r\n
off	on	off	on	0x45 0x0d 0x0a	E\r\n
off	on	on	off	0x46 0x0d 0x0a	F\r\n
off	on	on	on	0x47 0x0d 0x0a	G\r\n
on	off	off	off	0x48 0x0d 0x0a	H\r\n
on	off	off	on	0x49 0x0d 0x0a	I\r\n
on	off	on	off	0x4a 0x0d 0x0a	J\r\n
on	off	on	on	0x4b 0x0d 0x0a	K\r\n
on	on	off	off	0x4c 0x0d 0x0a	L\r\n
on	on	off	on	0x4d 0x0d 0x0a	M\r\n
on	on	on	off	0x4e 0x0d 0x0a	N\r\n
on	on	on	on	0x4f 0x0d 0x0a	O\r\n

The packet contents for all possible states of a five-switch device are:

Switch Number					Packet Contents	
5th	4th	3rd	2nd	1st	(hexadecimal)	(ASCII)
off	off	off	off	off	0x40 0x40 0x0d 0x0a	@@\r\n
off	off	off	off	on	0x41 0x40 0x0d 0x0a	A@\r\n
off	off	off	on	off	0x42 0x40 0x0d 0x0a	B@\r\n
off	off	off	on	on	0x43 0x40 0x0d 0x0a	C@\r\n
off	off	on	off	off	0x44 0x40 0x0d 0x0a	D@\r\n
off	off	on	off	on	0x45 0x40 0x0d 0x0a	E@\r\n
off	off	on	on	off	0x46 0x40 0x0d 0x0a	F@\r\n
off	off	on	on	on	0x47 0x40 0x0d 0x0a	G@\r\n
off	on	off	off	off	0x48 0x40 0x0d 0x0a	H@\r\n
off	on	off	off	on	0x49 0x40 0x0d 0x0a	I@\r\n
off	on	off	on	off	0x4a 0x40 0x0d 0x0a	J@\r\n
off	on	off	on	on	0x4b 0x40 0x0d 0x0a	K@\r\n
off	on	on	off	off	0x4c 0x40 0x0d 0x0a	L@\r\n
off	on	on	off	on	0x4d 0x40 0x0d 0x0a	M@\r\n
off	on	on	on	off	0x4e 0x40 0x0d 0x0a	N@\r\n
off	on	on	on	on	0x4f 0x40 0x0d 0x0a	O@\r\n
on	off	off	off	off	0x40 0x41 0x0d 0x0a	@A\r\n
on	off	off	off	on	0x41 0x41 0x0d 0x0a	AA\r\n
on	off	off	on	off	0x42 0x41 0x0d 0x0a	BA\r\n
on	off	off	on	on	0x43 0x41 0x0d 0x0a	CA\r\n
on	off	on	off	off	0x44 0x41 0x0d 0x0a	DA\r\n
on	off	on	off	on	0x45 0x41 0x0d 0x0a	EA\r\n
on	off	on	on	off	0x46 0x41 0x0d 0x0a	FA\r\n
on	off	on	on	on	0x47 0x41 0x0d 0x0a	GA\r\n
on	on	off	off	off	0x48 0x41 0x0d 0x0a	HA\r\n
on	on	off	off	on	0x49 0x41 0x0d 0x0a	IA\r\n
on	on	off	on	off	0x4a 0x41 0x0d 0x0a	JA\r\n
on	on	off	on	on	0x4b 0x41 0x0d 0x0a	KA\r\n
on	on	on	off	off	0x4c 0x41 0x0d 0x0a	LA\r\n
on	on	on	off	on	0x4d 0x41 0x0d 0x0a	MA\r\n
on	on	on	on	off	0x4e 0x41 0x0d 0x0a	NA\r\n
on	on	on	on	on	0x4f 0x41 0x0d 0x0a	OA\r\n

SEE ALSO

pdlbox(1), **mess_avc**(1)

BUGS

This protocol does not distinguish between "off" and "not present" states, thus, for example, a four-switch device with the 3rd and 4th switches in the "off" state cannot be programatically distinguished from a (merely) two-switch device.