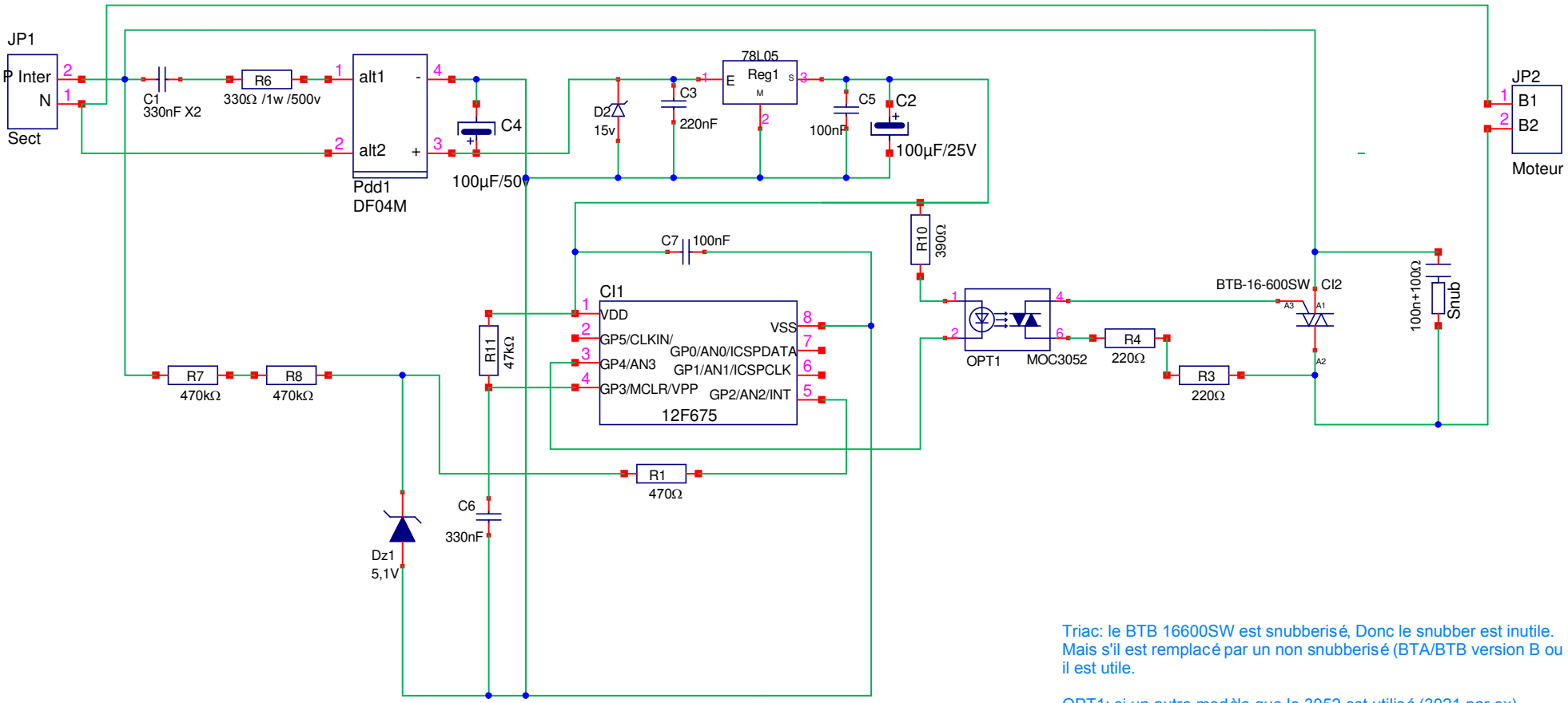
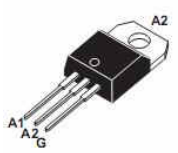


Triac: le BTB 16600SW est snubberisé, Donc le snubber est inutile.
Mais s'il est remplacé par un non snubberisé (BTA/BTB version B ou C), il est utile.

OPT1: si un autre modèle que le 3052 est utilisé (3021 par ex)
la résistance R10 doit être remplacée avec une valeur correspondante au courant nominal de la diode (220Ωpour le 3021)

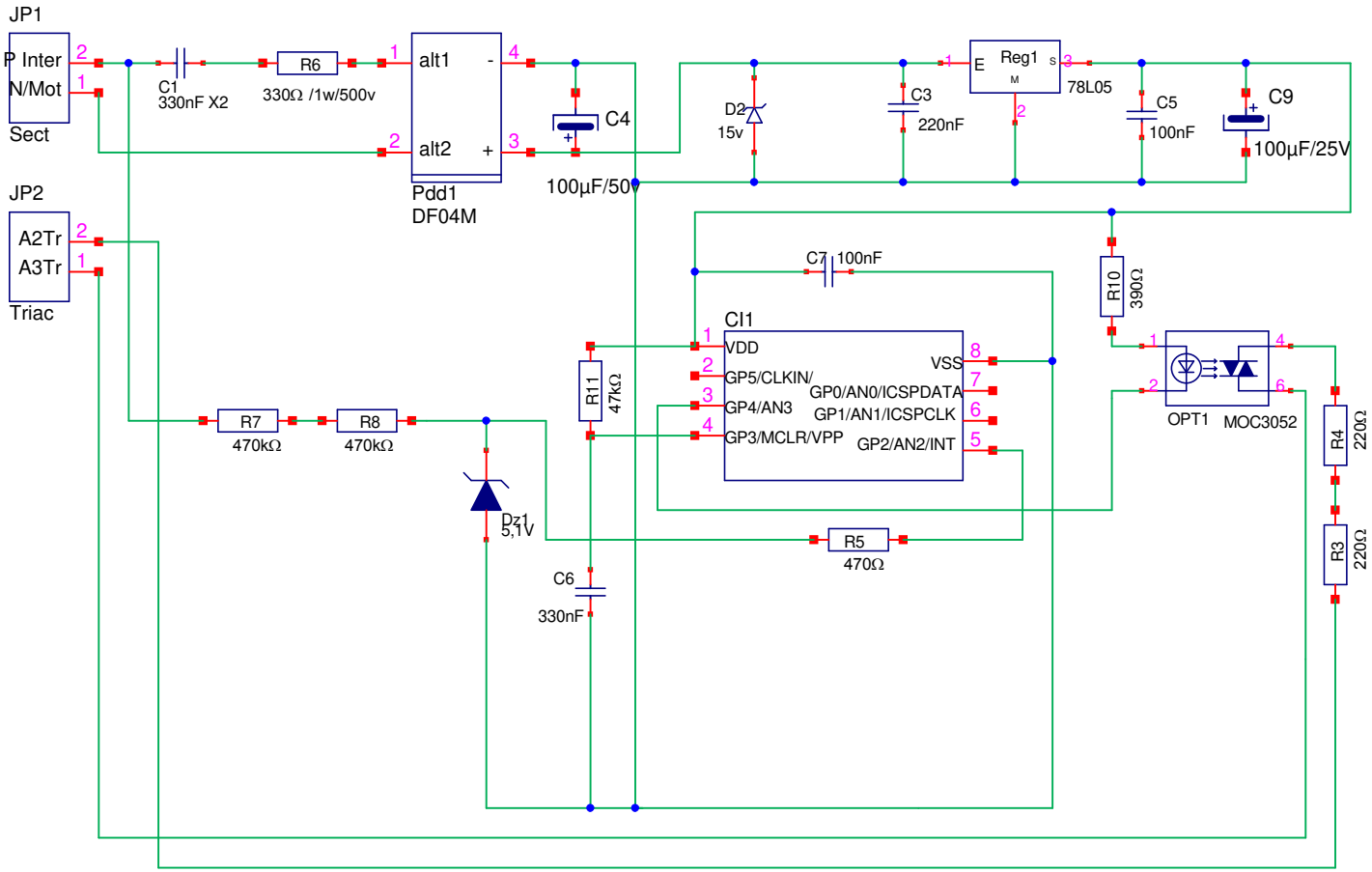
78L05

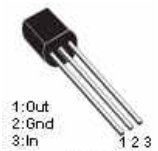


Triac BTB 16

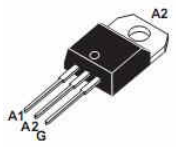
Les résistances en 1206 tolèrent maximum 200V.
2 en série tolèrent 400V

Soft Start 230v~
Principe
06/10/2015





78L05



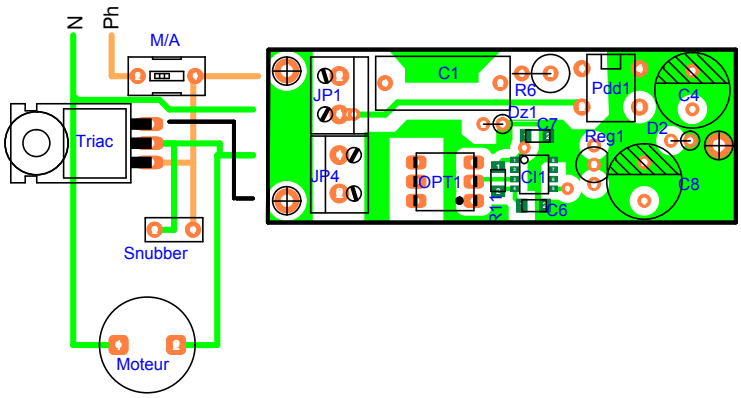
Triac BTB 16

Les résistances en 1206 tolèrent maximum 200V.
2 en série tolèrent 400V

Soft Start 230v~

Typon

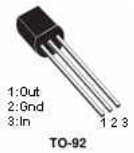
06/10/2015



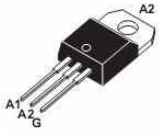
Composant	Valeur	Type
C1	330nF X2	6x2.54
C3	220nF	CMS 1206
C4	100µF/25v	radial 2x2.54
C5	100nF	CMS 1206
C6	330nF	CMS 1206
C7	100nF	CMS 1206
C8	100µF/25V	radial 2x2.54
CI1	12F675	SOIC 08
D2	15v/0.5W	Zener
Dz1	5,1V/0.5W	Zener
JP1	Sect	1x5.08
JP4	Triac	1x5.08
OPT1	MOC3052	DIL 06
Pdd1	DF04M	DIL 04
R2	470Ω	CMS 1206
R3	220Ω	CMS 1206
R4	220Ω	CMS 1206
R6	330Ω/1w	4x2.54
R7	470kΩ	CMS 1206
R8	470kΩ	CMS 1206
R10	390Ω	CMS 1206
R11	47kΩ	CMS 1206
Reg1	78L05	TO92
Snubber	100nF+100Ω	5x2.54
Triac	BTB 16 600SW	Triac TO220

Triac: le BTB 16600SW est snubberisé,
donc le snubber est inutile,
mais si vous utilisez un triac non snubérisé,
il est utile.

OPT1: si un autre modèle est utilisé, par ex le 3021,
il faut remplacer R10 par une 220Ω
pour que le courant dans la diode soit correct

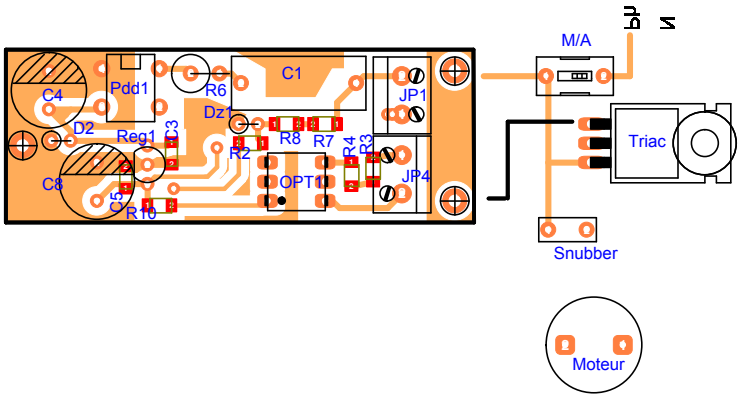


78L05



Triac

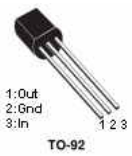
Soft Start 230V
Typon_5 mini
Jacques Menut
10/11/2015



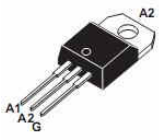
Composant	Valeur	Type
C1	330nF X2	6x2.54
C3	220nF	CMS 1206
C4	100µF/25v	radial 2x2.54
C5	100nF	CMS 1206
C6	330nF	CMS 1206
C7	100nF	CMS 1206
C8	100µF/25V	radial 2x2.54
CI1	12F675	SOIC 08
D2	15v/0.5W	Zener
Dz1	5,1V/0.5W	Zener
JP1	Sect	1x5.08
JP4	Triac	1x5.08
OPT1	MOC3052	DIL 06
Pdd1	DF04M	DIL 04
R2	470Ω	CMS 1206
R3	220Ω	CMS 1206
R4	220Ω	CMS 1206
R6	330Ω/1w	4x2.54
R7	470kΩ	CMS 1206
R8	470kΩ	CMS 1206
R10	390Ω	CMS 1206
R11	47kΩ	CMS 1206
Reg1	78L05	TO92
Snubber	100nF+100Ω	5x2.54
Triac	BTB 16 600SW	Triac TO220

Triac: le BTB 16600SW est snubberisé,
donc le snubber est inutile,
mais si vous utilisez un triac non snubérisé,
il est utile.

OPT1: si un autre modèle est utilisé, par ex le 3021,
il faut remplacer R10 par une 220Ω
pour que le courant dans la diode soit correct

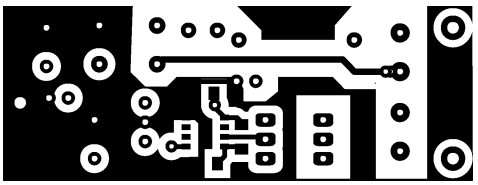


78L05

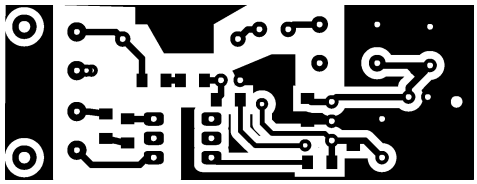


Triac

Soft Start 230V
Typon_5 mini
Jacques Menut
10/11/2015



face composants



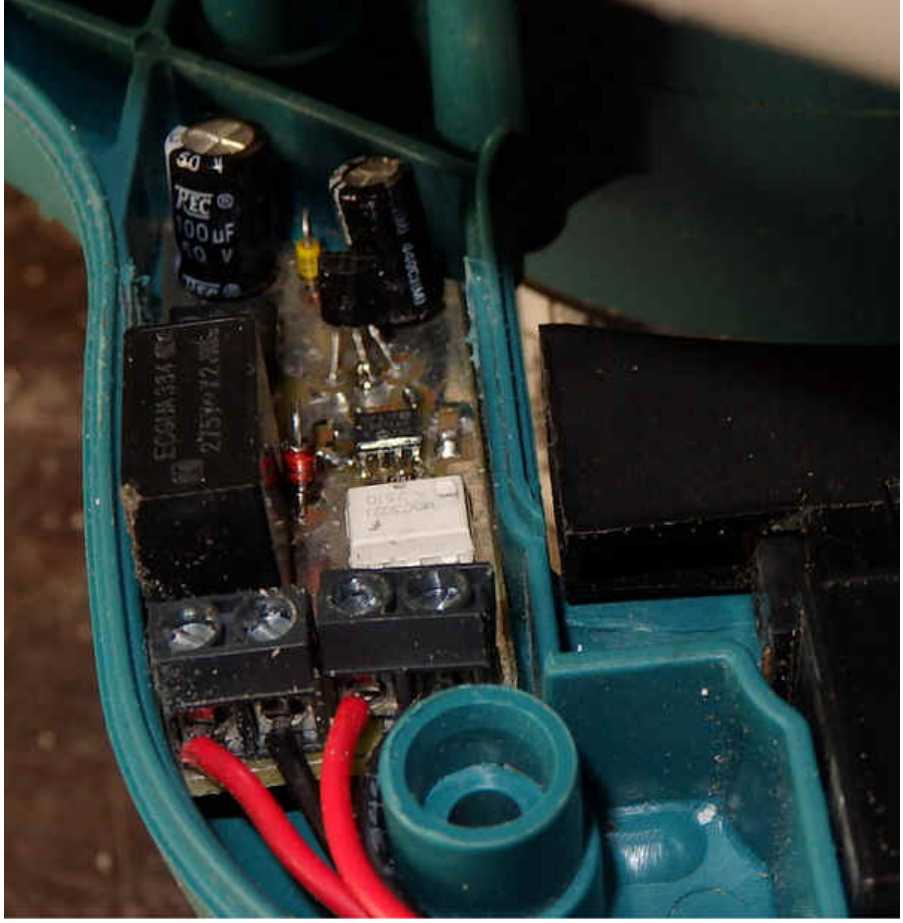
face cuivre

Composant	Valeur	Type	Fournisseur	Prix
C1	330nF X2	6x2.54	Conrad 1566338	0.55 €
C3	220nF	CMS 1206	Gotronic 16416	0.25 €
C4	100µF/25v	radial 2x2.54	Gotronic 22710	0.20 €
C5	100nF	CMS 1206	Gotronic 16415	0.18 €
C6	330nF	CMS 1206	RS 6983598	0.55 €
C7	100nF	CMS 1206	Gotronic 16415	0.18 €
C8	100µF/25V	radial 2x2.54	Gotronic 22710	0.20 €
CI1	12F675	SOIC 08	RS 5441709	1.00 €
D2	15v/0.5W	Zener	Gotronic 02553	0.10 €
Dz1	5,1V/0.5W	Zener	Gotronic 02542	0.15 €
JP1	Sect	1x5.08	Gotronic 49112	0.30 €
JP4	Triac	1x5.08	Gotronic 49112	0.30 €
OPT1	MOC3052	DIL 06	RS 6912293	0.62 €
Pdd1	DF04M	DIL 04	Gotronic 02636	0.70 €
R2	470Ω	CMS 1206	Gotronic 16233	0.03 €
R3	220Ω	CMS 1206	Gotronic 16229	0.03 €
R4	220Ω	CMS 1206	Gotronic 16229	0.03 €
R6	330Ω/1w	4x2.54	Gotronic 04230	0.10 €
R7	470kΩ	CMS 1206	Gotronic 16269	0.03 €
R8	470kΩ	CMS 1206	Gotronic 16269	0.03 €
R10	390Ω	CMS 1206	Gotronic 16232	0.03 €
R11	47kΩ	CMS 1206	Gotronic 16257	0.03 €
Reg1	78L05	TO92	Gotronic 02460	0.40 €
Snubber	100nF+100Ω	5x2.54	RS 2067881	2.65 €
Triac	BTB 16 600SW	Triac TO220	RS 8292709	1.50 €
			Total	10.14 €

```
 1: //
 2: //Commande soft start moteur universel 230V
 3: //
 4: //Auteur: Jacques Menut
 5: //
 6: //Créée le: 09/05/2015
 7: //Modifié le: 06/10/2015
 8: //
 9: //
10: //
11: //En faisant varier progressivement le temps de conduction du triac, le moteur
12: //démarre aussi progressivement.
13: //
14: //version:  2.1:  valeur retard tcp et tpc définies à l'initialisation
15: //                alternance non continue
16: //                retard initial plus grand pour gros moteur
17: //
18: //
19: //
20: //
21:
22:
23: #include <12F675.h>
24:
25:
26: #fuses INTRC_IO, NOWDT, PUT, MCLR, NOPROTECT, NOCPD
27: #use delay(clock=4000000)
28:
29: #define  TRIAC PIN_A4      //commande du triac
30: #define  SECT PIN_A2       //entrée secteur, interruption INT_EXT
31:
32:
33: short inter, phase, int_phase, r, sleep_mode;
34: // inter: flag interruption compararteur //phase: flag interruption changement de phase
35: // r: flag inutile!
36:
37:
38: int  palt, value;    //palt: premiers passages int_timer0 et int_ext
39: int16 tcp, tpc;
40: //    tcp: "temps" entre détection phase et déclenchement triac
41: //    tpc: "temps" entre détection phase et déclenchement triac
42:
43:
44: #INT_EXT
45: void interupt_zero()    //interruption au passage à zero
46: {
47:    if (tpc < 100)
48:       goto fintext;
49:    if (!palt)              // si démarrage, ou après sleep
50:       {
51:       sleep_mode=FALSE;
52:       palt = 1;
53:       if (input_state(PIN_A2))      //si phase positive
54:          {
55:          phase = true;
56:          ext_int_edge(H_TO_L);
57:          }
58:       else                          //si phase négative
59:          {
60:          phase = false;
61:          ext_int_edge(L_TO_H);
62:          }
63:       disable_interrupts(INT_TIMER0);
64:       goto fintext;
65:       }
66:
67:    if (phase)
68:       {
69:       phase = false;
70:       tcp = tcp - 80;
71:       ext_int_edge(L_TO_H);
72:       goto fintext;
73:       }
74:
```

```
 75:    if (!phase)
 76:       {
 77:       phase = true;
 78:       tpc = tpc - 80;
 79:       ext_int_edge(H_TO_L);
 80:       }
 81: fintext:
 82:    int_phase = true;
 83:    clear_interrupt(INT_EXT);
 84: }
 85:
 86: #INT_TIMER0                        //uniquement au démarrage si pas d'interruption
 87: void interupt_timer0()             //int_ext pendant ~2ms
 88: {
 89:    if (input_state(PIN_A2))        //si phase positive
 90:       ext_int_edge(H_TO_L);
 91:    else
 92:       ext_int_edge(L_TO_H);
 93:    clear_interrupt(INT_TIMER0);
 94: }
 95:
 96:
 97: void initialise()
 98: {
 99:    set_tris_a(0b000100);       //ports 0,1,3,4,5 en sortie; ports 2 en entrée
100:    port_a_pullups(FALSE);      //résistances pullup inactives
101:    setup_adc(ADC_OFF);         //convertisseur AN inactif
102:    disable_interrupts(INT_RA4);      //désactive      RA4      "
103:    disable_interrupts(INT_RA5);      // "        RA5            "
104:    output_high (TRIAC);
105:    disable_interrupts(INT_TIMER1);  // "        timer1         "
106:    setup_timer_0(T0_INTERNAL | T0_DIV_8);    //configure timer0, déborde à 2,048ms
107:    setup_adc_ports(NO_ANALOGS);
108:    setup_comparator(A0_A1);
109:    ext_int_edge(H_TO_L);         //interruption INT 1-->0
110: }
111:
112:
113: void main()
114: {
115:    initialise();
116:    phase = 0;
117:    inter = 0;
118:    int_phase = 0;
119: start:
120:    output_high (TRIAC);    //bloque le triac
121:    tpc = 5800;
122:    tcp = 5800;
123:    palt = 0;
124: debut:
125:    int_phase = false;
126:    clear_interrupt(INT_EXT);
127:    output_high(TRIAC);              //bloque le triac
128:    enable_interrupts(INT_TIMER0);   //active interruption timer0
129:    enable_interrupts(INT_EXT);      //active interruption INT
130:    enable_interrupts(GLOBAL);
131:
132:    while (tpc > 80)     //tant que tpc > 80
133:    {
134: debut1:
135:       while (!palt)
136:          {r = 0;}
137:
138:       if (phase)
139:          {
140:          int_phase = false;
141:          delay_us(tpc);
142:
143:          output_low(TRIAC);      //active triac
144:          delay_us(100);
145:          output_high(TRIAC);     //désactive triac
146:          delay_us(50);
147:          output_low(TRIAC);
148:          delay_us(100);
```

```
149:          output_high(TRIAC);
150:
151:          }
152:          if (!phase)
153:             {
154:             int_phase = false;
155:             delay_us(tcp);
156:             output_low(TRIAC);
157:             delay_us(100);
158:             output_high(TRIAC);
159:             delay_us(50);
160:             output_low(TRIAC);
161:             delay_us(100);
162:             output_high(TRIAC);
163:             }
164:
165:          while (!int_phase)
166:             {r = 0;}
167:    }
168:    output_low(TRIAC);            //active triac
169:    int_phase = false;
170:    disable_interrupts(INT_EXT);
171:    enable_interrupts(GLOBAL);
172:
173:    while(r==0)       //après phase démarrage
174:    {
175:
176:    delay_ms(200);               //tempo 200ms
177:    enable_interrupts(INT_EXT);   //réactive int_ext
178:    delay_ms(10);                 //delai 10ms pour tester si interruption int_ext
179:    if (int_phase)              //si interruption INT_EXT (début phase)
180:       {
181:       disable_interrupts(INT_EXT);     //désactive int_ext pdt 200ms
182:       int_phase = false;               //RaZ flag INT_EXT
183:       }
184:    else
185:       {
186:       disable_interrupts(GLOBAL);   //si pas interruption INT_EXT (inter sur arrêt)
187:       output_high(TRIAC);           //désactive triac
188:       goto fin;                     //passe en veille
189:       }
190:    }
191:
192: fin:
193:    sleep_mode=TRUE;
194:    phase = 0;
195:    int_phase = 0;
196:    inter = 0;
197:    palt = 0;
198:    sleep();                // veille
199:    sleep_mode=FALSE;       // réveil par INT_EXT (inter marche actif)
200:    clear_interrupt(INT_TIMER0);
201:    enable_interrupts(INT_TIMER0);     //
202:    clear_interrupt(INT_EXT);
203:    enable_interrupts(INT_EXT);
204:    enable_interrupts(GLOBAL);
205:
206:    goto start;
207:
208: }
```


PEC
100 uF