(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: **0 650 132 A1**

(12) # EUROPEAN PATENT APPLICATION

(21) Application number: **93117069.0**

(51) Int. Cl.[6]: **G06F 19/00**

(22) Date of filing: **21.10.93**

(43) Date of publication of application:
26.04.95 Bulletin 95/17

(84) Designated Contracting States:
DE DK ES FR GB GR NL PT SE

(71) Applicant: **COMPAGNIA INVENZIONI E BREVETTI ITALIANA S.r.l.**
Via P. Spino 28
I-24126 Bergamo (IT)

(72) Inventor: **Bertillo, Gaudenzio**
Via P. Spino 28
I- 24126 Bergamo (IT)

(54) **Electronic system for the control and elaboration in real time of the data typical of "Totocalcio" traditional system predictions.**

(57) The apparatus in question was produced to enable the "sistemista" (the Totocalcio player) to monitor in real time the score predicted by his own forecast. As it can be seen by the following description, such appliance consists of a programmed electronic device.

Its working is based on digital elaboration of the data inserted by the user through a keyboard. A liquid crystal display (lcd) permits to monitor in real time all the scores during a complete football day, divided game per game according to the "Totocalcio" football match calendars.

Setting hypothetical predictions, by systems with a maximum of three variables for each prediction, this device is able to give the user the whole and the single favourable events, distributed among the several systems, predicted by the forecast itself.

Furthermore, through more powerful elaboration data devices, it is possible to solve a considerable number of statistical, data elaboration and data memorization functions.

EP 0 650 132 A1

FOREWORD

The apparatus in question was produced to enable the "sistemista" (the Totocalcio player) to monitor in real time the score predicted by his own forecast. As it can be seen by the following description, such appliance consists of a programmed electronic device.

Its working is based on digital elaboration of the data inserted by the user through a keyboard. A liquid crystal display (lcd) permits to monitor in real time all the scores during a complete football day, divided game per game according to the "Totocalcio" football match calendars.

Setting hypothetical predictions, by systems with a maximum of three variables for each prediction, this device is able to give the user the whole and the single favourable events, distributed among the several systems, predicted by the forecast itself.

Furthermore, through more powerful elaboration data devices, it is possible to solve a considerable number of statistical, data elaboration and data memorization functions.

DESCRIPTION

HARDWARE DESCRIPTION

to make it more easily comprehensible, this description has been divided in three parts, corresponding to the single electronic circuit diagrams.

The diagrams are divided as follows:

1) keyboard
2) Control logic
3) Power supply

KEYBOARD

The keyboard is composed of thirty-two function and entry keys, plus a thirty-third one to turn on/off the apparatus.

All the keys, except the last one, are linked together by a matrix organized in four columns and eight rows.
they, so linked, can be easily located by a dedicated circuitry described hereunder.

CONTROL LOGIC

The device control logic is based on an integrated MPU circuit (Micro Processor Unit); its task is to perform, by a means of a program, determined functions according to the key digitized by the user.

The main characteristics of this integrated circuit are the following:

- system architecture	:8 bit
- clock frequency	:8 MHz
- supply voltage	:3-6 V
- program ROM memory (Read Only Memory)	:1828 bytes
- data ROM memory	:64 bytes
- data RAM memory (Random Access Memory)	:64 bytes
- user programmable I/O	:20
- not maskable interrupt	:1

Every time the user presses a key, the integrated U2 circuit dedicated to the matrix keyboard decodification, will inform, via a specific signal (Data Available), the MPU that, on the data pins of the decodification itself, a binary code, corresponding to the key digitized, is present. The identification of this particular event by the MPU takes place by means of a dedicated input pin (NMI) which, for each falling edge of the signal applied, sets the MPU in a particular operative cycle, called interrupt. The task of the four inputs NAND, which is connected as an inverter between the D.A. and NMI signals, is to correctly interface the two integrated circuits, since the D.A. pin validates the data present on its output pins with a rising edge of the signal, while the MPU needs a falling edge of the signal to initialize the interrupt cycle.

The recognition of the key pressed, by the integrated circuit decoder, is accomplished through a multiplexer. This device sets the proper output pins, connected with the columns of the keyboard, at logic level 0 in a sequential and cyclical trend. Every time a key is pressed, the row input correlated with it is set at logic level 0. Therefore, since the integrated circuit has identified the row/column key co-ordinates, it will convert these data into a binary code according to the key digitized and will set at high level the output D.A. signal.

However, the keyboard integrated circuit decoder works as interface for only sixteen keys. But the expansion concerns thirty-two keys, that is the number of the keys necessary to operate the device; this expansion was obtained by means of a network composed by D2, D3, D4, D5, R5, R6, R7, U3A. every time a key is pressed, it will or will not set the NAND U3A output at logic level 1, according to the value given to the key: if this is higher than sixteen, the output will have a logic level 1.

Once the data relating to the key pressed is introduced into the MPU, through the input/output port configured as PA input, the software will recognize the key either as a team key or as a function key. Once the kind of the key digitized has been recognized, the software will perform the operations related to the key pressed.

If for example, the key pressed is the one which increases the number of the goals scored by a team, the program will increase the score and will check the results displaying the symbol used to decide the outcome of the game.

If, on the contrary, it's a function key, the program will perform the appointed operations.

They are:

A) variables line position increment,

B) variables setting,

C) change page.

The device has a decrement function operating on all the increment ones in order to correct every mistake, if any, and to change uncertain data, if any.

Once the requested function is performed, the favourable predictions are checked and counted and then displayed, after which the cycle is ended and the program awaits for the pressing of a further key.

the commercial version display we use has its own control logic, prescribed by the CMOS LSI controller and dedicated to this kind of functions, located on the display control board.

This controller works as a visualization processor and converter of the data supplied by the MPU into character data to be visualized. This procedure is made possible through a set of hexadecimal software commands which implement a display mini-operative system. The main software commands this MPU uses are the ones which define the display point where to position the cursor and to write the characters.

the following is the command sequence used in our system to display the characters : bit BUSY reading, operating code relating to least significant address writing, least significant address, bit BUSY reading, operating code relating to Most significant address writing, Most significant address, bit BUSY reading, operating code relating to data writing or reading, character code writing or reading.

The display control signals are supplied to the controller through the two MPU PB and PC ports, which perform the following functions respectively: bi-directional data bus, control signals of the various data and operative codes writing and reading cycles. The two ports transfer the signals elaborated by the MPU to the display by means of the JP1 output connector.

The clock frequency, required by the MPU to time its own internal operations, is supplied to the integrated circuit by the resonance circuit composed by C1, C2, X1, while the missing part necessary for a proper oscillation of the system is inside the integrated circuit itself.

Through the circuital network composed by C3, R1, D1, SW4, it is possible to reset the appliance by pressing the key.

The device is also able to produce sounds by means of the piezoelectric buzzer commanded by the Q1 transistor and the relevant polarizer circuitry composed by R3, R4.

the MPU starts the buzzer sound through the "TIMER" pin, which is connected to a timer located inside the integrated circuit; the sound duration is determined by proper waiting cycles set in the software.

The LCD display requires an adjustment of the crystal polarization tension to reach the correct contrast level. The regulation is carried out by the trimmer R2 connected between the negative and positive power supply.

POWER SUPPLY

The power supply part which supplies the correct voltage to the various components consists of series stabilization power supply components (for the positive voltages). The output voltages are +8 and +5,

while for the - 13.5Volt negative voltage required by the LCD display, a DC-DC converter consisting of an oscillator and a tension-quadruplicator is used. To reduce the heat dissipation and the charge/discharge time of the capacitors, a high oscillating frequency is used; it consequently enhances the output efficiency of the converter itself. This allows to obtain a negative voltage of 20 Volt approximately. It will be subsequently stabilized by a dedicated stabilizer, of the series type, whose output voltage, however, will be adjusted by the R20 trimmer at the voltage required by the LCD.

This part also includes a digital switch, consisting of the FLIP_FLOP U8A and the transistor Q3, which allows the user to turn on/off the device, by the switch SW33, without necessarily connecting or disconnecting it to/from the power supply source.

To work properly, the apparatus requires a DC input voltage between 12 and 18Volt.

PROGRAM

N.B. The italics are external comments to the program operation; therefore they do not execute any instructions nor are they an integral part.

```
;*****  TREDICINO  REV  1.0  ******

;These assembler instructions allow to define the program priority parameters, which will be used by the compiler for the writing out of the program in machine code (HEX codes)
        ;DIRECTIVE

        .title " tredicino 1.0 "
        .vers  " ST6215 "
        .romsize 2
        .w_on
        .pp_on
        .input " 6215_reg.asm "
;data register


;********* data RAM **********.
```

The status indicate the program if predetermined events, situations, functions, must or must not be considered in the program decisional choices.
This variables arrangement makes the program structure more articulate, thus causing it to be even more versatile and synthetic.
The parameter assignment is performed by operating on each single BIT of the two RAM memory BYTES so destined.

```
status   .def 084h ,M          ;stato programma
;---------------------
b_msb        .equ 0
n_inc        .equ I
s_addr       .equ 2
w_r          .equ 3
p_or_m       .equ 4
```

```
change_v        .equ 5
col_t           .equ 6
ram_loc         .equ 7


status_b   .def 085h,        ;stato
programma
;---------------------
n_var           .equ 0
t_del_1         .equ 1
t_del_2         .equ 2
t_del_3         .equ 3
t_del_4         .equ 4
no_del_w        .equ 5
last_r          .equ 6

 ;******* REGISTERS **********
```

The registers used by the program are defined with the assembler command ".DEF"

```
 code           .def 086h,M
n_car_r         .def 087h,M
pos_l_3         .def 088h
v_delay         .def 089h
v_save          .def 08ah
xx              .def 08bh
reg_v           .def 08ch
reg_n_pag       .def 08dh
reg_pag         .def 08eh
reg_l_ram       .def 08fh

result          .def 093h

  ;****** CONSTANTS *********

tmz             .equ 7
eti             .equ 6
dout            .equ 4

   ;********** ROM ***********

      .section 1
      .org 80h
```

Data ROM memory is defined through the definition of BYTES representing the display character codes.

```
prt_res .byte 042H          ;B   40
        .byte 065H          ;e   1
        .byte 06eH          ;n   2
        .byte 076H          ;v   3
        .byte 065H          ;e   4
        .byte 06eH          ;n   5
        .byte 075H          ;u   6
        .byte 074H          ;t   7
        .byte 069H          ;i   8
        .byte 069h          ;i   9
        .byte 06eh          ;n   a
        .byte 03ah          ;    b
        .byte 02eh          ;    c

        .byte 02eh          ;    d
        .byte 02eh          ;    e
        .byte 022h          ;"   f
        .byte 054h          ;T   50
        .byte 052h          ;R   1
        .byte 045h          ;E   2
        .byte 044h          ;D   3
        .byte 049h          ;I   4
        .byte 043h          ;C   5
        .byte 049h          ;I   6
        .byte 04eh          ;N   7
        .byte 04fh          ;O   8
        .byte 022h          ;"   9
        .byte 02dH          ;-   a
        .byte 0a0H          ;s   b
        .byte 0a0H          ;s   c
        .byte 0a0H          ;s   d
        .byte 0a0H          ;s   e
        .byte 02dH          ;-   f
        .byte 02dH          ;-   60
        .byte 02dh          ;-   1
        .byte 030h          ;0   2
        .byte 020h          ;s   3
        .byte 030h          ;0   4
        .byte 020h          ;s   5
        .byte 07ch          ;!   6
        .byte 058h          ;X   7
        .byte 07ch          ;!   8
        .byte 020h          ;s   9
        .byte 020h          ;s   a
        .byte 07ch          ;!   b
        .byte 020h          ;s   c
        .byte 020h          ;s   d
```

```
        .byte 020h      ;s    e
        .byte 07ch      ;!    f
        .byte 02dh      ;-    70
        .byte 02dh      ;-    1
        .byte 07ch      ;!    2
        .byte 020h      ;s    3
        .byte 020h      ;s    4
        .byte 07ch      ;!    5
        .byte 02dh      ;-    6
        .byte 02dh      ;-    7

    ;********** RESET **********
```

initialization cycle of the device, where all the operations that (a) reset and configure the MPU, (b) reset and define the display functioning are carried out..

```
reset   ldi wdr,0feh
        ldi dra,00000000b
        ldi ddra,000h
        ldi ora,000h
        ldi drb,0ffh
        ldi ddrb,0ffh
        ldi orb,0ffh
        ldi drc,0ffh
        ldi          ddrc,11110000b
        ldi orc,0f0h
        ldi adcr,000h
        ldi psc,000h
        ldi tcr,000h
        ldi tscr,00101111b
        ldi drwr,prt_res.w
        ldi ior,00000000b
        call res_ram
        set no_del_w,status_b
        call res_dis
        reti

    ;*** MAIN PROGRAM ***
```

Defines the display graphic pages and, when ended, waits for an external INTERRUPT request.

```
        call pag_0
        call pag_1
        set t_del_1,status_b
        call bip
        res t_del_1,status_b
        set t_del_4,status_b
        call delay
        res t_del_4,status_b
        call pag_0
        call pag_2
        call score
        call arow_key
        ldi ior,010h
wait_1  ldi wdr,0feh
        jp wait_1

    ; ********* ROUTINES
          **********
```

;RAM resetting routine from the location 084h to the 0bfh.

```
res_ram nop
        ldi x,084h
inram   ldi wdr,0feh
        clr a
        ld (x),a
        inc x
        ld a,x
        cpi a,0c0h
        jrc inram
        ret
```

; routine for the display definition and function mode resetting.
Display mode: 8 x 8 pixel characters
30 row characters
128 vertical dots

```
res_dis ldi drc,00000000b
        ldi a,08ch
nop1    dec a
        ldi wdr,0feh
        jrnz nop1
        ldi drc,10000000b
        ldi code,000h
        ldi x,00110000b
        call w_r_c_d
        ldi code,01h
        ldi              x,01110111b
```

```
                call w_r_c_d
                ldi code,02h
                ldi             x,00011101b
                call w_r_c_d
                ldi code,03h
                ldi             x,00111111b
                call w_r_c_d
                ldi code,04h
                ldi             x,00000111b
                call w_r_c_d
                ldi code,08h
                ldi x,000h
                call w_r_c_d
                ldi code,09h
                ldi x,000h
                call w_r_c_d
                ldi code,00001010b
                ldi             x,00000000b
                call w_r_c_d
                ldi code,00001011b
                ldi x,00000000b
                call w_r_c_d
                ret

;Timing routine.
;The waiting time is determined by the status t bit of the xx at logic level 1.

delay           jrs t_del_4,status_b,delay_4
                jrs t_del_3,status_b,delay_3
                jrs t_del_2,status_b,delay_2
                jrs t_del_1,status_b,delay_1
                jp end_del
delay_4         ldi v_delay,0FFh
                jp fv_k
delay_3         ldi v_delay,07Fh
                jp fv_k
delay_2         ldi v_delay,03Fh
                jp fv_k
delay_1         ldi v_delay,01Fh
fv_k            ldi a,0FFh
fv_m            dec a
                ldi wdr,0feh
                jrnz fv_m
                dec v_delay
                jrnz fv_k
end_del         ret
```

```
; routine for the writing and reading of operating codes, to and from the display.

w_r_c_d         jrs
no_del_w,status_b,no_del_c
                call delay
no_del_c        call busy_f
                ldi drc,10010000b
                ldi drc,11010000b
                ld a,code
                ld drb,a
                ldi drc,10010000b
                jrs w_r,status,read
                ldi drc,11000000b
                ld a,x
                ld drb,a
                ldi drc,10000000b
end_w_r         ret

read            ldi wdr,0feh
                ldi drb,0ffh
                ldi ddrb,000h
                ldi orb,000h
                ldi drc,10100000b
                ldi drc,11100000b
                nop
                ld a,drb
                nop
                ldi drc,10100000b
                ldi drc,10000000b
                ldi drb,0ffh
                ldi ddrb,0ffh
                ldi orb,0ffh
                jp end_w_r

;Waiting routine for the display operation ending.

busy_f          ldi wdr,0feh
                ldi drb,0ffh
                ldi ddrb,000h
                ldi orb,000h
                ldi drc,10110000b
                ldi drc,11110000b
                nop
                ld a,drb
                nop
```

```
                ldi drc,10110000b
                ldi drc,10000000b
                ldi drb,0ffh
                ldi ddrb,0ffh
                ldi orb,0ffh
                jrs 7,a,busy_f
                ldi wdr,0feh
                ret

; display clearing routine.

pag_0           ldi xx,02h
                clr v
                clr w
                call set_addr
half            ldi n_car_r,0f0h
                ldi x,020h
                set no_del_w,status_b
cvb_m           call wr_cart
                dec n_car_r
                jrnz cvb_m
                dec xx
                jrz ret_r
                jp half
ret_r           ret

;Displaying message writing routine:
"BENVENUTI IN TREDICINO".

pag_1           ldi w,0ah
                ldi v,04h
                ldi y,prt_res.d
                ldi n_car_r,09h
                call pl
                ldi w,0ah
                ldi v,07h
                ldi n_car_r,06h
                call pl
                ldi w,09h
                ldi v,0bh
                ldi n_car_r,0bh
                res no_del_w,status_b
                set t_del_4,status_b
                 call delay
                call delay
                res t_del_4,status_b
                set t_del_1,status_b
                call pl
                res t_del_1,status_b
                ret

;Initial mask displaying routine.

pag_2           ldi v,01h
first_r         ldi w,0ch
                ldi n_car_r,08h
                call pl
                jrr last_r,status,next_r
                res last_r,status
exit_r          ret
next_r          ldi v,02h
inc_row         ldi w,00h
                ldi n_car_r,01h
                call pl
                set n_inc,status
                set s_addr,status
                ldi n_car_r,07h
                call pl
                res n_inc,status
                ldi n_car_r,0dh
                call pl
                res s_addr,status
                inc v
                ld a,v
                cpi a,0fh
                jrnc inc_r
                ldi y,062h
                jp inc_row
inc_r           set last_r,status
                ldi y,05ah
                jp first_r

;Score mask displaying routine.

score           ldi v,05h
first_s         ldi w,01bh
                ldi y,070h
                ldi n_car_r,02h
                call pl
                jrr last_r,status,next_s
                res last_r,status
exit_s          ret
next_s          ldi v,06h
inc_r_s         ldi w,01ah
                ldi n_car_r,04h
                call pl
```

```
                inc v
                ld a,v
                cpi a,0dh
                jrnc i_l_l_s
                ldi y,072h
                jp inc_r_s
i_l_l_s         set last_r,status
                jp first_s

;Cursor pointers displaying routine.

arow_key        ldi w,0fh
                ldi v,01h
                call set_addr
                ldi x,07eh
                call wr_cart
                ldi w,019h
                ldi v,06h
                call set_addr
                ldi x,07eh
                call wr_cart
                ldi reg_v,01h
                ldi reg_l_ram,090h
                set change_v,status
                ret

;INTERRUPT routine for the pressed key recognition and the related function execution.

ack_tas         ld a,dra
                ld n_car_r,a
                ldi code,00h
                ldi x,00110000b
                call w_r_c_d
                ld a,n_car_r
                set no_del_w,status_b
                set t_del_2,status_b
                andi a,00011111b
                ld y,a
                cpi a,01ah
                jrnc t_f
                addi a,02h
                cpi a,0fh
                jrc inc_col0
                jp inc_col9
t_f             jp tas_fun

;keys related to the teams.

inc_col0        ldi w,00h
                res col_t,status
                jp d_inc
inc_col9        ldi w,09h
                subi a,0dh
                set col_t,status

d_inc           ld v,a
                jrs
change_v,status,no_bip_1
                call bip
no_bip_1        call set_addr
                call rd_cart
                jrs p_or_m,status,m_col
                ld a,n_car_r
                cpi a,039h
                jrc inc_col
                call bip
                jp end_t_a
inc_col         inc n_car_r
                jp wr_r_c
m_col           ld a,n_car_r
                cpi a,030h
                jrnz dec_col
                call bip
                res p_or_m,status
                jp end_t_a
dec_col         dec n_car_r
                res p_or_m,status
wr_r_c          call set_addr
                ld a,n_car_r
                ld x,a
                call wr_cart

; score comparison.

comp_res        jrr col_t,status,cpi_c0
                ldi w,00h
                jp cpi_r
cpi_c0          ldi w,09h
cpi_r           ld a,x
                ld y,a
                call set_addr
                call rd_cart
                ld a,y
                cp a,n_car_r
```

```
            jrnz no_x
            jp prt_x
no_x        ldi n_car_r,031h
            jrc wr_2_1
            jrr col_t,status,prt_cart
            inc n_car_r
            jp prt_cart
wr_2_1      jrs col_t,status,prt_cart
            inc n_car_r
            jp prt_cart
prt_x       ldi n_car_r,058h
prt_cart    ldi w,0ch
            call set_addr
            ld a,n_car_r
            ld x,a
            call wr_cart
            jp end_t_a

; score comparison.

tas_fun     subi a,01ah
            jrnz no_mind
            jrr p_or_m,status,set_mind
            res p_or_m,status
            jp end_t_a

; Minus function

set_mind    set p_or_m,status
            jp end_t_a

; other functions.

no_mind     cpi a,01h
            jrz inc_dec
            jp no_tsf_1

inc_dec     res change_v,status
            ldi w,0fh
            ld a,reg_v
            ld v,a
            call set_addr
            ldi x,020h
            call wr_cart
            jrs p_or_m,status,mind_v
            inc reg_v
            ld a,reg_v
            ld v,a
            cpi a,0fh
            jrnc inc_p
            jp set_a
inc_p       ldi reg_v,01h
            set change_v,status
set_a       ld a,reg_v
            ld v,a
            call set_addr
            ldi x,07eh
            call wr_cart
            jp end_t_a
mind_v      ld a,reg_v
            cpi a,01h
            jrz pos_h
            dec reg_v
            ld a,reg_v
            cpi a,01h
            jrnz n_change
            set change_v,status
n_change    ld v,a
            res p_or_m,status
            jp set_a

;increment/decrement key of the
variables set row.


pos_h       ldi v,0eh
            ldi reg_v,0eh
            res p_or_m,status
            jp set_a

no_tsf_1    subi a,02h
            cpi a,03h
            jrc tsf_2_4
            jp no_tfs_4

;Variables selection keys.

tsf_2_4     cpi a,01h
            jrz t_a_a
            jrc t_b_b
            ldi w,013h
            jp code_v
t_a_a       ldi w,012h
            jp code_v
t_b_b       ldi w,011h
```

```
code_v      jrr change_v,status,mnb
            call bip
            jp end_t_a
mnb         ld a,reg_v
            ld v,a
            call set_addr
            call rd_cart
            ld a,n_car_r
            cpi a,020h
            jrz car_r_sp
            cpi a,031h
            jrz car_r_1
            cpi a,032h
            jrz car_r_2
            jp car_r_x

car_r_sp    ldi n_car_r,031h
            jp wr_v
car_r_1     ldi n_car_r,058h
            jp wr_v
car_r_2     ldi n_car_r,020h
            jp wr_v
car_r_x     ldi n_car_r,032h
            jp wr_v

wr_v        ld a,reg_v
            ld v,a
            call set_addr
            ld a,n_car_r
            ld x,a
            call wr_cart
            jp end_t_a

; page change key.

no_tfs_4    call bip
            jp end_t_a
end_t_a     res t_del_2,status_b
            call arow_p
            call wr_result
            reti

; displaying of the pointers correlated
with the favourable prediction.

arow_p      clr result
            ldi v,02h
again       ldi w,0ch
            call set_addr
            call rd_cart
            ld a,n_car_r
            ld y,a
            ldi w,011h
            call set_addr
            call rd_cart
            ld a,y
            cp a,n_car_r
            jrnz no_p_1
            jp wr_point
no_p_1      inc w
            call set_addr
            call rd_cart
            ld a,y
            cp a,n_car_r
            jrnz no_p_2
            jp wr_point
no_p_2      inc w
            call set_addr
            call rd_cart
            ld a,y
            cp a,n_car_r
            jrnz no_p_3
            jp wr_point
no_p_3      ldi w,015h
            call set_addr
            ldi x,020h
            call wr_cart
            inc v
            ld a,v
            cpi a,0fh
            jrnc exit_p
            jp again
exit_p      ret
wr_point    ldi w,015h
            call set_addr
            ldi x,07fh
            call wr_cart
            inc result
            inc v
            ld a,v
            cpi a,0fh
            jrnc exit_pp
            jp again
exit_pp     jp exit_p

; total score displaying.
```

```
wr_result   ldi v,06h
            ld a,result
            cpi a,0ah
            jrnc no_1_1
            jp no_1
no_1_1      subi a,0ah
            ld result,a
            ldi w,01bh
            call set_addr
            ldi x,031h
            call wr_cart
            jp yes_1
no_1        ldi w,01bh
            call set_addr
            ldi x,020h
            call wr_cart
yes_1       ldi w,01ch
            call set_addr
            ld a,result
            addi a,030h
            ld x,a
            call wr_cart
            ret
;Memorization routine of the data inserted in the systems, if any.

wr_ram      nop
            ret

;Reading routines of the memorized data inserted in the systems, if any.

rd_ram      nop
            ret

;Sound signal routine.

bip         set tmz,tscr
            set dout,tscr
            ldi tcr,000h
            call delay
            res dout,tscr
            ldi tcr,000h
            ret

;Displaying writing routine of the BYTE data memorized by the ROM.

pl          jrs s_addr,status,pl1
            call set_addr
pl1         call inc_v
            jrnz pl1
            ret

; displaying routine to set the cursor co-ordinates definition.

set_addr    call k_x_row
            ld x,a
            ldi code,00001010b
            call w_r_c_d
            jrr b_msb,status,no_msb
            ldi x,00000001b
            jp wr_msb
no_msb      ldi x,00000000b
wr_msb      ldi code,00001011b
            call w_r_c_d
            ret

;ROM pointer increment routine.

inc_v       ld a,(y)
            ld x,a
            call wr_cart
            jrs n_inc,status,inc_off
            inc y
inc_off     dec n_car_r
            ret

;Co-ordinates multiplication routine.

k_x_row     res b_msb,status
            ld a,v
            ld v_save,a
            jrnz mul
            jp end_mulèmul          addi
a,01dh
            jrnc k_x_row2
            set b_msb,status
k_x_row2    dec v_save
            jrnz mul
end_mul     add a,w
            jrnc exit_m
            set b_msb,status
exit_m      ret
```

```
; character writing displaying routine

wr_cart     ldi wdr,0feh
            ldi code,00001100b
            call w_r_c_d
            ret

; character reading displaying routine.

rd_cart     ldi wdr,0feh
            set w_r,status
            ldi code,00001101b
            call w_r_c_d
            ldi code,00001101b
            call w_r_c_d
            res w_r,status
            ld n_car_r,a
            ret

 ;@@ INTERRUPT VECTORS @@
            .section 32
            .org 00h

int4        nop
            reti
int3        nop
            reti
int2        nop
            reti
int1        nop
            reti

            .org 0ch

; Interrupt Vectors.

nmi         jp ack_tas
res                 jp reset
```

The software was realized with:

ST6 STARTER KIT
ST6210/15
SGS-THOMSON Microelectronics

In which it's possible to find:

ASSEMBLER Program
LINKER Program
Simulation software program
Programming of the integrated circuit program
Instruction manuals
Data Book
ST6xxx programmer

LIST OF COMPONENTS

	Quantity	Reference	Component Part
1	1	BZ1	buzzer piezoelectric
2	2	C1,C2	22pF
3	9	C3,C4,C7,C8,C13,C14,C21,C22,C26	100nF
4	7	C5,C6,C16,C19,C20,C25,C27	100mF
5	3	C9,C11,C23	10mF
6	3	C10,C15,C24	1mF
7	1	C12	470mF
8	1	C17	10nF
9	1	C18	470pF
10	12	D1,D2,D3,D4,D5,D7,D8,D9,D11,D12 D13,D14	1N4148
11	1	D6	led
12	2	D10,D15	1N4004
13	1	J1	dc plug
14	1	JP1	20 poles connector
15	4	JP2,JP3,JP5,JP6	14 poles connector
16	1	JP4	jumper
17	2	Q1,Q2	BC848
18	1	Q3	BD676A
19	3	R1,R15,R18	10KOhm
20	2	R2,R20	10KOhm trimmer
21	3	R3,R11,R16	1KOhm
22	2	R4,R10	2,2KOhm
23	5	R5,R6,R7,R8,R14	100KOhm
24	1	R9	680 Ohm
25	2	R12,R13	47 KOhm
26	1	R17	2,2 KOhm, 1/4 Watt
27	1	R19	5 KOhm
28	34	SW1,SW2,SW3,SW4,SW5,SW6,SW7, SW8,SW9,SW10,SW11,SW12,SW13, SW14,SW15,SW16,SW17,SW18,SW19, SW20,SW21,SW22,SW23,SW24,SW25, SW26,SW27,SW28,SW29,SW30,SW31, SW32,SW33,SW34	n. a. switch
29	1	U1	ST62E15
30	1	U2	74C922
31	1	U3	CD4012
32	1	U4	LM78L08
33	1	U5	LM7805
34	1	U6	NE555
35	1	U7	LM79L05
36	1	U8	CD4013
37	1	X1	Quartz 8 MHZ

displaying part LCD LM238XB Hitachi inc.

Claims

1. Electronic system for the control and elaboration in real time of the data typical of "Totocalcio" traditional system predictions.

a) a program which memorises, associates, processes and displays the data inserted by the user in many times. The program, as far as associative functions are concerned, is specifically dedicated to the data distributive logic configured by the adopted graphic "mask".

In its first part such mask reproduces the associative numeric sequence of the football teams, as it was presented in the weekly calendar of "Totocalcio" game.

In a second part the partial and then final scores of the corresponding matches are reproduced, employing the symbols commonly used to indicate the match result (1,2,X). In the third part the predictions introduced by the user referring to the single day are reported, up to a maximum of four hypothetical issues. Besides, the favourable data of the comparison between scores and predictions are displayed in real time. In the fourth and last part the favourable events sum (1 to 13) relating to the systems is indicated;

b) an electronic device consisting of a 33 function keys keyboard, an LCD display (interface machine-user) and an electronic control and management electronic board based on a component part working with programmable digital logic according to a). From a functional point of view, every element of this device is strictly necessary to the mask displaying and to the data input according to the distributive logic of the same.

SUB-CLAIMS

a. same system as sub 1, a), with the possibility of employing elaboration data devices of higher capacity to introduce data corresponding to more than four hypothetical score-issues;

b. same system as sub 1, as for sub-claim a., with the possibility to perform complex statistical functions;

c. same system as sub 1, with the possibility of a battery power supply or anyway without direct power supply from the mains;

d. same system as sub 1, with the possibility to employ any displaying system;

e. same system as sub 1, with the possibility to use any system based on a microprocessor or similar devices (i.e. micro-controllers);

f. same system as sub 1, with the possibility to employ semi-custom integrated circuits;

g. same system as sub 1, with the possibility to employ custom integrated circuits;

h. same system as sub 1, with the possibility to employ integrated circuits that implement any logical function whatsoever;

i. same system as sub 1, with the possibility to employ any memorization device.

European Patent Office

PARTIAL EUROPEAN SEARCH REPORT

which under Rule 45 of the European Patent Convention shall be considered, for the purposes of subsequent proceedings, as the European search report

Application Number EP 93 11 7069

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.5)
X	GB-A-2 150 331 (M.RINALDI, IT) 26 June 1985 * the whole document *	1	G06F15/44
X	GB-A-2 092 342 (DUTCHFORD LIMITED, UK) 11 August 1982 * the whole document *	1	
X	US-A-4 141 548 (I.M.EVERTON, US) 27 February 1979 * the whole document *	1	

TECHNICAL FIELDS SEARCHED (Int.Cl.5)

G06F

INCOMPLETE SEARCH

The Search Division considers that the present European patent application does not comply with the provisions of the European Patent Convention to such an extent that it is not possible to carry out a meaningful search into the state of the art on the basis of some of the claims

Claims searched completely :

Claims searched incompletely :

Claims not searched :

Reason for the limitation of the search:

program for computers

Place of search	Date of completion of the search	Examiner
THE HAGUE	28 December 1994	Barba, M

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C07)