DOS 6.20 (038-N) MCS-51 MACRO ASSEMBLER, V2.2
OBJECT MODULE PLACED IN TEM_PA.OBJ
ASSEMBLER INVOKED BY: C:\ASM51\ASM51.EXE TEM_PA.ASM

```
LOC OBJ LINE SOURCE
1 $TITLE( *** TEMPORIZZATORE PER PA VHF CLOCK 2 MHz Ver. 3.1 *** )
2 $NOLIST
6
7 ;0000000000000000000000000000000000000000000000000000000000000000000000000000
8 ;
9 ; PROGRAMMA PER TEMPORIZZATORE RITARDI
10 ;
11 ; AMPLIFICATORE VHF made in IW4BLG
12 ;
13 ;
14 ; BY IK4MTS VER. 3.1
15 ;
16 ; DATA: 04/12/1998
17 ;
18 ; PROGRAMMATORE: Grossi Claudio
19 ;
20 ; 8031 clock 2 MHz
21 ;
22 ;
23 ;0000000000000000000000000000000000000000000000000000000000000000000000000000
24 ;
113
114 ;#####################################################################
115 ;
116 ; T0 pin 14 = out MASTER attivo a livello basso
117 ; T1 pin 15 = out Watch-dog
118 ; P1.0 pin 1 = in pulsante ON attivo a livello basso
119 ; P1.1 pin 2 = in pulsante OFF attivo a livello basso
120 ; P1.2 pin 3 = out FAN_PA attivo a livello alto
121 ; P1.3 pin 4 = out HEATHER attivo a livello alto
122 ; P1.4 pin 5 = out GRID2 attivo a livello alto
123 ; P1.5 pin 6 = out GRID1 attivo a livello alto
124 ; P1.6 pin 7 = out HT2 attivo a livello alto
125 ; P1.7 pin 8 = out HT1 attivo a livello alto
126 ; P3.1 pin 11 = out messaggio "PWR FAIL" in CW
127 ;
128 ;#####################################################################
129
130
131 ;*********************************
132 ;* *
133 ;* Equate vari *
134 ;* *
135 ;*********************************
136
008C 137 on_off0 equ tr0
008E 138 on_off1 equ tr1
008D 139 flag0 equ tf0
008F 140 flag1 equ tf1
007F 141 end_ramint equ 07fh ;fissa l'indirizzo di fine ram interna
0019 142 num_deb equ 25 ;numero di fail consecutive per debonce
143
---- 144 dseg
0020 145 org 20h
146
0020 147 flag_gen: ds 1 ;flag generico
0001 148 stato bit flag_gen.1 ;se 0 stato OFF se 1 stato ON
0002 149 f_deb bit flag_gen.2 ;se 1 test debonce fallito
0003 150 run_change bit flag_gen.3 ;se 1 cambio stato in RUN
0004 151 pwr_fail bit flag_gen.4 ;se 1 pwr fail in corso
152
---- 153 bseg
154
0090 155 pa_on bit p1.0
0091 156 pa_off bit p1.1
0092 157 fan_pa bit p1.2
0093 158 heather bit p1.3
0094 159 grid2 bit p1.4
0095 160 grid1 bit p1.5
0096 161 ht2 bit p1.6
0097 162 ht1 bit p1.7
00B4 163 master bit t0
00B1 164 out_cw bit txd
165
166
---- 167 dseg
0030 168 org 30h
169
0030 170 cnt_er_deb: ds 1 ;contatore per debounce fail
0031 171 n_stato: ds 1 ;indice di stato ON/OFF
0032 172 cnt_char: ds 1 ;contatore caratteri messaggio CW
0033 173 cw_char: ds 1 ;pattern-bit del carattere cw
0034 174 cnt_shift: ds 1 ;contatore degli shift carattere
0035 175 cnt_wd: ds 1 ;contatore per watch dog
176
177 ;*****************************************************
178 ; DEFINIZIONE AUTOMATICA INIZIO STACK-POINTER
179 ;
180 ; Inserire PRIMA di queste righe le variabili di
181 ; RAM interna
182 ;*****************************************************
183
0036 184 beg_stk equ $ ;setta inizio stack QUI!
007F 185 end_stk equ end_ramint ;setta fine stack a fine ram interna
186
187
---- 188 cseg
```

```
0000 189 org 0
0000 802E 190 sjmp init
191
0030 192 org 30h
193
194 init:
195
0030 758136 196 mov sp,#beg_stk ;sposta lo stack pointer
197
198 ;#######################################################
199 ;*******************************************************
200 ; ROUTINE PER LA CANCELLAZIONE DEI PRIMI 10 BYTES
201 ;
202 ; DI RAM INTERNA PER EVITARE PROBLEMI DOPO UN RST SW
203 ;*******************************************************
204 ;
205 ; mov r0,#beg_stk
206 ;loop7:
207 ; mov @r0,#0
208 ; inc r0
209 ; cjne r0,#10+beg_stk,loop7
210 ;
211 ;*******************************************************
212 ;#######################################################
213 ;
214 ; ELIMINATA CON LA MODIFICA DI EXT_INT0
215 ;
216 ;#######################################################
217
218
0033 D2B4 219 setb master ;alza il pin 14
0035 D2AB 220 setb et1 ;abilita interrupt timer 1
0037 D288 221 setb it0 ;int0 trigger-edge
0039 D28A 222 setb it1 ;int1 trigger-edge
003B C292 223 clr fan_pa ;resetta uscita FAN_PA
003D C293 224 clr heather ;resetta uscita HEATHER
003F C294 225 clr grid2 ;resetta uscita GRID2
0041 C295 226 clr grid1 ;resetta uscita GRID1
0043 C296 227 clr ht2 ;resetta uscita HT2
0045 C297 228 clr ht1 ;resetta uscita HT1
0047 C2B5 229 clr t1 ;resetta pin 15
0049 C2B1 230 clr out_cw ;resetta pin 11
004B C204 231 clr pwr_fail ;resetta flag pwr_fail
004D C201 232 clr stato ;resetta flag di stato ( OFF )
004F 758DCC 233 mov th1,#0cch ;carica 0cch nel byte alto timer1
0052 758BCC 234 mov tl1,#0cch ;carica 0cch nel byte basso timer1
0055 758911 235 mov tmod,#00010001b ;setta T1 & T2 modo 1
0058 753100 236 mov n_stato,#0 ;azzera n_stato
005B 75320D 237 mov cnt_char,#13 ;carica il contatore caratteri con #13
005E 753300 238 mov cw_char,#0 ;carica 0 nel byte del char CW
0061 753408 239 mov cnt_shift,#8 ;carica #8 nel contatore degli shift
0064 753505 240 mov cnt_wd,#5 ;carica #5 nel contatore per wd
0067 C203 241 clr run_change ;resetta flag di cambio RUN-TIME
242
243 ;*********************************************************************
244 ; Routine di avvio del Timer1
245 ;
246 ; ad uso del RTI & WATCH-DOG
247
0069 D2AF 248 setb ea ;abilita gli interrupt individualmente
006B D28E 249 setb on_off1 ;avvia il timer1
250
251 ;*********************************************************************
252
006D 1201F8 253 lcall del_1ssp ;chiama la sub 1" di ritardo speciale
0070 C2B4 254 clr master ;abbassa il pin 14
0072 D2A8 255 setb ex0 ;abilita int0
0074 D2AA 256 setb ex1 ;abilita int1
257
258 main:
0076 300103 259 jnb stato,to_on ;se stato = 0 salta a to_on
0079 020139 260 ljmp to_off ;se stato <> 0 salta a to_off
261 to_on:
007C 2090F7 262 jb pa_on,main ;se pin 1 a livello alto salta a main
263
264 ;*********************************
265 ;* Inizio debounce di un secondo *
266 ;* *
267 ;* su ingresso ON *
268 ;* *
269 ;* con frequenza di 3,9 msec *
270 ;*********************************
271
007F C202 272 clr f_deb ;azzera il flag per debounce
0081 753019 273 mov cnt_er_deb,#num_deb ;carica il valore per il filtro sw
0084 7AFF 274 mov r2,#255 ;in r2 il numero dei loop da eseguire
275
276 de_off_on:
0086 C28D 277 clr flag0 ;resetta flag timer0
0088 758CFD 278 mov th0,#0fdh ;inizializza th0
008B 758A75 279 mov tl0,#75h ;inizializza tl0
008E D28C 280 setb on_off0 ;avvia il timer0
0090 308DFD 281 jnb flag0,$ ;attendi qui che flag0 = 1
0093 C28C 282 clr on_off0 ;ferma il timer0
0095 309008 283 jnb pa_on,step_1a ;se pin 1 = 0 debounce OK, salta a step_1a
0098 D202 284 setb f_deb ;setta flag per debounce
009A D5300D 285 djnz cnt_er_deb,step_2a ;dec e se <> 0 salta a step_2a
009D 020076 286 ljmp main ;se = 0 debounce fallito percio' main
```

```
287 step_1a:
00A0 300207 288 jnb f_deb,step_2a ;se flag = 0 allora step_2a
```

```
00A3 C202 289 clr f_deb ;azzera il flag f_deb
00A5 753019 290 mov cnt_er_deb,#num_deb ;carica il valore per il filtro sw
00A8 7AFF 291 mov r2,#255 ;in r2 il numero dei loop da eseguire
292 step_2a:
00AA DADA 293 djnz r2,de_off_on ;dec r2 e salta a de_off_on se <> 0
294
00AC D201 295 setb stato ;setta il flag stato ( ON )
296
297 ;*********************************************************************
298 ;* *
299 ;* Inizio della procedura di ON con le seguenti temporizzazioni: *
300 ;* *
301 ;* Istante 0" ON FAN_PA *
302 ;* " 1 ON HEATHER *
303 ;* " 181" ON GRID1 *
304 ;* " 183" ON HT1 *
305 ;* " 188" ON HT2 *
306 ;* " 191" ON GRID2 *
307 ;* *
308 ;*********************************************************************
309
310 caso_on_0:
00AE D292 311 setb fan_pa ;avvia la ventola di raffreddamento
00B0 753102 312 mov n_stato,#2 ;n_stato = 2
00B3 12020C 313 lcall del_1s ;chiama la sub 1" di ritardo
00B6 300305 314 jnb run_change,caso_on_1;se flag = 0 salta
00B9 C203 315 clr run_change ;azzera il flag
00BB 020127 316 ljmp pro_on_off ;salta a pro_on_off
317 caso_on_1:
00BE D293 318 setb heather ;attiva i filamenti
00C0 753104 319 mov n_stato,#4 ;n_stato = 4
00C3 120262 320 lcall del_180s ;chiama la sub 180" di ritardo
00C6 300305 321 jnb run_change,caso_on_2;se flag = 0 salta
00C9 C203 322 clr run_change ;azzera il flag
00CB 020127 323 ljmp pro_on_off ;salta a pro_on_off
324 caso_on_2:
00CE D295 325 setb grid1 ;attiva tensione griglia 1
00D0 753106 326 mov n_stato,#6 ;n_stato = 6
00D3 12020C 327 lcall del_1s ;chiama la sub 1" di ritardo
00D6 300305 328 jnb run_change,cas_on_2b;se flag = 0 salta
00D9 C203 329 clr run_change ;azzera il flag
00DB 020127 330 ljmp pro_on_off ;salta a pro_on_off
331 cas_on_2b:
00DE 12020C 332 lcall del_1s ;chiama la sub 1" di ritardo
00E1 300305 333 jnb run_change,caso_on_3;se flag = 0 salta
00E4 C203 334 clr run_change ;azzera il flag
00E6 020127 335 ljmp pro_on_off ;salta a pro_on_off
336 caso_on_3:
00E9 D297 337 setb ht1 ;attiva alta tensione 1
00EB 753108 338 mov n_stato,#8 ;n_stato = 8
00EE 120237 339 lcall del_5s ;chiama la sub 5" di ritardo
00F1 300305 340 jnb run_change,caso_on_4;se flag = 0 salta
00F4 C203 341 clr run_change ;azzera il flag
00F6 020127 342 ljmp pro_on_off ;salta a pro_on_off
343 caso_on_4:
00F9 D296 344 setb ht2 ;attiva alta tensione 2
00FB 75310A 345 mov n_stato,#10 ;n_stato = 10
00FE 12020C 346 lcall del_1s ;chiama la sub 1" di ritardo
0101 300305 347 jnb run_change,cas_on_4b;se flag = 0 salta
0104 C203 348 clr run_change ;azzera il flag
0106 020127 349 ljmp pro_on_off ;salta a pro_on_off
350 cas_on_4b:
0109 12020C 351 lcall del_1s ;chiama la sub 1" di ritardo
010C 300305 352 jnb run_change,cas_on_4c;se flag = 0 salta
010F C203 353 clr run_change ;azzera il flag
0111 020127 354 ljmp pro_on_off ;salta a pro_on_off
355 cas_on_4c:
0114 12020C 356 lcall del_1s ;chiama la sub 1" di ritardo
0117 300305 357 jnb run_change,caso_on_5;se flag = 0 salta
011A C203 358 clr run_change ;azzera il flag
011C 020127 359 ljmp pro_on_off ;salta a pro_on_off
360 caso_on_5:
011F D294 361 setb grid2 ;attiva tensione griglia 2
0121 75310C 362 mov n_stato,#12 ;n_stato = 12
0124 020076 363 ljmp main ;salta a main
364
365 ;*****************************************
366 ; Procedura di spegnimento
367 ; RUN-TIME
368 ;*****************************************
369 pro_on_off:
0127 E531 370 mov a,n_stato ;copia in A n_stato
0129 90012D 371 mov dptr,#tab_on_off ;carica indir. tabella tab_on_off
012C 73 372 jmp @a+dptr ;indirizzamento del salto
373 tab_on_off:
012D 00 374 nop ;NON MODIFICARE serve come offset
012E 00 375 nop ;NON MODIFICARE serve come offset
012F 21DE 376 ajmp caso_off_1 ;|
0131 21CE 377 ajmp caso_off_2 ;||
0133 21BE 378 ajmp caso_off_3 ;||| selezione caso_off_(x)
0135 21A3 379 ajmp caso_off_4 ;||
0137 2188 380 ajmp caso_off_5 ;|
381
382 to_off:
0139 309102 383 jnb pa_off,offset_ko1 ;se pin 2 a livello basso salta a debounce
013C 0176 384 ajmp main ;se = 1 salta a main
Pagina 4
tem_pa.lst
385 offset_ko1:
386
387 ;*********************************
388 ;* Inizio debounce di un secondo *
389 ;* *
390 ;* su ingresso OFF *
```

```
391 ;* *
392 ;* con frequenza di 3,9 msec *
393 ;*********************************
394
013E C202 395 clr f_deb ;azzera il flag per debounce
0140 753019 396 mov cnt_er_deb,#num_deb ;carica il valore per il filtro sw
0143 7AFF 397 mov r2,#255 ;in r2 il numero dei loop da eseguire
398
399 de_on_off:
0145 C28D 400 clr flag0 ;resetta flag timer0
0147 758CFD 401 mov th0,#0fdh ;inizializza th0
014A 758A75 402 mov tl0,#75h ;inizializza tl0
014D D28C 403 setb on_off0 ;avvia il timer0
014F 308DFD 404 jnb flag0,$ ;attendi qui che flag0 = 1
0152 C28C 405 clr on_off0 ;ferma il timer0
0154 309108 406 jnb pa_off,step_1b ;se pin 2 = 0 debounce OK, salta a step_1b
0157 D202 407 setb f_deb ;setta flag per debounce
0159 D5300D 408 djnz cnt_er_deb,step_2b ;dec e se <> 0 salta a step_2b
015C 020076 409 ljmp main ;se = 0 debounce fallito percio' main
410 step_1b:
015F 300207 411 jnb f_deb,step_2b ;se flag = 0 allora step_2b
0162 C202 412 clr f_deb ;azzera il flag f_deb
0164 753019 413 mov cnt_er_deb,#num_deb ;carica il valore per il filtro sw
0167 7AFF 414 mov r2,#255 ;in r2 il numero dei loop da eseguire
415 step_2b:
0169 DADA 416 djnz r2,de_on_off ;dec r2 e salta a de_on_off se <> 0
417
016B C201 418 clr stato ;resetta il flag stato ( OFF )
419
420 ;*********************************************************************
421 ;* *
422 ;* Inizio della procedura di OFF con le seguenti temporizzazioni: *
423 ;* *
424 ;* Istante 0" OFF GRID2 *
425 ;* " 2" OFF HT2 *
426 ;* " 4" OFF HT1 *
427 ;* " 6" OFF GRID1 *
428 ;* " 11" OFF HEATHER *
429 ;* " 241" OFF FAN_PA *
430 ;* *
431 ;*********************************************************************
432
433 caso_off_6:
016D C294 434 clr grid2 ;togli tensione griglia 2
016F 75310A 435 mov n_stato,#10 ;n_stato = 10
0172 12020C 436 lcall del_1s ;chiama la sub 1" di ritardo
0175 300305 437 jnb run_change,ca_off_6b ;se flag = 0 salta
0178 C203 438 clr run_change ;azzera il flag
017A 0201E6 439 ljmp pro_off_on ;salta a pro_off_on
440 ca_off_6b:
017D 12020C 441 lcall del_1s ;chiama la sub 1" di ritardo
0180 300305 442 jnb run_change,caso_off_5 ;se flag = 0 salta
0183 C203 443 clr run_change ;azzera il flag
0185 0201E6 444 ljmp pro_off_on ;salta a pro_off_on
445 caso_off_5:
0188 C296 446 clr ht2 ;togli alta tensione 2
018A 753108 447 mov n_stato,#8 ;n_stato = 8
018D 12020C 448 lcall del_1s ;chiama la sub 1" di ritardo
0190 300305 449 jnb run_change,ca_off_5b ;se flag = 0 salta
0193 C203 450 clr run_change ;azzera il flag
0195 0201E6 451 ljmp pro_off_on ;salta a pro_off_on
452 ca_off_5b:
0198 12020C 453 lcall del_1s ;chiama la sub 1" di ritardo
019B 300305 454 jnb run_change,caso_off_4 ;se flag = 0 salta
019E C203 455 clr run_change ;azzera il flag
01A0 0201E6 456 ljmp pro_off_on ;salta a pro_off_on
457 caso_off_4:
01A3 C297 458 clr ht1 ;togli alta tensione 1
01A5 753106 459 mov n_stato,#6 ;n_stato = 6
01A8 12020C 460 lcall del_1s ;chiama la sub 1" di ritardo
01AB 300305 461 jnb run_change,ca_off_4b ;se flag = 0 salta
01AE C203 462 clr run_change ;azzera il flag
01B0 0201E6 463 ljmp pro_off_on ;salta a pro_off_on
464 ca_off_4b:
01B3 12020C 465 lcall del_1s ;chiama la sub 1" di ritardo
01B6 300305 466 jnb run_change,caso_off_3 ;se flag = 0 salta
01B9 C203 467 clr run_change ;azzera il flag
01BB 0201E6 468 ljmp pro_off_on ;salta a pro_off_on
469 caso_off_3:
01BE C295 470 clr grid1 ;togli tensione griglia 1
01C0 753104 471 mov n_stato,#4 ;n_stato = 4
01C3 120237 472 lcall del_5s ;chiama la sub 5" di ritardo
01C6 300305 473 jnb run_change,caso_off_2 ;se flag = 0 salta
01C9 C203 474 clr run_change ;azzera il flag
01CB 0201E6 475 ljmp pro_off_on ;salta a pro_off_on
476 caso_off_2:
01CE C293 477 clr heather ;togli tensione ai filamenti
01D0 753102 478 mov n_stato,#2 ;n_stato = 2
01D3 120291 479 lcall del_230s ;chiama la sub 230" di ritardo
01D6 300305 480 jnb run_change,caso_off_1 ;se flag = 0 salta
01D9 C203 481 clr run_change ;azzera il flag
01DB 0201E6 482 ljmp pro_off_on ;salta a pro_off_on
Pagina 5
tem_pa.lst
483 caso_off_1:
01DE C292 484 clr fan_pa ;arresta la ventola di raffreddamento
01E0 753100 485 mov n_stato,#0 ;n_stato = 0
01E3 020076 486 ljmp main ;salta a main
487
488
489 ;*****************************************
490 ; Procedura di accensione
491 ; RUN-TIME
492 ;*****************************************
```

```
493 pro_off_on:
01E6 E531 494 mov a,n_stato ;copia in A n_stato
01E8 9001EC 495 mov dptr,#tab_off_on ;carica indir. tabella tab_off_on
01EB 73 496 jmp @a+dptr ;indirizzamento del salto
497 tab_off_on:
01EC 00 498 nop ;NON MODIFICARE serve come offset
01ED 00 499 nop ;NON MODIFICARE serve come offset
01EE 01BE 500 ajmp caso_on_1 ;|
01F0 01CE 501 ajmp caso_on_2 ;||
01F2 01E9 502 ajmp caso_on_3 ;||| selezione caso_on_(x)
01F4 01F9 503 ajmp caso_on_4 ;||
01F6 211F 504 ajmp caso_on_5 ;|
505
506
507
508 ;*********************************
509 ;* Routine di generazione *
510 ;* *
511 ;* ritardo di 1 sec *
512 ;* *
513 ;* senza test di inversione *
514 ;*********************************
515 del_1ssp:
01F8 7A03 516 mov r2,#3 ;in r2 il primo # di loop
517 loop8:
01FA C28D 518 clr flag0 ;resetta flag timer0
01FC 758C27 519 mov th0,#27h ;inizializza th0
01FF 758A33 520 mov tl0,#33h ;inizializza tl0
0202 D28C 521 setb on_off0 ;avvia il timer0
0204 308DFD 522 jnb flag0,$ ;attendi qui che flag0 = 1
0207 C28C 523 clr on_off0 ;ferma il timer0
0209 DAEF 524 djnz r2,loop8 ;se r3 <> 0 salta a loop8
020B 22 525 ret ;ritorna alla chiamata
526
527
528
529 ;*********************************
530 ;* Routine di generazione *
531 ;* *
532 ;* ritardo di 1 sec *
533 ;* *
534 ;* con test di inversione *
535 ;*********************************
536 del_1s:
020C 7A03 537 mov r2,#3 ;in r2 il primo # di loop
538 loop3:
020E C28D 539 clr flag0 ;resetta flag timer0
0210 758C27 540 mov th0,#27h ;inizializza th0
0213 758A33 541 mov tl0,#33h ;inizializza tl0
0216 D28C 542 setb on_off0 ;avvia il timer0
0218 308DFD 543 jnb flag0,$ ;attendi qui che flag0 = 1
021B C28C 544 clr on_off0 ;ferma il timer0
021D 30010A 545 jnb stato,tst_on1 ;se stato = 0 test su ON else su OFF
546 tst_off1:
0220 209111 547 jb pa_off,com_1 ;se in-OFF = 1 continua delay (com_x)
0223 1202FD 548 lcall run_de_off ;se = 0 chiama il debonce run-time
0226 30030B 549 jnb run_change,com_1;se flag run_change = 0 vai com(x)
0229 22 550 ret ;else termina delay
551 tst_on1:
022A 209007 552 jb pa_on,com_1 ;se in-ON = 1 continua delay (com_x)
022D 1202C0 553 lcall run_de_on ;se = 0 chiama il debonce run-time
0230 300301 554 jnb run_change,com_1;se flag run_change = 0 vai com(x)
0233 22 555 ret ;else termina delay
556 com_1:
0234 DAD8 557 djnz r2,loop3 ;se r2 <> 0 salta a loop3
0236 22 558 ret ;ritorna alla chiamata
559
560
561 ;*********************************
562 ;* Routine di generazione *
563 ;* *
564 ;* ritardo di 5 sec *
565 ;* *
566 ;* con test di inversione *
567 ;*********************************
568 del_5s:
0237 7A0F 569 mov r2,#15 ;in r2 il primo # di loop
570 loop6:
0239 C28D 571 clr flag0 ;resetta flag timer0
023B 758C27 572 mov th0,#27h ;inizializza th0
023E 758A33 573 mov tl0,#33h ;inizializza tl0
0241 D28C 574 setb on_off0 ;avvia il timer0
0243 308DFD 575 jnb flag0,$ ;attendi qui che flag0 = 1
0246 C28C 576 clr on_off0 ;ferma il timer0
0248 30010A 577 jnb stato,tst_on2 ;se stato = 0 test su ON else su OFF
578 tst_off2:
024B 209111 579 jb pa_off,com_2 ;se in-OFF = 1 continua delay (com_x)
024E 1202FD 580 lcall run_de_off ;se = 0 chiama il debonce run-time
Pagina 6
tem_pa.lst
0251 30030B 581 jnb run_change,com_2;se flag run_change = 0 vai com(x)
0254 22 582 ret ;else termina delay
583 tst_on2:
0255 209007 584 jb pa_on,com_2 ;se in-ON = 1 continua delay (com_x)
0258 1202C0 585 lcall run_de_on ;se = 0 chiama il debonce run-time
025B 300301 586 jnb run_change,com_2;se flag run_change = 0 vai com(x)
025E 22 587 ret ;else termina delay
588 com_2:
025F DAD8 589 djnz r2,loop6 ;se r3 <> 0 salta a loop6
0261 22 590 ret ;ritorna alla chiamata
591
592
593
594 ;*********************************
```

```
595 ;* Routine di generazione *
596 ;* *
597 ;* ritardo di 180 sec *
598 ;* *
599 ;* con test di inversione *
600 ;**********************************
601 del_180s:
0262 7A03 602 mov r2,#3 ;in r2 il primo # di loop
603 loop1:
0264 7B98 604 mov r3,#98h ;in r3 il secondo # di loop
605 loop2:
0266 C28D 606 clr flag0 ;resetta flag timer0
0268 758C00 607 mov th0,#0 ;inizializza th0
026B 758A00 608 mov tl0,#0h ;inizializza tl0
026E D28C 609 setb on_off0 ;avvia il timer0
0270 308DFD 610 jnb flag0,$ ;attendi qui che flag0 = 1
0273 C28C 611 clr on_off0 ;ferma il timer0
0275 30010A 612 jnb stato,tst_on3 ;se stato = 0 test su ON else su OFF
613 tst_off3:
0278 209111 614 jb pa_off,com_3 ;se in-OFF = 1 continua delay (com_x)
027B 1202FD 615 lcall run_de_off ;se = 0 chiama il debonce run-time
027E 30030B 616 jnb run_change,com_3;se flag run_change = 0 vai com(x)
0281 22 617 ret ;else termina delay
618 tst_on3:
0282 209007 619 jb pa_on,com_3 ;se in-ON = 1 continua delay (com_x)
0285 1202C0 620 lcall run_de_on ;se = 0 chiama il debonce run-time
0288 300301 621 jnb run_change,com_3;se flag run_change = 0 vai com(x)
028B 22 622 ret ;else termina delay
623 com_3:
028C DBD8 624 djnz r3,loop2 ;se r3 <> 0 salta a loop2
028E DAD4 625 djnz r2,loop1 ;se r2 <> 0 salta a loop1
0290 22 626 ret ;ritorna alla chiamata
627
628
629
630 ;**********************************
631 ;* Routine di generazione *
632 ;* *
633 ;* ritardo di 230 sec *
634 ;* *
635 ;* con test di inversione *
636 ;**********************************
637 del_230s:
0291 7A04 638 mov r2,#4 ;in r2 il primo # di loop
639 loop4:
0293 7B92 640 mov r3,#92h ;in r3 il secondo # di loop
641 loop5:
0295 C28D 642 clr flag0 ;resetta flag timer0
0297 758C00 643 mov th0,#0 ;inizializza th0
029A 758A00 644 mov tl0,#0h ;inizializza tl0
029D D28C 645 setb on_off0 ;avvia il timer0
029F 308DFD 646 jnb flag0,$ ;attendi qui che flag0 = 1
02A2 C28C 647 clr on_off0 ;ferma il timer0
02A4 30010A 648 jnb stato,tst_on4 ;se stato = 0 test su ON else su OFF
649 tst_off4:
02A7 209111 650 jb pa_off,com_4 ;se in-OFF = 1 continua delay (com_x)
02AA 1202FD 651 lcall run_de_off ;se = 0 chiama il debonce run-time
02AD 30030B 652 jnb run_change,com_4;se flag run_change = 0 vai com(x)
02B0 22 653 ret ;else termina delay
654 tst_on4:
02B1 209007 655 jb pa_on,com_4 ;se in-ON = 1 continua delay (com_x)
02B4 1202C0 656 lcall run_de_on ;se = 0 chiama il debonce run-time
02B7 300301 657 jnb run_change,com_4;se flag run_change = 0 vai com(x)
02BA 22 658 ret ;else termina delay
659 com_4:
02BB DBD8 660 djnz r3,loop5 ;se r3 <> 0 salta a loop5
02BD DAD4 661 djnz r2,loop4 ;se r2 <> 0 salta a loop4
02BF 22 662 ret ;ritorna alla chiamata
663
664
665 ;**********************************
666 ;* Inizio debounce di un secondo *
667 ;* su ingresso ON *
668 ;* in RUN TIME *
669 ;* con frequenza di 3,9 msec *
670 ;**********************************
671 run_de_on:
02C0 C08A 672 push tl0 ;salva tl0
02C2 C08C 673 push th0 ;salva th0
02C4 C002 674 push 02h ;salva r2
02C6 C202 675 clr f_deb ;azzera il flag per debounce
02C8 753019 676 mov cnt_er_deb,#num_deb ;carica il valore per il filtro sw
02CB 7AFF 677 mov r2,#255 ;in r2 il numero dei loop da eseguire
678
Pagina 7
tem_pa.lst
679 de_run_on:
02CD C28D 680 clr flag0 ;resetta flag timer0
02CF 758CFD 681 mov th0,#0fdh ;inizializza th0
02D2 758A75 682 mov tl0,#75h ;inizializza tl0
02D5 D28C 683 setb on_off0 ;avvia il timer0
02D7 308DFD 684 jnb flag0,$ ;attendi qui che flag0 = 1
02DA C28C 685 clr on_off0 ;ferma il timer0
02DC 309007 686 jnb pa_on,pass_1a ;se pin 1 = 0 debounce OK, salta a pass_1a
02DF D202 687 setb f_deb ;setta flag per debounce
02E1 D5300C 688 djnz cnt_er_deb,pass_2a ;dec e se <> 0 salta a pass_2a
02E4 8010 689 sjmp torna ;se = 0 debounce fallito percio' torna
690 pass_1a:
02E6 300207 691 jnb f_deb,pass_2a ;se flag = 0 allora pass_2a
02E9 C202 692 clr f_deb ;azzera il flag f_deb
02EB 753019 693 mov cnt_er_deb,#num_deb ;carica il valore per il filtro sw
02EE 7AFF 694 mov r2,#255 ;in r2 il numero dei loop da eseguire
695 pass_2a:
02F0 DADB 696 djnz r2,de_run_on ;dec r2 e salta a de_run_on se <> 0
```

```
02F2 D203 697 setb run_change ;setta flag di cambio run-time
02F4 D201 698 setb stato ;setta il flag stato ( ON )
699 torna:
02F6 D002 700 pop 02h ;restore r2
02F8 D08C 701 pop th0 ;restore th0
02FA D08A 702 pop tl0 ;restore tl0
02FC 22 703 ret ;termina subroutine
704
705
706 ;*********************************
707 ;* Inizio debounce di un secondo *
708 ;* su ingresso OFF *
709 ;* in RUN TIME *
710 ;* con frequenza di 3,9 msec *
711 ;*********************************
712 run_de_off:
02FD C08A 713 push tl0 ;salva tl0
02FF C08C 714 push th0 ;salva th0
0301 C002 715 push 02h ;salva r2
0303 C202 716 clr f_deb ;azzera il flag per debounce
0305 753019 717 mov cnt_er_deb,#num_deb ;carica il valore per il filtro sw
0308 7AFF 718 mov r2,#255 ;in r2 il numero dei loop da eseguire
719
720 de_run_off:
030A C28D 721 clr flag0 ;resetta flag timer0
030C 758CFD 722 mov th0,#0fdh ;inizializza th0
030F 758A75 723 mov tl0,#75h ;inizializza tl0
0312 D28C 724 setb on_off0 ;avvia il timer0
0314 308DFD 725 jnb flag0,$ ;attendi qui che flag0 = 1
0317 C28C 726 clr on_off0 ;ferma il timer0
0319 309107 727 jnb pa_off,pass_1b ;se pin 2 = 0 debounce OK, salta a pass_1b
031C D202 728 setb f_deb ;setta flag per debounce
031E D5300C 729 djnz cnt_er_deb,pass_2b ;dec e se <> 0 salta a pass_2b
0321 8010 730 sjmp tornb ;se = 0 debounce fallito percio' tornb
731 pass_1b:
0323 300207 732 jnb f_deb,pass_2b ;se flag = 0 allora pass_2b
0326 C202 733 clr f_deb ;azzera il flag f_deb
0328 753019 734 mov cnt_er_deb,#num_deb ;carica il valore per il filtro sw
032B 7AFF 735 mov r2,#255 ;in r2 il numero dei loop da eseguire
736 pass_2b:
032D DADB 737 djnz r2,de_run_off ;dec r2 e salta a de_run_off se <> 0
032F D203 738 setb run_change ;setta flag di cambio run-time
0331 C201 739 clr stato ;resetta il flag stato ( OFF )
740 tornb:
0333 D002 741 pop 02h ;restore r2
0335 D08C 742 pop th0 ;restore th0
0337 D08A 743 pop tl0 ;restore tl0
0339 22 744 ret ;termina subroutine
745
746 cw_msg:
033A 00 747 nop ;offset per la tabella
033B 00 748 db 00000000b ;inizio tabella messaggio CW
033C 00 749 db 00000000b ;
033D 00 750 db 00000000b ;scansionata dal "basso" verso "l'alto"
033E A0 751 db 10100000b
033F 2E 752 db 00101110b
0340 8A 753 db 10001010b
0341 8B 754 db 10001011b
0342 AE 755 db 10101110b
0343 80 756 db 10000000b
0344 2E 757 db 00101110b
0345 EE 758 db 11101110b
0346 A2 759 db 10100010b
0347 BB 760 db 10111011b ;fine tabella messaggio CW
761
762
763
764 ;*********************************
765 ;* *
766 ;* Routine di interrupt *
767 ;* *
768 ;* R T I ogni 78.6 msec *
769 ;* *
770 ;*********************************
771
0348 772 rti equ $
773
001B 774 org 1bh ;indirizzo vettore OV timer1
001B 020348 775 ljmp rti ;salta alla routine di interrupt
776
Pagina 8
tem_pa.lst
0348 777 org rti
778
0348 C2AF 779 clr ea ;disabilita tutti gli interrupt
034A C0E0 780 push acc ;salva A
034C C28E 781 clr on_off1 ;ferma il timer1
034E 30043A 782 jnb pwr_fail,w_dog ;salta a w_dog se pwr_fail = 0
0351 309006 783 jnb pa_on,res_fail ;salta se pa_on = 0
0354 309103 784 jnb pa_off,res_fail ;salta se pa_off = 0
0357 02036A 785 ljmp tx_code ;salta a tx_code
786 res_fail:
035A C204 787 clr pwr_fail ;resetta flag pwr_fail
035C C2B1 788 clr out_cw ;resetta uscita CW
035E 75320D 789 mov cnt_char,#13 ;carica il contatore caratteri con #13
0361 753300 790 mov cw_char,#0 ;carica 0 nel byte del char CW
0364 753408 791 mov cnt_shift,#8 ;carica #8 nel contatore degli shift
0367 02038B 792 ljmp w_dog ;salta a w_dog
793 tx_code:
036A E534 794 mov a,cnt_shift ;carica A con cnt_shift
036C B40808 795 cjne a,#8,solo_sh ;se <> da #8 salta a solo_sh
036F E532 796 mov a,cnt_char ;carica A con il valore di cnt_char
0371 90033A 797 mov dptr,#cw_msg ;carica il puntatore con indirizzo tabella
0374 93 798 movc a,@a+dptr ;carica in ACC il carattere da TX
```

```
0375 F533 799 mov cw_char,a ;salva in cw_char il contenuto di ACC
800 solo_sh:
0377 E533 801 mov a,cw_char ;sposta in ACC in contenuto di cw_char
0379 C3 802 clr c ;azzera il carry
037A 33 803 rlc a ;ruota ACC a sx attraverso il carry
037B 92B1 804 mov out_cw,c ;muovi in carry in TXD
037D F533 805 mov cw_char,a ;salva in cw_char il contenuto di ACC
037F D53409 806 djnz cnt_shift,w_dog ;dec e salta a w_dog se <> 0
0382 753408 807 mov cnt_shift,#8 ;carica #8 in cnt_shift
0385 D53203 808 djnz cnt_char,w_dog ;dec e salta a w_dog se <> 0
0388 75320D 809 mov cnt_char,#13 ;carica #13 in cnt_msg
810
811 ;*********************************
812 ;* *
813 ;* Routine di gestione *
814 ;* *
815 ;* W A T C H D O G *
816 ;* *
817 ;*********************************
818
819 w_dog:
038B D53505 820 djnz cnt_wd,stesso ;dec e salta se <> 0 a stesso
038E B2B5 821 cpl t1 ;complementa uscita t1 (pin 15)
0390 753505 822 mov cnt_wd,#5 ;carica #5 in cnt_wd
823 stesso:
0393 758DCC 824 mov th1,#0cch ;carica 0cch nel byte alto timer1
0396 758BCC 825 mov tl1,#0cch ;carica 0cch nel byte basso timer1
0399 D28E 826 setb on_off1 ;avvia il timer1
039B D0E0 827 pop acc ;restore A
039D D2AF 828 setb ea ;abilita interrupt singolarmente
829
039F 32 830 reti ;fine interrupt
831
832
833 ;*********************************
834 ;* *
835 ;* Routine di interrupt *
836 ;* per ingresso Power Fail *
837 ;* E X T I N T 0 *
838 ;* *
839 ;*********************************
840
03A0 841 task_int0 equ $
842
0003 843 org 03h ;indirizzo vettore INT0
844
0003 0203A0 845 ljmp task_int0 ;salta alla routine di interrupt
846
03A0 847 org task_int0
848
03A0 C2A8 849 clr ex0 ;disabilita altri INT0
03A2 E531 850 mov a,n_stato ;metti in A il contenuto di n_stato
03A4 D204 851 setb pwr_fail ;setta flag di power fail attivo
03A6 B40002 852 cjne a,#0,tst_div2 ;se <> 0 salta al prossimo test
03A9 8022 853 sjmp fin_int0 ;se = 0 salta a fine interrupt
854 tst_div2:
03AB B40202 855 cjne a,#2,diver1 ;se <> 2 salta a diver1
03AE 801D 856 sjmp fin_int0 ;se = 0 salta a fine interrupt
857 diver1:
03B0 C201 858 clr stato ;resetta flag stato (condizione OFF)
03B2 C293 859 clr heather ;resetta uscita HEATHER
03B4 C295 860 clr grid1 ;resetta uscita GRID1
03B6 C297 861 clr ht1 ;resetta uscita HT1
03B8 C296 862 clr ht2 ;resetta uscita HT2
03BA C294 863 clr grid2 ;resetta uscita GRID2
03BC 753102 864 mov n_stato,#2 ;n_stato = 2
865
03BF C2AF 866 clr ea ;disabilita tutti gli interrupt
03C1 9001CE 867 mov dptr,#caso_off_2 ;punta indirizzo di caso_off_2
03C4 A881 868 mov r0,sp ;indirizzo SP in R0
03C6 A683 869 mov @r0,dph ;contenuto di DPH in @R0
03C8 18 870 dec r0 ;decrementa R0
03C9 A682 871 mov @r0,dpl ;contenuto di DPL in @R0
03CB D2AF 872 setb ea ;abilita interrupt singolarmente
873 fin_int0:
03CD D2A8 874 setb ex0 ;riabilita INT0
Pagina 9
tem_pa.lst
03CF 32 875 reti
876
877
878 ;*********************************
879 ;* *
880 ;* Routine di interrupt *
881 ;* per test in debug *
882 ;* E X T I N T 1 *
883 ;* *
884 ;*********************************
885
03D0 886 task_int1 equ $
887
0013 888 org 13h ;indirizzo vettore INT1
889
0013 0203D0 890 ljmp task_int1 ;salta alla routine di interrupt
891
03D0 892 org task_int1
893
03D0 C2AA 894 clr ex1 ;disabilita altri INT1
03D2 C0E0 895 push acc ;salva A
03D4 C083 896 push dph ;salva DPH
03D6 C082 897 push dpl ;salva DPL
03D8 C002 898 push 02h ;salva R2
03DA C08C 899 push th0 ;salva TH0
03DC C08A 900 push tl0 ;salva TL0
```

```
901
902 ;*********************************
903 ;* Inizio debounce di 47 msec *
904 ;* *
905 ;* su ingresso INT1 *
906 ;* *
907 ;* con frequenza di 3,9 msec *
908 ;*********************************
909 ;###############################################################################
910
03DE C202 911 clr f_deb ;azzera il flag per debounce
03E0 753005 912 mov cnt_er_deb,#5 ;carica il valore per il filtro sw
03E3 7A0C 913 mov r2,#12 ;in r2 il numero dei loop da eseguire
914
915 de_int1:
03E5 C28D 916 clr flag0 ;resetta flag timer0
03E7 758CFD 917 mov th0,#0fdh ;inizializza th0
03EA 758A75 918 mov tl0,#75h ;inizializza tl0
03ED D28C 919 setb on_off0 ;avvia il timer0
03EF 308DFD 920 jnb flag0,$ ;attendi qui che flag0 = 1
03F2 C28C 921 clr on_off0 ;ferma il timer0
03F4 30B308 922 jnb int1,stint1 ;se pin 13 = 0 debounce OK, salta a stint1
03F7 D202 923 setb f_deb ;setta flag per debounce
03F9 D530DD 924 djnz cnt_er_deb,stint2 ;dec e se <> 0 salta a stint2
03FC 020466 925 ljmp fine_int1 ;se = 0 debounce fallito percio' fine
926 stint1:
03FF 300207 927 jnb f_deb,stint2 ;se flag = 0 allora stint2
0402 C202 928 clr f_deb ;azzera il flag f_deb
0404 753005 929 mov cnt_er_deb,#5 ;carica il valore per il filtro sw
0407 7A0C 930 mov r2,#12 ;in r2 il numero dei loop da eseguire
931 stint2:
0409 DADA 932 djnz r2,de_int1 ;dec r2 e salta a de_on_off se <> 0
933
934 ;###############################################################################
935
040B E531 936 mov a,n_stato ;carica A con n_stato
040D 900411 937 mov dptr,#jmp_table ;carica dptr con indir. jmp_table
0410 73 938 jmp @a+dptr ;vai a selezionare il salto
939 jmp_table:
0411 811F 940 ajmp test1 ;|
0413 8128 941 ajmp test2 ;||
0415 8131 942 ajmp test3 ;|||
0417 813A 943 ajmp test4 ;|||| tabella salto routine
0419 8143 944 ajmp test5 ;|||
041B 814C 945 ajmp test6 ;||
041D 8155 946 ajmp test7 ;|
947 test1:
041F D201 948 setb stato ;setta flag stato (condizione di ON)
0421 D292 949 setb fan_pa ;avvia la ventola di raffreddamento
0423 753102 950 mov n_stato,#2 ;n_stato = 2
0426 803E 951 sjmp fine_int1 ;salta a fine_int1
952 test2:
0428 D201 953 setb stato ;setta flag stato (condizione di ON)
042A D293 954 setb heather ;attiva i filamenti
042C 753104 955 mov n_stato,#4 ;n_stato = 4
042F 8035 956 sjmp fine_int1 ;salta a fine_int1
957 test3:
0431 D201 958 setb stato ;setta flag stato (condizione di ON)
0433 D295 959 setb grid1 ;attiva tensione griglia 1
0435 753106 960 mov n_stato,#6 ;n_stato = 6
0438 802C 961 sjmp fine_int1 ;salta a fine_int1
962 test4:
043A D201 963 setb stato ;setta flag stato (condizione di ON)
043C D297 964 setb ht1 ;attiva alta tensione 1
043E 753108 965 mov n_stato,#8 ;n_stato = 8
0441 8023 966 sjmp fine_int1 ;salta a fine_int1
967 test5:
0443 D201 968 setb stato ;setta flag stato (condizione di ON)
0445 D296 969 setb ht2 ;attiva alta tensione 2
0447 75310A 970 mov n_stato,#10 ;n_stato = 10
044A 801A 971 sjmp fine_int1 ;salta a fine_int1
972 test6:
Pagina 10
tem_pa.lst
044C D201 973 setb stato ;setta flag stato (condizione di ON)
044E D294 974 setb grid2 ;attiva tensione griglia 2
0450 75310C 975 mov n_stato,#12 ;n_stato = 12
0453 8011 976 sjmp fine_int1 ;salta a fine_int1
977 test7:
0455 C201 978 clr stato ;resetta flag stato (condizione OFF)
0457 C292 979 clr fan_pa ;resetta uscita FAN_PA
0459 C293 980 clr heather ;resetta uscita HEATHER
045B C295 981 clr grid1 ;resetta uscita GRID1
045D C297 982 clr ht1 ;resetta uscita HT1
045F C296 983 clr ht2 ;resetta uscita HT2
0461 C294 984 clr grid2 ;resetta uscita GRID2
0463 753100 985 mov n_stato,#0 ;n_stato = 0
986 fine_int1:
0466 D08A 987 pop tl0 ;restore TL0
0468 D08C 988 pop th0 ;restore TH0
046A D002 989 pop 02h ;restore R2
046C D082 990 pop dpl ;restore DPL
046E D083 991 pop dph ;restore DPH
0470 D0E0 992 pop acc ;restore A
0472 C28B 993 clr ie1 ;pulisci flag INT1
0474 D2AA 994 setb ex1 ;riabilita INT1
0476 32 995 reti
996 end
MCS-51 MACRO ASSEMBLER *** TEMPORIZZATORE PER PA VHF CLOCK 2 MHz Ver. 3.1 ***
12/04/98 PAGE 2
SYMBOL TABLE LISTING
------ ----- -------
N A M E T Y P E V A L U E A T T R I B U T E S
ACC. . . . D ADDR 00E0H A
```

```
BEG_STK. . D ADDR 0036H A
CA_OFF_4B. C ADDR 01B3H A
CA_OFF_5B. C ADDR 0198H A
CA_OFF_6B. C ADDR 017DH A
CAS_ON_2B. C ADDR 00DEH A
CAS_ON_4B. C ADDR 0109H A
CAS_ON_4C. C ADDR 0114H A
CASO_OFF_1 C ADDR 01DEH A
CASO_OFF_2 C ADDR 01CEH A
CASO_OFF_3 C ADDR 01BEH A
CASO_OFF_4 C ADDR 01A3H A
CASO_OFF_5 C ADDR 0188H A
CASO_OFF_6 C ADDR 016DH A
CASO_ON_0. C ADDR 00AEH A
CASO_ON_1. C ADDR 00BEH A
CASO_ON_2. C ADDR 00CEH A
CASO_ON_3. C ADDR 00E9H A
CASO_ON_4. C ADDR 00F9H A
CASO_ON_5. C ADDR 011FH A
CNT_CHAR . D ADDR 0032H A
CNT_ER_DEB D ADDR 0030H A
CNT_SHIFT. D ADDR 0034H A
CNT_WD . . D ADDR 0035H A
COM_1. . . C ADDR 0234H A
COM_2. . . C ADDR 025FH A
COM_3. . . C ADDR 028CH A
COM_4. . . C ADDR 02BBH A
CW_CHAR. . D ADDR 0033H A
CW_MSG . . C ADDR 033AH A
DE_INT1. . C ADDR 03E5H A
DE_OFF_ON. C ADDR 0086H A
DE_ON_OFF. C ADDR 0145H A
DE_RUN_OFF C ADDR 030AH A
DE_RUN_ON. C ADDR 02CDH A
DEL_180S . C ADDR 0262H A
DEL_1S . . C ADDR 020CH A
DEL_1SSP . C ADDR 01F8H A
DEL_230S . C ADDR 0291H A
DEL_5S . . C ADDR 0237H A
DIVER1 . . C ADDR 03B0H A
DPH. . . . D ADDR 0083H A
DPL. . . . D ADDR 0082H A
EA . . . . B ADDR 00A8H.7 A
END_RAMINT NUMB 007FH A
END_STK. . NUMB 007FH A
ET1. . . . B ADDR 00A8H.3 A
EX0. . . . B ADDR 00A8H.0 A
EX1. . . . B ADDR 00A8H.2 A
F_DEB. . . B ADDR 0020H.2 A
FAN_PA . . B ADDR 0090H.2 A
FIN_INT0 . C ADDR 03CDH A
FINE_INT1. C ADDR 0466H A
FLAG_GEN . D ADDR 0020H A
FLAG0. . . B ADDR 0088H.5 A
FLAG1. . . B ADDR 0088H.7 A
GRID1. . . B ADDR 0090H.5 A
GRID2. . . B ADDR 0090H.4 A
HEATHER. . B ADDR 0090H.3 A
HT1. . . . B ADDR 0090H.7 A
HT2. . . . B ADDR 0090H.6 A
IE1. . . . B ADDR 0088H.3 A
INIT . . . C ADDR 0030H A
INT1 . . . B ADDR 00B0H.3 A
Pagina 11
tem_pa.lst
IT0. . . . B ADDR 0088H.0 A
IT1. . . . B ADDR 0088H.2 A
JMP_TABLE. C ADDR 0411H A
LOOP1. . . C ADDR 0264H A
LOOP2. . . C ADDR 0266H A
LOOP3. . . C ADDR 020EH A
LOOP4. . . C ADDR 0293H A
LOOP5. . . C ADDR 0295H A
LOOP6. . . C ADDR 0239H A
LOOP8. . . C ADDR 01FAH A
MAIN . . . C ADDR 0076H A
MASTER . . B ADDR 00B0H.4 A
N_STATO. . D ADDR 0031H A
NUM_DEB. . NUMB 0019H A
OFFSET_KO1 C ADDR 013EH A
ON_OFF0. . B ADDR 0088H.4 A
ON_OFF1. . B ADDR 0088H.6 A
OUT_CW . . B ADDR 00B0H.1 A
P1 . . . . D ADDR 0090H A
PA_OFF . . B ADDR 0090H.1 A
PA_ON. . . B ADDR 0090H.0 A
PASS_1A. . C ADDR 02E6H A
PASS_1B. . C ADDR 0323H A
PASS_2A. . C ADDR 02F0H A
PASS_2B. . C ADDR 032DH A
PRO_OFF_ON C ADDR 01E6H A
PRO_ON_OFF C ADDR 0127H A
PWR_FAIL . B ADDR 0020H.4 A
RES_FAIL . C ADDR 035AH A
RTI. . . . C ADDR 0348H A
RUN_CHANGE B ADDR 0020H.3 A
RUN_DE_OFF C ADDR 02FDH A
RUN_DE_ON. C ADDR 02C0H A
SOLO_SH. . C ADDR 0377H A
SP . . . . D ADDR 0081H A
STATO. . . B ADDR 0020H.1 A
STEP_1A. . C ADDR 00A0H A
STEP_1B. . C ADDR 015FH A
STEP_2A. . C ADDR 00AAH A
```

```
STEP_2B. . C ADDR 0169H A
STESSO . . C ADDR 0393H A
STINT1 . . C ADDR 03FFH A
STINT2 . . C ADDR 0409H A
T0 . . . . B ADDR 00B0H.4 A
T1 . . . . B ADDR 00B0H.5 A
TAB_OFF_ON C ADDR 01ECH A
TAB_ON_OFF C ADDR 012DH A
TASK_INT0. C ADDR 03A0H A
TASK_INT1. C ADDR 03D0H A
TEST1. . . C ADDR 041FH A
TEST2. . . C ADDR 0428H A
TEST3. . . C ADDR 0431H A
TEST4. . . C ADDR 043AH A
TEST5. . . C ADDR 0443H A
TEST6. . . C ADDR 044CH A
TEST7. . . C ADDR 0455H A
TF0. . . . B ADDR 0088H.5 A
TF1. . . . B ADDR 0088H.7 A
TH0. . . . D ADDR 008CH A
TH1. . . . D ADDR 008DH A
TL0. . . . D ADDR 008AH A
TL1. . . . D ADDR 008BH A
TMOD . . . D ADDR 0089H A
TO_OFF . . C ADDR 0139H A
TO_ON. . . C ADDR 007CH A
TORNA. . . C ADDR 02F6H A
TORNB. . . C ADDR 0333H A
TR0. . . . B ADDR 0088H.4 A
TR1. . . . B ADDR 0088H.6 A
TST_DIV2 . C ADDR 03ABH A
TST_OFF1 . C ADDR 0220H A
TST_OFF2 . C ADDR 024BH A
TST_OFF3 . C ADDR 0278H A
TST_OFF4 . C ADDR 02A7H A
TST_ON1. . C ADDR 022AH A
TST_ON2. . C ADDR 0255H A
TST_ON3. . C ADDR 0282H A
TST_ON4. . C ADDR 02B1H A
TX_CODE. . C ADDR 036AH A
TXD. . . . B ADDR 00B0H.1 A
W_DOG. . . C ADDR 038BH A
REGISTER BANK(S) USED: 0
ASSEMBLY COMPLETE, NO ERRORS FOUND
```