==========
ksq30.asm

Aby rozpoznać, który klawisz został naciśnięty należy zaprogramować pętlę
obiegającą wyświetlacze/klawisze.

Przez sześć obrotów pętli, zmieniamy kolejno wyświetlacze (POJEDYNCZO)
i dodajemy jeszcze testowanie stanu klawiatury, linijka 53:

```
      JNB   SEQKEY, LoopRunNoKey    ;klawisz odpowiadający masce w R6 nie jest naciśnięty
```

Jeżeli wykryjemy naciśnięcie klawisza (P3.5) odpowiadającego wybranemu wyświetlaczowi
(odpowiednia maska bitowa w R6) to dodajemy odpowiedni bit do agregowanego stanu klawiatury
linijka 56

```
      ORL   AgrKey, A         ;agreguj bit naciśniętego klawisza do AgrKey
```

W zasadzie robimy to przez 7 obrotów pętli - za siódmym razem bez sensu
bo tam nie ma żadnego klawisza, ale wstawianie kodu pomijającego testowanie klawiatury
dla R6=64 to większy wysiłek niż bezsensowne sprawdzenie nieistniejącego klawisza.

Po obiegnięciu wszystkich wyświetlaczy/klawiszy w AgrKey zbierze się nam
 0 jeżeli nie naciśnięto żadnego klawisza,
 1 jeżeli naciśnięto Enter,
 2 " " Esc,
 4 " " W_prawo,
 8 " " W_górę,
16 " " W_dół,
32 " " W_lewo,
albo suma tych wartości jeżeli naciśnięto więcej niż jeden klawisz.
(c:\skotyra\doc\dsm51_IO.pdf, strona 26, ramka
"Dołączenie wyświetlacza 7-segmentowego i klawiatur", kolumny
"Bufor wyboru wsk." "wskaźnik" / "klawisz".

Przy stanie R7=7 wysyłamy AgrKey na diody F1.. F4, OK, ER.

W programie ksq20.asm stan klawiatury był testowany ~1000 razy na sekundę,
w programie ksq30.asm każdy klawisz jest testowany osobno i odbywa się to
1000 / 7 ~= 143 razy na sekundę. Oznacza to, że wymagana liczba powtórzeń,
po której można stan klawiatury traktować jako ustabilizowany
jest siedmiokrotnie mniejsza. W zupełności wystarczy 5 lub 8 powtórzeń.
Przy wartościach większych od 10 łatwo osiągnąć stan, gdy zostanie przegapione
celowe dwukrotne szybkie naciśniecie klawisza.

Podsumowując: w stosunku do 7seg13.asm do pętli wyświetlającego cyfry 654321
(oraz wzorek cyfry 7 na diodach F1.. F4, OK, ER)
w pętli LoopRun programu ksq30.asm dołożono:
1) kod prowizorycznie zastępujący oczekiwanie na przerwanie zegarowe (linijki 41.. 45),
2) testowanie i agregowanie stanu klawiatury (linijki 53, 56),
3) w linijce 62 rozdzielamy wyświetlanie na regularne wyświetlenie cyfr 654321
   co robi kod z linijek 66 do 70, albo wyświetlenie stanu klawiatury na diodach
   F1.. F4, OK, ER za co odpowiada kod z linijek 63, 64.
   Część wspólna wyświetlania to linijki 72.. 76.

Po obiegnięciu wszystkich wyświetlaczy i zagregowaniu stanu KSQ program przechodzi
do linijki 81 i następnych. Ogólnie idea działania tego fragmentu kodu jest taka sama
jak w programie ksq20.asm
1) jeżeli nic nie jest naciśnięte albo akurat wszystkie klawisze zostały zwolnione
   (co testujemy w linijce 84) ograniczamy się do odnowienia stanu licznika powtórzeń
   (linijka 101) i zapamiętania poprzedniego stanu klawiatury (linijka 102)
2) jeżeli obecny stan klawiatury jest inny niż poprzedni stan klawiatury
   (między innymi naciśnięcie/zwolnienie dodatkowego klawisza)
   (co testujemy w linijce 87) robimy to samo co w przypadku 1.
3) w linijkach 89, 90 sprawdzamy czy licznik powtórzeń klawiatury ma wartość zero.
   oznacza to, że naciśnięty klawisz został już zauważony i obsłużony wcześniej
   i teraz niczego więcej nie musimy robić

```
      JZ    LoopIni           ;klawisz już wcześniej został zauważony i obsłużony
```

4) w linijce 92 obniżamy stan licznika powtórzeń KeyCount i testujemy
   czy akurat nie osiągnął stanu zero. Jeżeli nie, to stan klawiatury
   może jeszcze nie być ustabilizowany,
5) jeżeli w linijce 92 akurat osiągnięto stan zero, oznacza to, że stan klawiatury
   możemy traktować jako ustabilizowany i trzeba klawiaturę obsłużyć.
   W omawianym programie ograniczamy się wyłącznie do odwrócenia stanu diody TEST
   (linijka 96: CPL LDTEST) co umożliwia zaobserwowanie, że obsługujemy klawiaturę
   bez zastrzeżeń (nie ma podwójnych reakcji ani reakcji na zwalnianie klawisza).