IPAC Programming Protocol

All programming info is sent by the use of the keyboard LEDs. The three LEDs are used to send each byte in 3 chunks. The first chunk is the 3 least significant bytes, the next chunk is the 3 middle bytes and the third chunk is the two most significant plus one parity bit. (the parity is actually not checked in the present I-PAC code) This is the only method of sending data to the keyboard which is supported in the USB standard.

Bit layout:

Scroll lock	Bits 0,3,6
Num lock	Bits 1,4,7
Caps lock	Bits 2.5.P

LED codes consist of 2 bytes as per the IBM spec. Each byte is acknowledged by the I-PAC and the "ack" must be waited for before sending the next byte.

The download is started by the following sequence:

PC sends any LED code except "all LEDS on"
PC sends 5 successive "all LEDs on" codes.
I-PAC returns a character. This is as follows:
"Y" means "OK, ready for code block"
"J" means "Jumper set to MAME, cannot program"
If either of these codes is not received within 500ms, the program should error with a timeout.

Then a 66 byte code block is sent as follows:

Byte 1: Shift number.
Bytes 2-65: key codes
Byte 66: Checksum

The above are defined as follows:

Shift Number:
This is the number which corresponds to the I-PAC input connection which is being used as the shift button. These numbers are shown in the right-hand column of the table below. The number is between 1 AND 1Ch. If no shift key is desired, this should be FFh.

Key codes:
The key codes are sent starting with the top line in the table below and continuing on the order shown. The table shows the following:
Left column: This shows the codes which are part of the default MAME set in ROM. These are not significant here. Note the codes are the IBM internal scan codes not the codes which are normally used when writing programs on the PC which access the keyboard. The codes are documented in a Microsoft paper here:
http://www.microsoft.com/hwdev/input/download/translate.pdf
The code set used is "PS/2 set 2" which is the only set supported by Windows.
Next column shows the inputs as they are marked on the I-PAC PCB.
The next shows the function which is part of the MAME default set. Not significant here.

The next shows the I-PAC chip port and bit number. Not important here.
The next shows the “input number” which is used to reference the shift button.
Note the reason for the strange order of the codes is because they are determined by the physical track layout on the PCB. The inputs were arranged to enable the PCB to be routed 100% without jumpers on a single-sided PCB.
Dummy Bits: Important. The bits marked with *** are not used in determining key codes. They exist for the same reason as above, PCB layout, and are used for the aux keyboard input and jumpers. These 4 bits must be set to 00 in the code block. They must not be left out otherwise the codes will get out of sync. (Note: this is needed for backwards compatibility, I would not have done this if starting from scratch!)
“E0” codes: In the IBM key code table there are some keys which send an “E0” in front of the normal code. Examples are the arrow keys and right ALT and CTRL. For these codes, bit 7 should be set in the code table. For example left arrow becomes EBh instead of 6Bh. The print-screen and pause keys have multi-byte codes and are not supported. The Numpad “/” key has a totally weird code string and is also not supported.

Checksum:
This is calculated over the entire 65 bytes. It is a “checksum-8” which is simply adding each byte into an 8 bit register and throwing away the carry.

I-PAC Reply:
After receiving all 66 bytes the I-PAC checks the checksum. It returns one of the following characters:
“Y” = “all OK”
“C” = “checksum error”
The program can respond based on these codes. If it does not get any code within 500ms it should generate a “timeout” error.

The program then needs to re-set the keyboard LEDs to the value they were before programming.

The program needs to have a delay setting switch. This introduces a delay between the sending of every byte. The reason for this is that some PCs have problems sending at the full speed especially in USB mode. This seems to be caused by bugs in the PC BIOS.

Unshifted keys

DB	0F5H	;1 UP	UP	PORT0 BIT0	1
DB	0F4H	;1 RIGHT	RIGHT	PORT0 BIT1	2
DB	14H	;1 BT 1	L-CTRL	PORT0 BIT2	3
DB	0EBH	;1 LEFT	LEFT	PORT0 BIT3	4
DB	29H	;1 BT 3	SPACE	PORT0 BIT4	5
DB	0F2H	;1 DOWN	DOWN	PORT0 BIT5	6
DB	1AH	;1 BT 5	Z	PORT0 BIT6	7
DB	11H	;1 BT 2	ALT	PORT0 BIT7	8
DB	34H	;2 RIGHT	G	PORT1 BIT0	9
DB	12H	;1 BT 4	L-SHIFT	PORT1 BIT1	A
DB	23H	;2 LEFT	D	PORT1 BIT2	B
DB	22H	;1 BT 6	X	PORT1 BIT3	C
DB	2DH	;2 UP	R	PORT1 BIT4	D
DB	00	;AUTO	***	PORT1 BIT5	E
DB	2BH	;2 DOWN	F	PORT1 BIT6	F
DB	00	;JP1	***	PORT1 BIT7	10
DB	1CH	;2 BT 1	A	PORT2 BIT0	11
DB	16H	;START 1	1	PORT2 BIT1	12

DB	1BH	;2 BT 2	S	PORT2 BIT2	13
DB	1EH	;START 2	2	PORT2 BIT3	14
DB	15H	;2 BT 3	Q	PORT2 BIT4	15
DB	2EH	;COIN 1	5	PORT2 BIT5	16
DB	1DH	;2 BT 4	W	PORT2 BIT6	17
DB	36H	;COIN 2	6	PORT2 BIT7	18
DB	43H	;2 BT 5	I	PORT3 BIT0	19
DB	21H	;1 BT 7	C	PORT3 BIT1	1A
DB	42H	;2 BT 6	K	PORT3 BIT2	1B
DB	2AH	;1 BT 8	V	PORT3 BIT3	1C
DB	3BH	;2 BT 7	J	PORT3 BIT4	1D
DB	00	;AUXCLK	***	PORT3 BIT5	1E
DB	00	;AUXDATA	***	PORT3 BIT6	1F
DB	4BH	;2 BT 8	L	PORT3 BIT7	20

Shifted keys

DB	2AH	;1 UP	V	PORT0 BIT0	21
DB	22H	;1 RIGHT	X	PORT0 BIT1	22
DB	1AH	;1 BT 1	Z	PORT0 BIT2	23
DB	35H	;1 LEFT	Y	PORT0 BIT3	24
DB	00	;1 BT 3		PORT0 BIT4	25
DB	1DH	;1 DOWN	W	PORT0 BIT5	26
DB	00	;1 BT 5		PORT0 BIT6	27
DB	00	;1 BT 2		PORT0 BIT7	28
DB	00	;2 RIGHT		PORT1 BIT0	29
DB	00	;1 BT 4		PORT1 BIT1	2A
DB	00	;2 LEFT		PORT1 BIT2	2B
DB	00	;1 BT 6		PORT1 BIT3	2C
DB	00	;2 UP		PORT1 BIT4	2D
DB	00	;AUTO	***	PORT1 BIT5	2E
DB	00	;2 DOWN		PORT1 BIT6	2F
DB	00	;JP1	***	PORT1 BIT7	30
DB	00	;2 BT 1		PORT2 BIT0	31
DB	16H	;START 1		PORT2 BIT1	32
DB	00	;2 BT 2		PORT2 BIT2	33
DB	76H	;START 2	ESC	PORT2 BIT3	34
DB	00	;2 BT 3		PORT2 BIT4	35
DB	00	;COIN 1		PORT2 BIT5	36
DB	00	;2 BT 4		PORT2 BIT6	37
DB	00	;COIN 2		PORT2 BIT7	38
DB	00	;2 BT 5		PORT3 BIT0	39
DB	00	;1 BT 7		PORT3 BIT1	3A
DB	00	;2 BT 6		PORT3 BIT2	3B
DB	00	;1 BT 8		PORT3 BIT3	3C
DB	00	;2 BT 7		PORT3 BIT4	3D
DB	00	;AUXCLK	***	PORT3 BIT5	3E
DB	00	;AUXDATA	***	PORT3 BIT6	3F
DB	00	;2 BT 8		PORT3 BIT7	40

Two-Board support (already implemented)

For programming two boards together, the first step is a once-only step done when the user receives the board. This must be done with one board only connected. It permanently changes the board to “board number 2”. Command-line switch is /P:2. This setting is retained by the board for ever.
send a normal programming cycle but with the shift key number set to FE to set the board to "board 2".

send a programming cycle with shift number set to FD resets back to board 1
(in theory never needed): Command-line /P:1

Once set to board number 2, it will respond as follows:

To program board 2,(command line switch /2) instead of initiating the program mode by sending 5
"all on" LEDs, send instead 5 "caps lock off, other 2 LEDs on" commands
(011b)

I am going to do an I-PAC with 56 inputs which will actually be 2 I-PACs in one. It will conform to this spec unchanged. The inputs on the second half will be marked player 3 and player 4, coin 3, coin 4 start 3 start 4.

Future Plans:

The following functions are not implemented in IPACUTIL or in the I-PAC. If any program is written which uses this function I will add it into I-PAC.

Macros:

Macros for a maximum of 4 keys with 3 keystrokes each can be implemented by sending a second 16 byte code block as follows:

Initial handshake as above.
First byte of the code block is Feh to signify a macro block. (in place of the shift input number)
Follow with 4 blocks of 4 bytes for macro codes.

Follow with checksum.

In the main code block download, codes which have macros associated with them have a value between A0 and A3 instead of the normal code. The I-PAC will recognise this range and look in a macro table for the string. The code A0 will look at the first macro, A1 the second, etc.

Two-Board GUI (not yet implemented)

We already have the ability to program each board by running the program twice, once with the switch “/2”. The GUI needs to be changed to program both boards in one hit. Two more pages will be needed, for player 3,4 start 3,4 etc shifted and unshifted. (Note use of the term pages rather than boards because this functionality will also be used for the 4-player I-PAC).
Two options: more tabs across the top or a button in the lower part, pressed gives page 2.

Things not yet thought about (ideas welcome!): Config file. How to merge and detect single/double file etc.
Also can one program handle both single and double board set-ups or do we need 2 programs? (much easier with 2 but then 2 code versions to support).

Config files:

A config file format with comments to help editing (ignored when programming).

Stuck-key detection: The MAC version has this and it’s a nice addition.