```
1 REM *******************************************
2 REM *  HERE FIRST THE SOFT-            *
3 REM *  SWITCHES / PEEKS AND            *
4 REM *  POKES USED WITH THE             *
5 REM *  GAMEPORT – THIS BLOCK           *
6 REM *  MAY BE REMOVED TO KEEP          *
7 REM * PROGRAMCODE COMPACT              *
8 REM *      R = READ       W= WRITE     *
9 REM *      R/W = READ AND WRITE        *
10 REM * ALL ADRESSES ARE MARKED *
11 REM * AS HEXADECIMAL ADRESSES *
12 REM * $C040 = UTILITY STROBE   R    *
13 REM * $C058 =  AN0 = LOW         R/W *
14 REM * $C059 =  AN0 = HIGH        R/W *
15 REM * $C05A =  AN1 = LOW         R/W *
16 REM * $C05B =  AN1 = HIGH        R/W *
17 REM * $C05C =  AN2 = LOW         R/W *
18 REM * $C05D =  AN2 = HIGH        R/W *
19 REM * $C05E =  AN3 = LOW         R/W *
20 REM * $C05F =  AN3 = HIGH        R/W *
21 REM ******************************************
22 REM * HERE ARE THE PUSHBUTTON*
23 REM * ADRESSES WHICH WE WILL    *
24 REM * USE AS INPUT DETECTORS    *
25 REM ******************************************
26 REM * $C061 = PB0                    R *
27 REM * $C062 = PB1                    R *
28 REM * $C063 = PB2                    R *
29 REM * $C064 = PADDLE 1 - X AXIS  R *
30 REM * $C065 = PADDLE 1 - Y AXIS  R *
31 REM * $C066 = PADDLE 2 - X AXIS  R *
32 REM * $C067 = PADDLE 2 - Y AXIS  R *
33 REM ******************************************
34 REM * HERE ARE THE ADRESSES      *
35 REM * LISTED ALSO AS DECIMALS     *
36 REM * STROBE% = 49216        $C040   *
37 REM * AN0% = 49240 = LOW     $C058   *
38 REM * AN0% = 49241 = HIGH    $C059   *
39 REM * AN1% = 49242 = LOW     $C05A  *
40 REM * AN1% = 49243 = HIGH    $C05B  *
41 REM * AN2% = 49244 = LOW     $C05C  *
42 REM * AN2% = 49245 = HIGH    $C05D  *
43 REM * AN3% = 49246 = LOW     $C05E  *
44 REM * AN3% = 49247 = HIGH    $C05F  *
45 REM ******************************************
46 REM *    PB0%  =  49249 =  $C061         *
47 REM *    PB1%  =  49250 =  $C062         *
48 REM *    PB2%  =  49251 =  $C063         *
49 REM *    P1X%  =  49252 =  $C064        *
50 REM *    P1Y%  =  49253 =  $C065        *
51 REM *    P2X%  =  49254 =  $C066        *
52 REM *    P2Y%  =  49255 =  $C067        *
53 REM ********************************************
54 REM ********************************************
55 REM * DEFINE VARIABLES                    *
56 REM ********************************************
57 LET STROBE% = 49216
58 ***************************************************
59 LET AN0OFF% = 49240
60 LET AN0ON% = 49241
61 LET AN1OFF% = 49242
62 LET AN1ON% = 49243
63 LET AN2OFF% = 49244
64 LET AN2ON% = 49245
65 LET AN3OFF% = 49246
66 LET AN3ON% = 49247
67 REM *******************************************
68 LET PB0% = 49249
69 LET PB1% = 49250
70 LET PB2% = 49251
71 REM *******************************************
72 LET P1X% = 49252
73 LET P1Y% = 49253
74 LET P2X% = 49254
75 LET P2Y% = 49255
76 REM *******************************************
77 REM *******************************************
78 REM *  Lets first instruct the user about    *
79 REM *  the input-options on the screen      *
80 REM *******************************************
81 PRINT " TYP IN A NUMBER FROM 0 TO
16 AND HIT THEN ENTER"
82 PRINT " 0  QUITS THE PROGRAM"
82 PRINT " 1  SWITCHES RELAY 1 ON =
LIGHTBULB RED "
83 PRINT " 2 SWITCHES RELAY 2 ON =
LIGHTBULB ORANGE "
84 PRINT " 3 SWITCHES RELAY 4 ON =
LIGHTBULB YELLOW "
85 PRINT " 4 SWITCHES RELAY 5 ON =
LIGHTBULB GREEN "
86 PRINT " 5 SWITCHES RELAY 6 ON =
LIGHTBULB VIOLETT "
87 PRINT " 7 SWITCHES RELAY 7 ON =
LIGHTBULB BLUE "
88 PRINT " 8 SWITCHES RELAY 8 ON =
STROBOSKOPE "
89 PRINT " 9 SWITCHES RELAY 9 ON =
BASSHORN "
90 PRINT " 10 SWITCHES RELAY 10 ON =
MIDHORN "
91 PRINT " 11 SWITCHES RELAY 11 ON =
TWEETHORN "
92 PRINT " 12 SWITCHES RELAY 12 ON =
SIRENE "
93 PRINT " 13 SWITCHES RELAY 13 ON =
POWERCORD CAMERA "
94 PRINT " 14 SWITCHES RELAY 14 ON =
RADIO "
95 PRINT " 15 SWITCHES RELAY 15 ON =
CASSETEPLAYER BARKING DOG "
96 PRINT " 16 SWITCHES RELAY 16 ON =
POWERCORD ELECTRIC MAGNET"
97 PRINT " PLEASE CHOOSE DEVICE "
99 INPUT CHOICE%
100 REM ** SHORT INPUTCHECK **
```

```
101 IF  CHOICE% < 17 THEN GOTO 120
100 IF CHOICE% > 16 THEN PRINT " THE
NUMBER IS TOO LARGE "
109 REM * WRONG NUMBER GO BACK TO
INPUT *
110 GOTO 81
114 REM ** LINE 2000 IS PROGRAM END **
115 IF CHOICE% = 0 GOTO 2000
118 REM *INPUT OK GO AHEAD TO SIEVE*
119 REM ***  ENTRYPOINT OF SIEVE AND
TURN NUMBER TO HEXVALUE ***
120 CHOICE% = CHOICE% -1
122 REM * LETS PICKUP HIGHEST BIT *
123 IF CHOICE% > 7 THEN VAAN3% = 1
124 IF CHOICE% < 8 THEN VAAS3% = 0
125 IF CHOICE% > 7 THEN CHOICE% =
CHOICE% - 8
126 REM * LETS PICKUP THIRD BIT *
127 IF CHOICE% > 3 THEN VAAN2% = 1
128 IF CHOICE% < 4 THEN VAAN2% = 0
129 IF CHOICE% > 3 THEN CHOICE% =
CHOICE% - 4
130 REM * LETS GET SECOND BIT *
131 IF CHOICE% >1 THEN VAAN1% = 1
132 IF CHOICE% < 2 THEN VAAN1% = 0
133 IF CHOICE% > 1 THEN CHOICE% =
CHOICE% - 2
134 REM * NOW PICKUP LAST SMALLEST
BIT *
135 IF CHOICE% > 0 THEN VAAN0% = 1
136 IF CHOICE% < 1 THEN CHOICE% = 0
136 IF CHOICE% >0 THEN CHOICE% =
CHOICE% - 1
137 REM * SO NOW CHOICE% IS 0 TILL
NEXT INPUT *
138 REM * BITS ARE NOW SET TO 1 OR 0 *
139 REM * HERE A JUMP CAN BE DONE IF
NEEDED *
150 IF VAAN3% = 0 THEN POKE 49246,0
151 IF VAAN3% = 1 THEN POKE 49247,1
152 IF VAAN2% = 0 THEN POKE 49244,0
153 IF VAAN2% = 1 THEN POKE 49245,1
154 IF VAAN1% = 0 THEN POKE 49242,0
155 IF VAAN1% = 1 THEN POKE 49243,1
156 IF VAAN0% = 0 THEN POKE 49240,0
157 IF VAAN0% = 1 THEN POKE 49241,1
158 REM *** NEXT SHOOTOUT STROBE ***
159 A = PEEK(49216)
160 HOME
161 PRINT " THE DEMANDED ITEM HAS
BEEN SWITCHED ON."
162 PRINT " HIT ANY KEY TO CONTINUE"
163 PAUSE
164 GOTO 1900
1900 HOME
1901 GOTO 80
2000 END
```

*** *lines printed italic will be changed* ****

****
* UP TO THIS POINT THE   *
* PREVIOUS PROGRAM IS   *
* REPEATED JUST FOR   *
* REVISION TO SEE WHAT   *
* WILL BE NEEDED IN A   *
* NEW ALTERED PROGRAM *
****

Line 1 to 76 only contains the commented definitions of the used adresses to control the inputs and outputs of the boards – they remain the same also in a new peogram and therefore it would be usefull to keep this lines in a new program too for documentation-purposes….

The lines 77 to 115 contain a user-menu and a simple input-routine to execute immediately a desired action and switch on a desired device. This part of the program (the menu) might still be usefull and the input should just be modified slightly for altered use….

The lines from 118 to 159 contain the sieve to determine the choice (i.e. decision taken) from the menu and translating that choice with a sieve into 4 Bits that will be set on the ports of the annunciators and decoded within the 74LS154 and thereafter these BITS are setup and handed with a "strobe"-signal to the 74LS154 and that chip sets up the desired and decoded output line. This part of the program will also be used in nearly all similar programs that make use of one of the PCBs without bothering if it's the Base PCB with the relays or if it's the "Piggy-Pack-PCB" that switches the output with a solid-state-relay. But up to now this part of the program does not use any of the pushbutton connectors **or** any input from the variable resistors that emulate a joystick - **as an input** .

The lines 160 to 164 should be removed because in the new program we need this lines to extend the previous part of the program and expand it with routines to make use of the additional inputs.

The only point important in the rest of the coding is the line 2000 which is related to the line 115 and is up till now used as exit. If this line is relocated to a line with another number then the reference in line 115 - MUST BE ALTERED TOO !
****

So first of all lets set the exit-point of the program to a point far above so that changes in the future may be made without causing this part of the program to be altered any more:

114 REM ** LINE 9999 IS PROGRAM END **
115 IF CHOICE% = 0 THEN GOTO 9999

And deleting the old line 2000 of coure also needs the inserting of the new exit:

9999 END

Assuming that the sieve of the program will also be used in this program we should start to make changes to other parts of the program and instead of leaving the program in the old sequential structure the changes should split the parts to modular structure that can be used with a "GOSUB (program-line-number)"-command and the related modules should be closed with a "RETURN"- command.

This will make sure that the main-program in the future will continue with the next line of the program that continues after the "GOSUB".

One point should always be taken care of:
YOU SHOULD NEVER USE THE GOSUB COMMAND IN A MODULE THAT OFFERS WITHIN ITSELF OTHER EXIT-POINTS !

First of all such a bad habit forces the use of the POP-command - but much more dangerous is the fact, that programs with such bad habit become difficult to trace and more difficult to alter !

Next changes affect the part of the program, that is used for information of the user and for input of the choice of a device determined to be switched. The line 80 shall be used as entry-point of the routine and it should be altered to make sure that each time you enter the routine here the display is set to the same condition so this line should be altered to:

80 HOME

and the end of the routine as module should be set to the new line 115 :

115 RETURN

With the data of the chosen device to be stored in the variable CHOICE% .

Of course also the lines 101 to110 must be altered.

101 IF CHOICE% < 17 THEN GOTO 120
should be altered to :
101 IF CHOICE% < 17 THEN RETURN

This will be besides line 115 a "legal exit" to bring you back to the "calling-point" where you came from with the GOSUB-command.

100 IF CHOICE% > 16 THEN PRINT " THE NUMBER IS TOO LARGE ! Hit any Key to get back for new choice…. "
*109 REM * WRONG NUMBER GO BACK TO INPUT **

This routine may stay as check for the input to make sure that the entered number is within the permitted choice and otherwise turns you back to the "point of choice".

But the line 110 which is to be executed if the entered number is "illegal" must be altered to:

110 PAUSE
113 GOTO 80

Similar changes must be made to the Program-part from line 118 to 159 , the "sieve" to determine the BITs of the variable CHOICE% and the Routine to pass the 4 BITs over to the 74LS154 and give it the strobe-signal to start the decoding of the active line.

The call of the routine will start with a "GOSUB 118 –command and the routine should turn back to the calling program with the line
160 HOME
changed to the new code:
160 RETURN

The lines from 161 to 2000 may now be removed and the expansion of the program may start from this point.

The last point to decide at the moment, is the entry-point of the main-program because the program of course should not execute subroutines before they are needed. To leave enough space for other subroutines lets set the start of the main-program to line 6000:
76 REM **
should be altered to:
76 GOTO 6000
6000 REM ** HERE MAIN-PROGRAM STARTS **

Up till now the current program is coded till line 160 and with a main-program starting at 6000 we´ll now add the subroutines for the pushbuttons. We will leave some space and linenumbers for possible modifications. There should be two different routines:
The first one should assume the button to be in a released status und findout if the button has been pressed (i.e. check for a change to “high” status) and the second routine should check for the release of the button (i.e. the button is pressed down and the release of the button marks a change to “low” status):

```
200 REM ** CHECK STATUS PB0 *********
201 REM * PB0STA% = 1 = HIGH          *
202 REM * PB0STA% = 0 = LOW           *
203 REM * TEMPSTA% = INTERIM          *
204 REM * PROOFMODE% = 0 OR 1         *
205 REM * 0 = 1 CYCLE ** 1=LOOP       *
206 REM *********************************
207 REM * ENTRYPOINT FOR CHANGE *
208 REM * FROM PRESS TO RELEASE  *
209 REM *********************************
210 TEMPSTA% = PEEK( 49249)
211 IF TEMPSTA%>127 THEN PB0STA% = 1
212 IF TEMPSTA%<128 THEN PB0STA% = 0
213 IF PROOFMODE% = 0 THEN RETURN
219 IF PB0STA% = 1 THEN GOTO 210
220 IF PB0STA% = 0 THEN RETURN
221 REM **** END OF ROUTINE P2R *******
225 REM ************************************
226 REM * ENTRYPOINT FOR CHANGE      *
227 REM * FROM RELEASED TO PRESS    *
228 REM ************************************
230 TEMPSTA% = PEEK( 49249)
231 IF TEMPSTA%>127 THEN PB0STA% = 1
232 IF TEMPSTA%<128 THEN PB0STA% = 0
233 IF PROOFMODE% = 0 THEN RETURN
234 IF PB0STA% = 0 THEN GOTO 230
235 IF PB0STA% = 1 THEN RETURN
236 REM *** END OF ROUTINE R2P ********
237 REM ***********************************
238 REM *  END OF ROUTINES PB 0        *
239 REM ***********************************

240 REM ** CHECK STATUS PB1 *********
241 REM * PB1STA% = 1 = HIGH          *
242 REM * PB1STA% = 0 = LOW           *
243 REM * TEMPSTA% = INTERIM          *
244 REM * PROOFMODE% = 0 OR 1         *
245 REM * 0 = 1 CYCLE ** 1=LOOP       *
246 REM *********************************
247 REM * ENTRYPOINT FOR CHANGE *
248 REM * FROM PRESS TO RELEASE  *
249 REM *********************************
250 TEMPSTA% = PEEK(49250)
251 IF TEMPSTA%>127 THEN PB1STA% = 1
252 IF TEMPSTA%<128 THEN PB1STA% = 0
253 IF PROOFMODE% = 0 THEN RETURN
259 IF PB1STA% = 1 THEN GOTO 250
260 IF PB1STA% = 0 THEN RETURN
261 REM **** END OF ROUTINE P2R *******
265 REM ************************************
266 REM * ENTRYPOINT FOR CHANGE      *
267 REM * FROM RELEASED TO PRESS    *
268 REM ************************************
270 TEMPSTA% = PEEK( 49250)
271 IF TEMPSTA%>127 THEN PB1STA% = 1
272 IF TEMPSTA%<128 THEN PB1STA% = 0
273 IF PROOFMODE% = 0 THEN RETURN
274 IF PB1STA% = 0 THEN GOTO 270
275 IF PB1STA% = 1 THEN RETURN
276 REM *** END OF ROUTINE R2P ********
277 REM ***********************************
278 REM *  END OF ROUTINES PB 1        *
279 REM ***********************************

280 REM ** CHECK STATUS PB2 *********
281 REM * PB2STA% = 1 = HIGH          *
282 REM * PB2STA% = 0 = LOW           *
283 REM * TEMPSTA% = INTERIM          *
284 REM * PROOFMODE% = 0 OR 1         *
285 REM * 0 = 1 CYCLE ** 1=LOOP       *
286 REM *********************************
287 REM * ENTRYPOINT FOR CHANGE *
288 REM * FROM PRESS TO RELEASE  *
289 REM *********************************
290 TEMPSTA% = PEEK( 49251)
291 IF TEMPSTA%>127 THEN PB2STA% = 1
292 IF TEMPSTA%<128 THEN PB2STA% = 0
293 IF PROOFMODE% = 0 THEN RETURN
299 IF PB2STA% = 1 THEN GOTO 290
300 IF PB2STA% = 0 THEN RETURN
301 REM **** END OF ROUTINE P2R *******
305 REM ************************************
306 REM * ENTRYPOINT FOR CHANGE      *
307 REM * FROM RELEASED TO PRESS    *
308 REM ************************************
310 TEMPSTA% = PEEK( 49251)
311 IF TEMPSTA%>127 THEN PB2STA% = 1
312 IF TEMPSTA%<128 THEN PB2STA% = 0
313 IF PROOFMODE% = 0 THEN RETURN
314 IF PB2STA% = 0 THEN GOTO 310
315 IF PB2STA% = 1 THEN RETURN
316 REM *** END OF ROUTINE R2P ********
317 REM ***********************************
318 REM *  END OF ROUTINES PB 2        *
319 REM ***********************************
```

Check for Pressbutton 3 (PB3) is only needed if the program is used with a IIGS ! If you intend to use this program also with a IIGS, then you should include the following part too to add the

last button (which is only available at the IIGS) within the program.

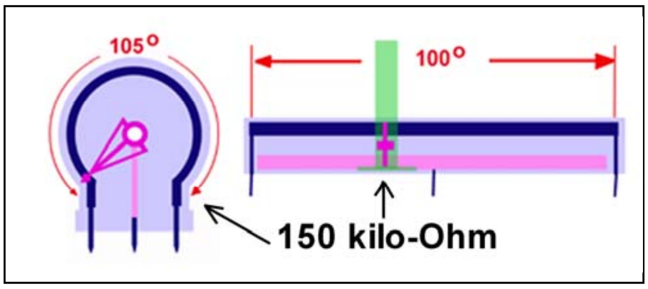
```
320 REM ** CHECK STATUS PB3 *********
321 REM * PB3STA% = 1 = HIGH           *
322 REM * PB3STA% = 0 = LOW            *
323 REM * TEMPSTA% = INTERIM           *
324 REM * PROOFMODE% = 0 OR 1         *
325 REM * 0 = 1 CYCLE ** 1=LOOP        *
326 REM ************************************
327 REM * ENTRYPOINT FOR CHANGE *
328 REM * FROM PRESS TO RELEASE  *
329 REM ************************************
330 TEMPSTA% = PEEK( 49252)
331 IF TEMPSTA%>127 THEN PB3STA% = 1
332 IF TEMPSTA%<128 THEN PB3STA% = 0
333 IF PROOFMODE% = 0 THEN RETURN
339 IF PB3STA% = 1 THEN GOTO 330
340 IF PB3STA% = 0 THEN RETURN
341 REM **** END OF ROUTINE P2R *******
345 REM ****************************************
346 REM * ENTRYPOINT FOR CHANGE      *
347 REM * FROM RELEASED TO PRESS    *
348 REM ****************************************
350 TEMPSTA% = PEEK( 49252)
351 IF TEMPSTA%>127 THEN PB3STA% = 1
352 IF TEMPSTA%<128 THEN PB3STA% = 0
353 IF PROOFMODE% = 0 THEN RETURN
354 IF PB3STA% = 0 THEN GOTO 350
355 IF PB3STA% = 1 THEN RETURN
356 REM *** END OF ROUTINE R2P ********
357 REM ***************************************
358 REM *  END OF ROUTINES PB 3        *
359 REM ***************************************
```

Allthough in the experiments at the moment **the variable resistors** from the „PIGGY-PACK-PCB“ are not used **- its a good idea to include the modules for picking up this values here too...**

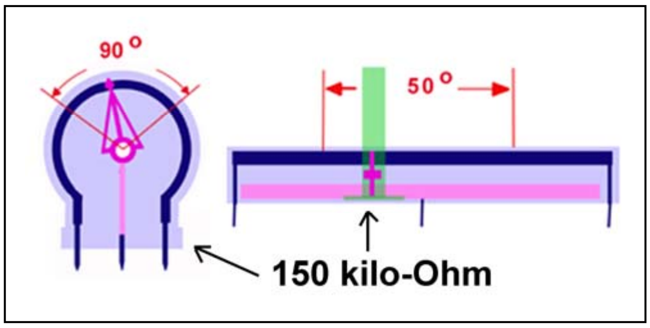
In normal cases it would be practical to only read that value from each resistor once. Just even if you do use the resistor in a loop (for example to draw a continuing line) each value that is taken from the resistors **must be followed up by a routine to put the values up to the screen before you can pickup the next values –** so the “looping” must be located in the program to set there the condition if the loop is to be continued or not (for example if the values are remaining the same for 250 cycles – which would indicate that the resistors are not moved anymore – or if for example a pushbutton has been pressed or not - or if a keystroke has occured…….

But at this point its good too – to take a closer view to the physical “playground” - to be precise **- the limitation within which the resistors are permitted to be used** …. so lets look at the two pictures below:

Resistor on PCB:



Resistor in Joystick :



In the PCB the full Range of the resistors is available – **But this would lead to a disaster and harm the chip on the mainboard!**
First of all it is possible to shortcut between the pickup-contact and the upper or lower contact !

In fact in the real joystick only less than half of the “action-area” is available !

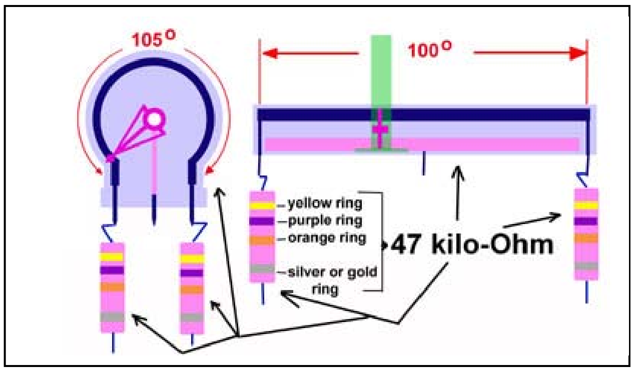
There is always a minimum of at least 50 kiloOhm left between the “pickup-contact” and the both ends of the resistor !

And the chip on the mainboard is determined only to decode the values between 50 kiloOhm an 100 kiloOhm with slight adjustment.

So <u>if you want to use the full range of the resistors</u> used on the PCB <u>the variable resistors must be replaced with 47 kiloOhm variable resistors and at both ends 47 kiloOhm resistors must be added</u> !

Only with such a modification it is permitted to use the full range of the variable resistors !

And now the replacement:



So now let´s change back to the code:

```
500 REM *******************************************
501 REM * START TO PICK VALUES FROM *
502 REM * THE VARIABLE RESISTORS OR *
503 REM * THE JOYSTICK                    *
540 REM *******************************************
541 REM * THIS MODULE CONTAINS THE  *
542 REM * BLOCKS FOR BOTH PADDLES  *
543 REM * OR JOYSTICKS ! FIRST THOSE *
544 REM * FOR DEVICE 1 AND THEN THE  *
545 REM * DEVICE 2 – BOTH IN THE          *
546 REM * X-AXIS FIRST  Y-AXIS SECOND *
547 REM * P1VALX% = VALUE X AXIS        *
548 REM * P1VALY% = VALUE Y AXIS        *
549 REM ******************************************
550 P1VALX% = PEEK( 49252 )
551 P1VALY% = PEEK( 49253 )
552 REM ** CHECK PERMITTED VALUES **
553 IF P1VALX% < 0 THEN GOTO 550
554 IF P1VALY% < 0 THEN GOTO 550
555 IF P1VALX% > 255 THEN GOTO 550
556 IF P1VALY% > 255 THEN GOTO 550
557 RETURN
558 REM ** END OF PICKUP JOYSTICK 1 **
559 REM ******************************************
560 REM * START PICKUP JOYSTICK 2      *
561 REM * P2VALX% = VALUE X AXIS       *
562 REM * P2VALY% = VALUE Y AXIS       *
563 REM ******************************************
570 P2VALX% = PEEK( 49254 )
571 P2VALY% = PEEK( 49255 )
572 REM ** CHECK PERMITTED VALUES **
573 IF P2VALX% < 0 THEN GOTO 570
574 IF P2VALY% < 0 THEN GOTO 570
575 IF P2VALX% > 255 THEN GOTO 570
576 IF P2VALY% > 255 THEN GOTO 570
557 RETURN
558 REM ** END OF PICKUP JOYSTICK 2 **
559 REM ******************************************
```

So up to this point we are now finished with those parts and modules of the program that are responsible for picking up the values from the both PCB´s and that are responsible for setting up the conditions and switches to drive the connected items !

If you just want to make a short program related to that sketch of a security-system according to the previous drawing you can **jump to the next red marking with large letters that indicate the beginning of the coding of the main program**....

Otherwise if you like to workout some additional tools and modules you should stay “on track” with us now and add some other usefull subroutines….

Here we start-up with the coding of the main-program !

```
6000 REM *** MAIN-PROGRAM STARTS ***
```