БЪЛГАРСКА АКАДЕМИЯ НА НАУКИТЕ. BULGARIAN ACADEMY OF SCIENCES

ПРОБЛЕМИ НА ТЕХНИЧЕСКАТА КИБЕРНЕТИКА И РОБОТИКАТА, 53
PROBLEMS OF ENGINEERING CYBERNETICS AND ROBOTICS, 53

София . 2002 . Sofia

Software for Making Commands in Unitary Code

Rumiana Bozhkova

Central Laboratory of Mechatronics and Instrumentation, 1113 Sofia

1. Preface

The computerized system for remote control generally consists of encoding-transmitting and receiving hardware, controlling software, and mechatronical device – Fig 1, where 1 is controlling software; 2 – transmitting hardware; 3 – receiving hardware; 4 – mechatronical device.

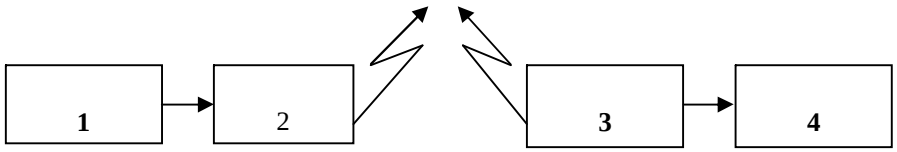

Fig. 1. Block diagram of the computerized system for remote control

During the past years the operating system Linux gained great popularity and wide spreading. Its characteristics like safety, stability, reliability, as well as the fact that it is practically free of charge makes it very attractive.

2. Description of the parallel port

A parallel port links software to the peripheral devices. To software, the parallel port is three 8-bit registers occupying three consecutive addresses in the I/O space. To hardware, the port is a female 25-pin D-sub connector,

carrying twelve latched outputs from the computer, accepting five inputs into the computer, with eight ground lines (pins 18-25). Table 1 shows the pinout.

Table 1

Pin	Signal	Description
1	STROBE	input/output
2	D0	output
3	D1	output
4	D2	output
5	D3	output
6	D4	output
7	D5	output
8	D6	output
9	D7	output
10	ACK	input
11	BUSY	input
12	NOPAPER	input
13	SELECTED	input
14	AUTOFEED	input/output
15	ERROR	input
16	INITIALIZE	input/output
17	SELECT	input/output
18	ground	
...		
25	ground	

A parallel port is identified by its **I/O base address**, and also by its LPT **port number**. The BIOS power-on self-test checks specific I/O addresses for the presence of a parallel port, and builds a table of I/O addresses in the low memory BIOS data area, starting at address 0040:0008 (or 0000:0408).

The parallel port I/O address table contains up to three 16-bit words (four on some BIOSes). Each entry gives the I/O base address of a parallel port. The first word is the I/O base address of LPT1, the second is LPT2, etc. If less than three ports were found, the remaining entries in the table are zero. DOS, and the BIOS printer functions (accessed via int 17 h), use this table to translate an LPT port number to a port address, to access the appropriate physical port.

The power-on self-test checks these addresses in a specific order, and addresses are put into the table as they are found, so the table will never have gaps. A particular I/O address does not necessarily always equate to the same specific LPT port number, although there are conventions.

The video cards parallel port is normally at 3BCh. This address is the first to be checked by the BIOS, so if a port exists there, it will become LPT1. The BIOS then checks at 378 h, then at 278 h. I know of no standard address for a fourth port.

A parallel port consists of three 8-bit registers at adjacent addresses in the processor's I/O space. The registers are defined relative to the **I/O base address**, and are at IOBase+0, IOBase+1 and IOBase+2 (for example if IOBase is 3BCh, then the registers are at 3BCh, 3BDh and 3BEh).

The *data register* is at IOBase+0. It may be read and written (using the IN and OUT instructions, or inportb( ) and outportb( ) or inp( ) and outp( )). Writing a byte to this register causes the byte value to appear on the data signals, on pins 2 to 9 inclusive of the D-sub connector (unless the port is bidirectional and is set to input mode). The value will **remain latched** and stable until a different value is written to the data register. Reading this register yields the state of the data signal lines at the time of the read access.

Data register: LPTBase+0, read/write, driven by software (driven by hardware in input mode) (Table 2).

Table 2

7	6	5	4	3	2	1	0	Name	Pin
*								D7	9
	*							D6	8
		*						D5	7
			*					D4	3
				*				D3	5
					*			D2	4
						*		D1	3
							*	D0	2

The status register is at IOBase+1. It is read-only (writes will be ignored). Reading the port yields the state of the five status input pins on the parallel port connector at the time of the read access (Table 3).

Status register: *LPTBase+1*, read-only, driven by hardware.

Table 3

7	6	5	4	3	2	1	0	Name	Pin
*								**BUSY**	11
	*							**-ACK**	10
		*						**NOPAPER**	12
			*					**SELECTED**	13
				*				**–ERROR**	15
					*	*	*	undefined	

The control register is at IOBase+2. It can be read and written. Bits 7 and 6 are unimplemented (when read, they yield undefined values, often 1,1, and when written, they are ignored). Bit 5 is also unimplemented on the standard parallel port, but is a normal read/write bit on the PS/2 port. Bit 4 is

a normal read/write bit. Bits 3, 2, 1 and 0 are special – see the following section.

Control register: *LPTBase+2*, read/write (see Table 4), driven by software and hardware.

Table 4

7	6	5	4	3	2	1	0	Name	Pin
*	*							**Undefined**	-
		*						**Inp mod**	-
			*					**IRQ enable**	-
				*				**-SELECT**	17
					*			**INITIALIZE**	16
						*		**AUTOFEED**	14
							*	STROBE	1

3. How the software works

Normally the port sends information to the peripheral device through the eight data lines D0-D7 – the pins from 2 to 9. Other pins are used for controlling data flow and indication of various events. To be able to control our system, we need to make the port working in a different way. We need to control individual pins on the parallel port. So we need to program directly the parallel port registers. From the above information it is clear, that from all the 25 pins, 8 are grounds, 5 are only input pins – the pins 10, 11, 12, 13 and 15. There remain 12 pins, usable for sending data to the mechatronical device.

Because the transmitting and receiving hardware works with 9-bit word, we need 9 pins for control. So we use the eight data-pins – the pins from 2nd to 9th pin, and the first pin- **"STROBE"** – in other words the software controls the data register **IOBase+0**, and the control register **IOBase+2**. We have to remark, that the data-pins 2–9 are gathered. That is, their directions are not individually controllable; they must be either all inputs or all outputs [2].

Note: There are old parallel ports, which do not support switching between inputs and outputs on pins 2–9 at all, so pins 2–9 are always outputs.

As an interface between the program and the user we used the **TurboVision** libraries.

We made two variants of the program – the first one, working on the PC with installed Linux, the other one – mobile – installed on 3 bootable floppy disks. The second one allows the program to be run on every PC, independent on the operating system, installed on it. Because of the lack of space on the floppy disks, we used in this variant of the program the lightened shell **ash**, instead of the standard for the Linux, **Bourne-Again** shell-**bash.**

4. The interface

The program has two menus which can be used by the mouse, or by the keyboard – the button F10 activates the menus, and the arrows Left and Right may be selected in the proper menu. The first one - for changing the directory, output to the shell without closing the program and end of the program. The other menu opens a window with buttons, related to the control functions (Table 5).

Table 5

	Button	**Unitary code**
	STOP	100000000
	Speed 1	010000000
	Speed 2	001000000
	Reverse 1	000100000
	Speed 3	000010000
	Speed 4	000001000
	Speed 5	000000100
	Reverse 2	000000010
	Trajectory	000000001

The control buttons may be selected by the mouse, or by the keys “Tab” and “Enter”.

In Fig. 2 the graphical interface of the control software is shown schematically.

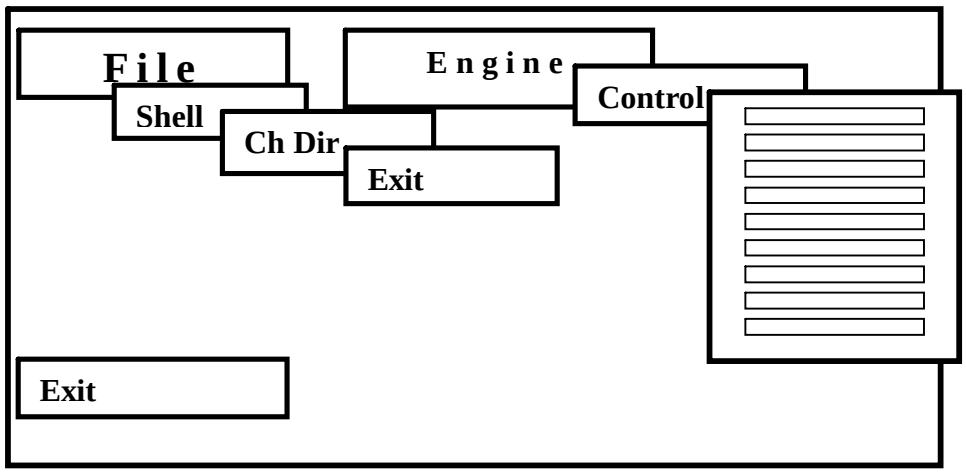

Fig. 2. Graphical interface

R e f e r e n c e s

1. Kris Heidenstrom's PC Parallel Port Mini-FAQ – http://home.clear.net.nz/pages/kheidens/

2. IBM Parallel Port FAQ/Tutorial Version 0.96 9/1/94 by Zhahai Stewart.

hftp://ftp.rmii.com/pub2/hisys/parport/

Софтвер для генерирования команд в унитарном коде

Румяна Божкова

Центральная лаборатория мехатроники и приборостроения, 1113 София

(Р е з ю м е)

Описывается софтвер для генерирования команд в унитарном коде, используемый в модуле компьютерной системы для дистанционного управления мехатронных устройств. Команды представляют 9-битовые слова, которые передаются в PC в двоичном коде. Связь с передатчиком осуществляется при помощи паралельного порта PC, так как необходимо передавать 9 битов в одном и тоже времени. Закодирование команд осуществляется программой в языке C^{++}, работающей в операционной среде Linux. Програма высилает числа в унитарном коде на определенных выводах паралельного порта. Выводы поставлены на каждых 10 ms – время, необходимо для хардуера.