



# **SOMMAIRE**

## **1. Le RCM3100 (version américaine)**

- 1.1. RCM3100 Digital Inputs and Outputs**
- 1.2. Electrical and Mechanical Characteristics**
- 1.3. Exclusion Zone**
- 1.4. Headers**
- 1.5. Physical mounting**

## **2. Le PK4PWM**

### **2.1. Structure interne de la carte servant à plugger le RCM3110**

- CI Face composant
- CI Face soudure
- Equipement CI face bottom
- Equipement CI face top
- Plan mécanique du CI

### **2.2. Le boîtier du PK4PWM – PACTEC FLX2535-028**

### **2.3. Caractéristiques du PK4PWM**

### **2.4. Programmation du PK4PWM**

- 2.4.1. Prise en main du PK4PWM**
- 2.4.2. Fonctions utiles du Dynamic C Premier**
- 2.4.3. Exemple typique de programme pour faire fonctionner le PK4PWM**

### **2.5. Schéma électronique du PK4PWM**

# 1. Le RCM3100 (version américaine)

## 1.1 RCM3100 Digital Inputs and Outputs

The RCM3100 has 54 parallel I/O lines grouped in seven 8-bit ports available on headers J1 and J2. The 46 bidirectional I/O lines are located on pins PA0–PA7, PB0, PB2–PB7, PD0–PD7, PE0–PE1, PE3–PE7, PF0–PF7, and PG0–PG7.

Figure 1 shows the RCM3100 RabbitCore series pinouts for headers J1 and J2.

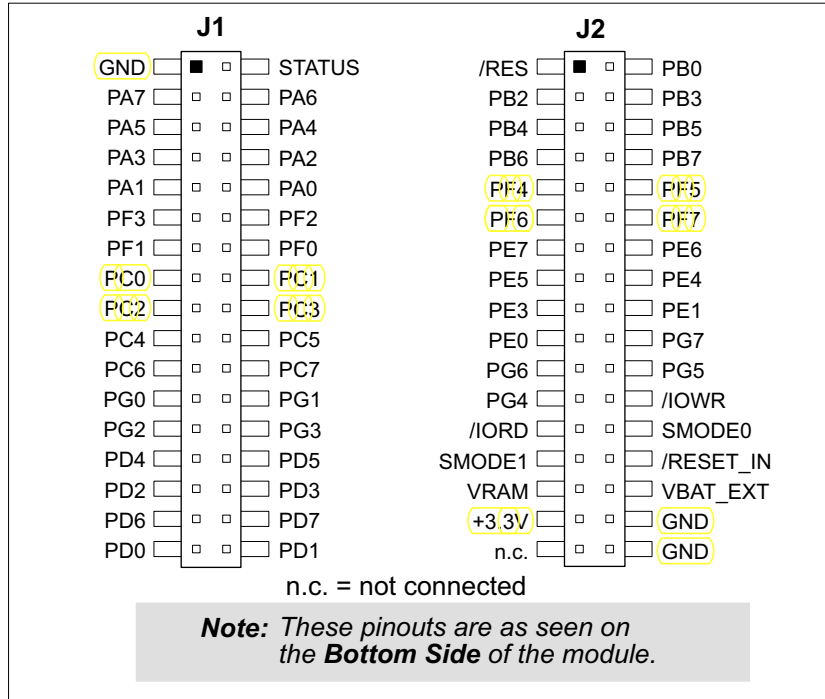
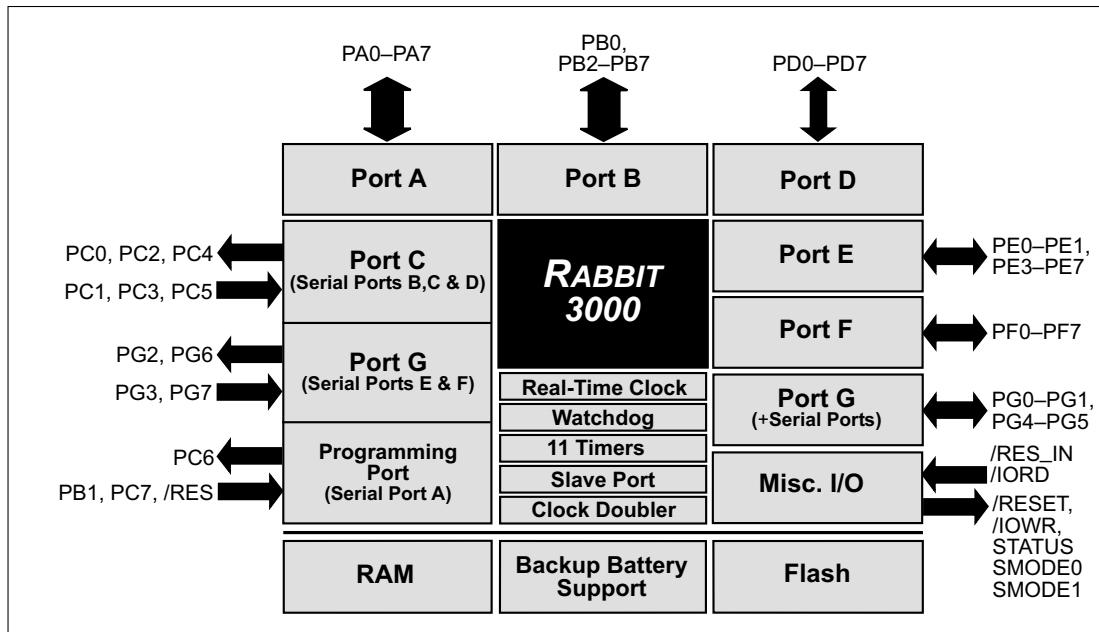


Figure 1. RCM3100 Pinouts

Headers J1 and J2 are standard 2 × 34 headers with a nominal 2 mm pitch.

The signals labeled PD0–PD3, PD6, and PD7 on header J1 (pins 29–34) and the pin that is not connected (pin 33 on header J2) are reserved for future use on other RabbitCore modules.

Figure 2 shows the use of the Rabbit 3000 ports in the RCM3100 series RabbitCore modules.



**Figure 2. Use of Rabbit 3000 Ports**

The ports on the Rabbit 3000 microprocessor used in the RCM3100 Series are configurable, and so the factory defaults can be reconfigured. Table 1 lists the Rabbit 3000 factory defaults and the alternate configurations.

**Table 1. RCM3100 Pinout Configurations**

Pin	Pin Name	Default Use	Alternate Use	Notes
1	GND			
2	STATUS	Output (Status)	Output	
3–10	PA[7:0]	Parallel I/O	External data bus (ID0–ID7) Slave port data bus (SD0–SD7)	
11	PF3	Input/Output	QD2A	
12	PF2	Input/Output	QD2B	
13	PF1	Input/Output	QD1A CLKC	
14	PF0	Input/Output	QD1B CLKD	
15	PC0	Output	TXD	Serial Port D
16	PC1	Input	RXD	
17	PC2	Output	TXC	Serial Port C
18	PC3	Input	RXC	
19	PC4	Output	TXB	Serial Port B
20	PC5	Input	RXB	
21	PC6	Output	TXA	Serial Port A (programming port)
22	PC7	Input	RXA	
23	PG0	Input/Output	TCLKF	Serial Clock F output
24	PG1	Input/Output	RCLKF	Serial Clock F input
25	PG2	Output	TXF	Serial Port F
26	PG3	Input	RXF	
27	PD4	Input/Output	ATXB	
28	PD5	Input/Output	ARXB	
29*	PD2	Input/Output		
30*	PD3	Input/Output		
31*	PD6	Input/Output		
32*	PD7	Input/Output		
33*	PD0	Input/Output		
34*	PD1	Input/Output		

Header J1

\* Pins 29–34 are reserved for future RCM3100 series RabbitCore modules.

**Table 1. RCM3100 Pinout Configurations (continued)**

Pin	Pin Name	Default Use	Alternate Use	Notes	
Header J2	1	/RES	Reset output	Reset input	Reset output from Reset Generator
	2	PB0	Input/Output	CLKB	
	3	PB2	Input/Output	IA0 /SWR	External Address 0 Slave port write
	4	PB3	Input/Output	IA1 /SRD	External Address 1 Slave port read
	5	PB4	Input/Output	IA2 SA0	External Address 2 Slave port Address 0
	6	PB5	Input/Output	IA3 SA1	External Address 3 Slave port Address 1
	7	PB6	Input/Output	IA4	External Address 4
	8	PB7	Input/Output	IA5 /SLAVEATTN	External Address 5 Slave Attention
	9	PF4	Input/Output	AQD1B PWM0	
	10	PF5	Input/Output	AQD1A PWM1	
	11	PF6	Input/Output	AQD2B PWM2	
	12	PF7	Input/Output	AQD2A PWM3	
	13	PE7	Input/Output	I7 /SCS	
	14	PE6	Input/Output	I6	
	15	PE5	Input/Output	I5 INT1B	
	16	PE4	Input/Output	I4 INT0B	
	17	PE3	Input/Output	I3	
	18	PE1	Input/Output	I1 INT1A	I/O Strobe 1 Interrupt 1A
	19	PE0	Input/Output	I0 INT0A	I/O Strobe 0 Interrupt 0A

**Table 1. RCM3100 Pinout Configurations (continued)**

Pin	Pin Name	Default Use	Alternate Use	Notes	
Header J2	20	PG7	Input/Output	RXE	Serial Port E
	21	PG6	Input/Output	TXE	
	22	PG5	Input/Output	RCLKE	Serial Clock E input
	23	PG4	Input/Output	TCLKE	Serial Clock E output
	24	/IOWR	Output		External write strobe
	25	/IORD	Input		External read strobe
	26–27	SMODE0, SMODE1	(0,0)—start executing at address zero (0,1)—cold boot from slave port (1,0)—cold boot from clocked Serial Port A  SMODE0 =1, SMODE1 = 1 Cold boot from asynchronous Serial Port A at 2400 bps (programming cable connected)		Also connected to programming cable
	28	/RESET_IN	Input		Input to Reset Generator
	29	VRAM	Output		Maximum Current Draw 15 $\mu$ A
	30	VBAT_EXT	3 V battery Input		Minimum battery voltage 2.8 V
	31	+3.3V	Input		3.15–3.45 V DC
	32	GND			
	33	n.c.			
	34	GND			



Table 2 lists the electrical, mechanical, and environmental specifications for the RCM3100

**Table 2 - RabbitCore RCM3100 Specifications**

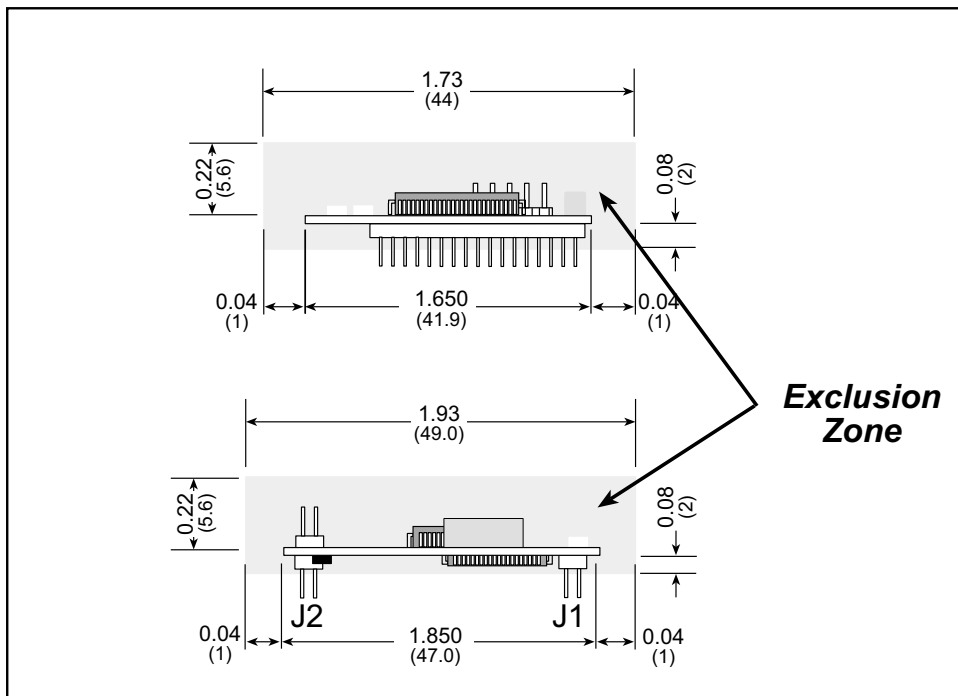
Feature	RCM3100	RCM3110
Microprocessor	Rabbit 3000™ at 29.4 MHz	
EMI Reduction	Spectrum spreader for reduced EMI (radiated emissions)	
Flash Memory	512K (2 × 256K)	256K
SRAM	512K	128K
Backup Battery	Connection for user-supplied backup battery to support RTC and SRAM)	
General-Purpose I/O	54 parallel digital I/O lines: <ul style="list-style-type: none"> <li>• 46 configurable I/O,</li> <li>• 4 fixed inputs,</li> <li>• 4 fixed outputs</li> </ul>	
Additional Digital Inputs	2 startup mode, reset in	
Additional Digital Outputs	Status, reset out	
Auxiliary I/O Bus	8 data lines and 6 address lines (shared with I/O) plus I/O read/write	
Serial Ports	6 shared high-speed, CMOS-compatible ports: <ul style="list-style-type: none"> <li>• 6 configurable as asynchronous (with IrDA), 4 as clocked serial(SPI), and 2 as SDLC/HDLC (with IrDA)</li> <li>• 1 asynchronous clocked serial port dedicated for programming</li> <li>• support for MIR/SIR IrDA transceiver</li> </ul>	
Serial Rate	Max. asynchronous baud rate = CLK/8	
Slave Interface	A slave port allows the RCM3100 to be used as a master or as an intelligent peripheral device with Rabbit-based or any other type of processor	
Real-Time Clock	Yes	
Timers	Ten 8-bit timers (6 cascadable from the first), one 10-bit timer with 2 match registers	
Watchdog/Supervisor	Yes	
Pulse-Width Modulators	10-bit free-running counter and four pulse-width registers	
Input Capture	2- channel input capture can be used to time input signals from various port pins	
Quadrature Decoder	2-channel quadrature decoder accepts inputs from external incremental encoder modules	
Power	3.15 V to 3.45 V DC 75 mA @ 3.3 V	

**Table 2. RabbitCore RCM3100 Specifications (continued)**

Feature	RCM3100	RCM3110
Operating Temperature	-40°C to +85°C	
Humidity	5% to 95%, noncondensing	
Connectors (for connection to headers J4 and J5)	Two 2 × 17, 2 mm pitch	
Board Size	1.850" × 2.725" × 0.86" (47 mm × 69 mm × 22 mm)	

### 1.3 Exclusion Zone

It is recommended that you allow for an “exclusion zone” of 0.04" (1 mm) around the RCM3100 in all directions when the RCM3100 is incorporated into an assembly that includes other printed circuit boards. This “exclusion zone” that you keep free of other components and boards will allow for sufficient air flow, and will help to minimize any electrical or electromagnetic interference between adjacent boards. An “exclusion zone” of 0.08" (2 mm) is recommended below the RCM3100 when the RCM3100 is plugged into another assembly using the shortest connectors for headers J1 and J2. Figure 4 shows this “exclusion zone.”

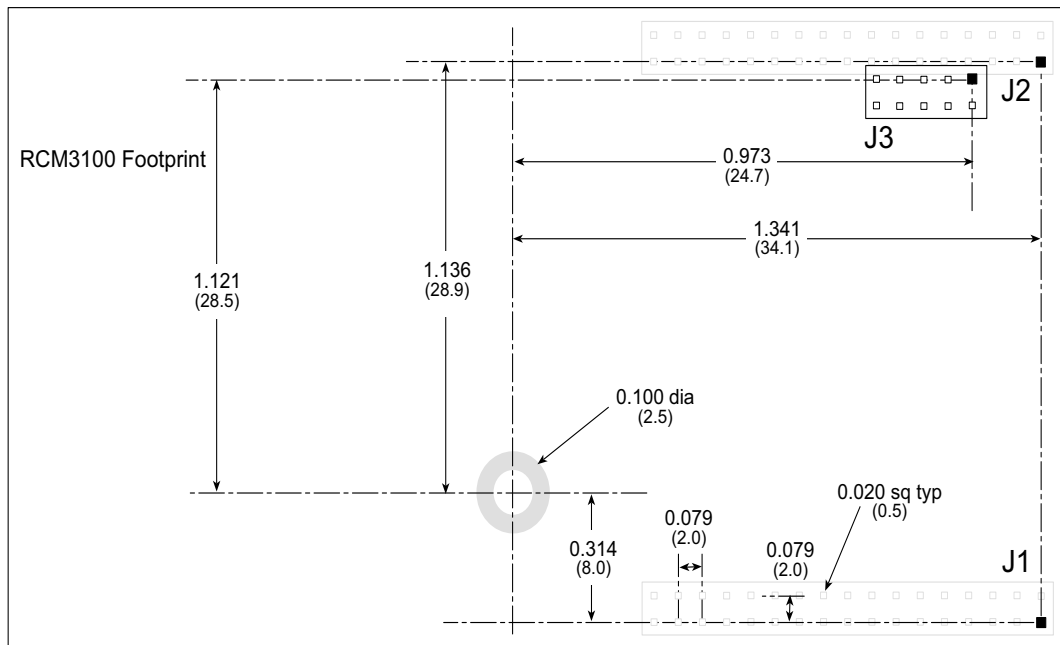


**Figure 4. RCM3100 “Exclusion Zone”**

## 1.4 Headers

The RCM3100 uses headers at J1 and J2 for physical connection to other boards. J1 and J2 are  $2 \times 17$  SMT headers with a 2 mm pin spacing. J3, the programming port, is a  $2 \times 5$  header with a 2 mm pin spacing.

Figure 5 shows the layout of another board for the RCM3100 to be plugged into. These values are relative to the mounting hole.



**Figure 5. User Board Footprint for RCM3100**

## 1.5 Physical Mounting

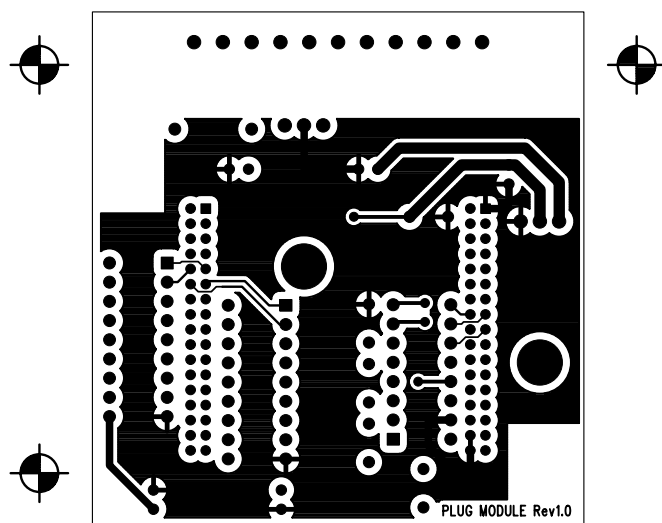
A standoff with a 2-56 screw is recommended to attach the RCM3100 to a user board at the hole position shown in Figure 5.

## 2. Le PK4PWM

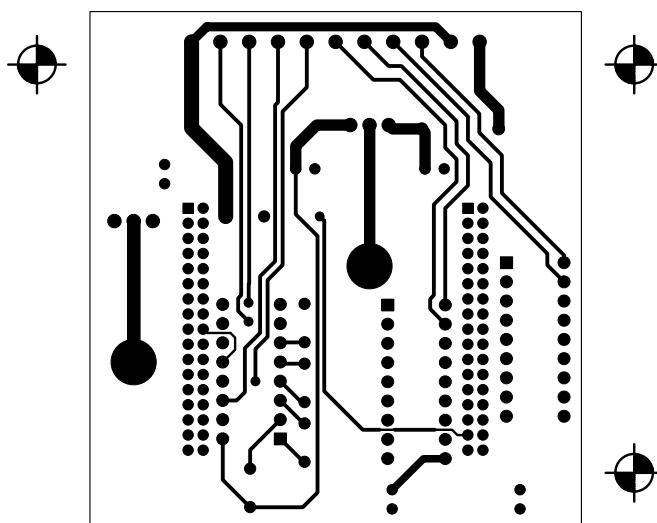
### 2.1. Structure interne de la carte servant à plugger le RCM3110

- CI Face composant
- CI Face soudure
- Equipement CI face bottom
- Equipement CI face top
- Plan mécanique du CI

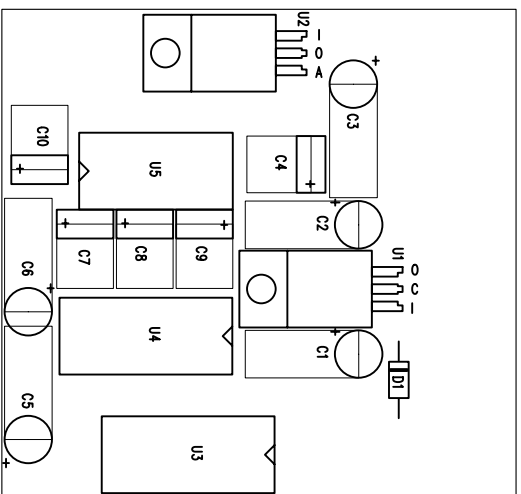
### 2.2. Le boîtier du PK4PWM - PACTEC FLX2535-028



FILM: COMPOSANT	ARTEMIS
DOSSIER: PLUG MODULE Rev1.0	05/03/03



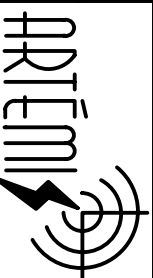
FILM: SOUDURE	ARTEMIS
DOSSIER: PLUG MODULE Rev1	05/03/03



**EQUIPEMENT COTE SOUDURE**

**CARTE PLUG MODULE**

DESSINATEUR	DATE	INDICE	FOLIO
MASSIOT JF	05/03/03	Rev1	2/3

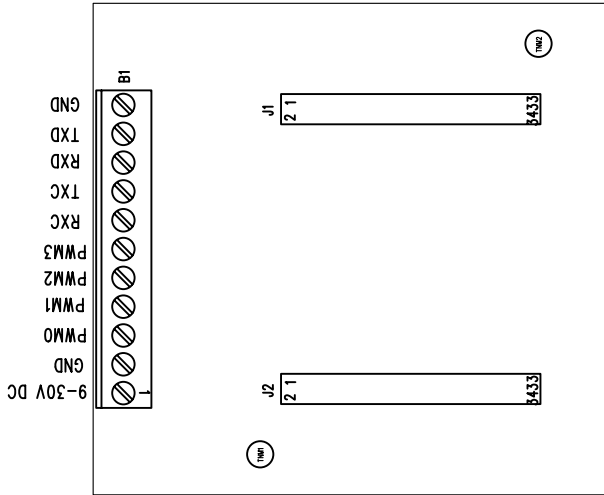


Angers Technopole  
 3, Avenue du Bois l'Abbe  
 49070 BEAUCOUZE  
 Tél : 02 41 48 41 40  
 FAX : 02 41 48 41 44  
 Email : ortemiscod@adl.com

1 2 3 4 5 6

User's manual

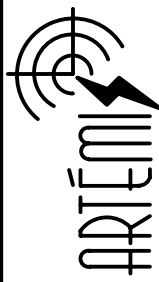
13



EQUIPEMENT COTE COMPOSANT

CARTE PLUG MODULE

DESSINATEUR	DATE	INDICE	FOLIO
MASSIOT JF	05/03/03	Rev1	1/3

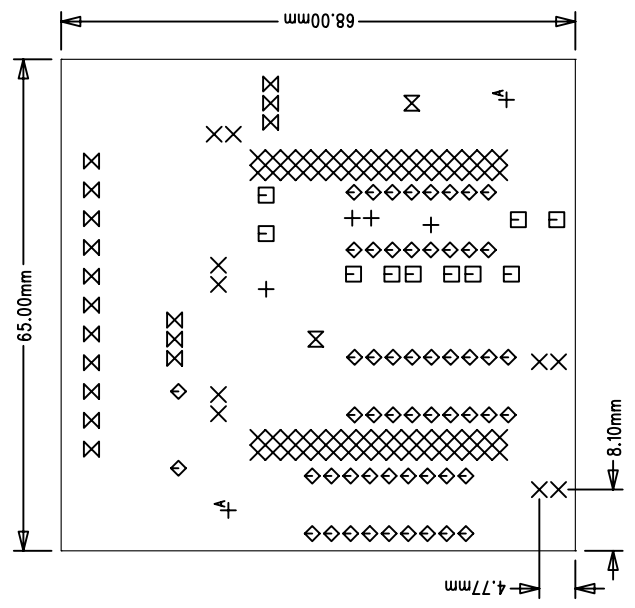


Angers Technopole  
 3, Avenue du Bois l'Abbe  
 49070 BEAUCOUZE  
 Tel : 02 41 48 41 40  
 FAX : 02 41 48 41 44  
 Email : artemiscad@aol.com

1 2 3 4 5 6

1 2 3 4 5 6

SPECIFICATION CIRCUIT IMPRIME  
(Suivant norme :NF C 93-713)



SIZE	QTY	SYM	PLTD
0.7	4	+	PLTD
0.8	78	X	PLTD
0.9	10	□	PLTD
1	54	◇	PLTD
1.2	17	⊗	PLTD
3.2	2	⊗	PLTD
3.5	2	A	NPLTD

Référence du Circuit : **PLUG MODULE Rev1**

Cl. Unitaire : Dimensions du circuit : 68.00 x 65.00 en mm  
 Cl. En Flan de : **XX** Dimensions du flan : XXX.X x XXX.X en mm

Nbre couche : **2**  : Trou Mét. Epaisseur C.I. (mm) **16/10**

Classe: **3**

EPAISSEUR Cu 35u 70u XXu VERNIS : FC FS  
 (fin) Faces ext.    Encre    
 Faces int.    Ph.Imageable    
  Pelable

FINITION : MATIERE : DORURE :  
 SnPb refondu  CEM1  Standard   
 SnPb selectif  FR2  Ni/Au Chimique   
 Cu passive  FR4

SERIGRAPHIE : FC  BLANCHE  CARBONE   
 FS  NOIRE

Test électrique :  COMPARAISON  DONNEES NUMERIQUES  
 Impédance contrôlée :  XX OHMS COUCHE INTX  
 Vias laser :  ENTRE XXXXXXX ET XXXXXX

Angers Technopole  
3, Avenue du Bois l'Abbé  
49070 BEAUCOUZE  
Tél : 02 41 48 41 40  
FAX : 02 41 48 41 44  
Email : artemiscad@aol.com

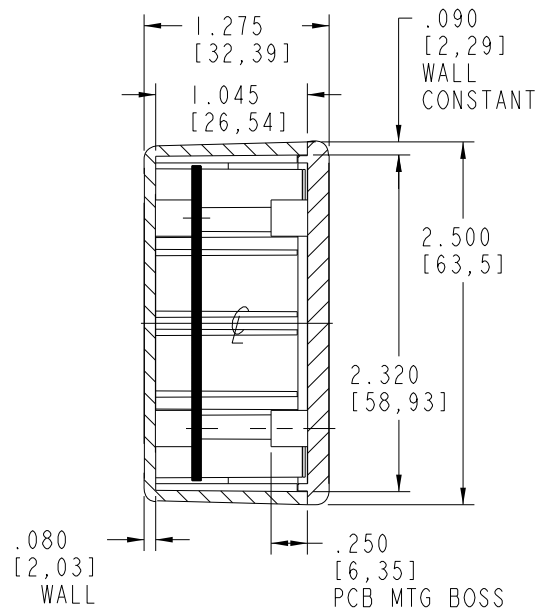
**PLAN DE PERCAGE ET DECOUPE**

**CARTE PLUG MODULE**

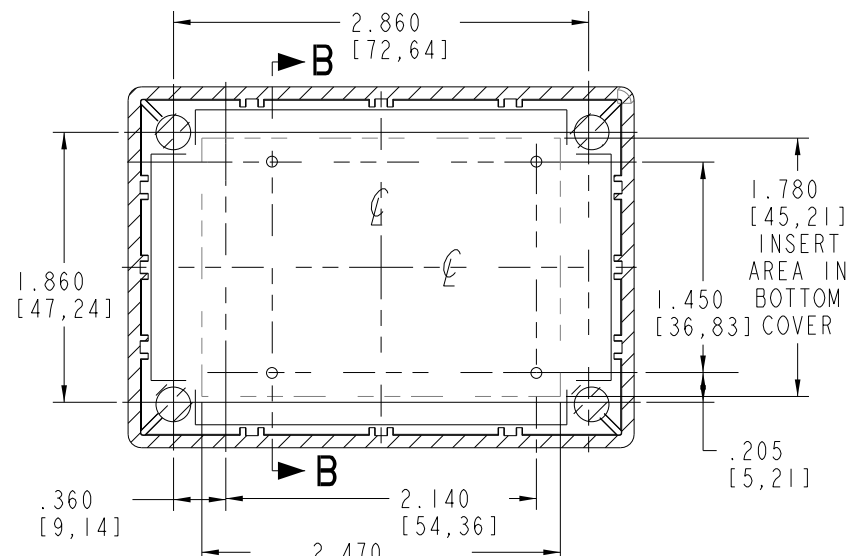
DESSINATEUR	DATE	INDICE	FOLIO
MASSIOT JF	05/03/03	Rev1	3/3

1 2 3 4 5 6

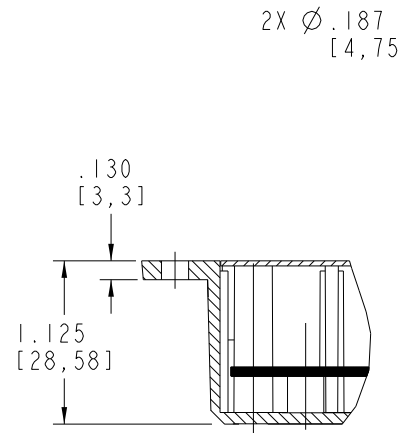
PADS FORMAT DIN A4



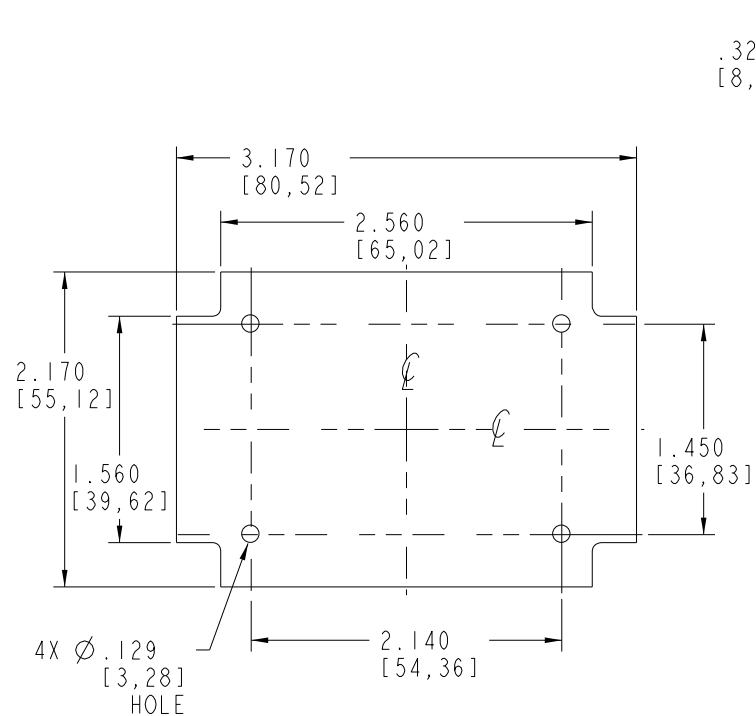
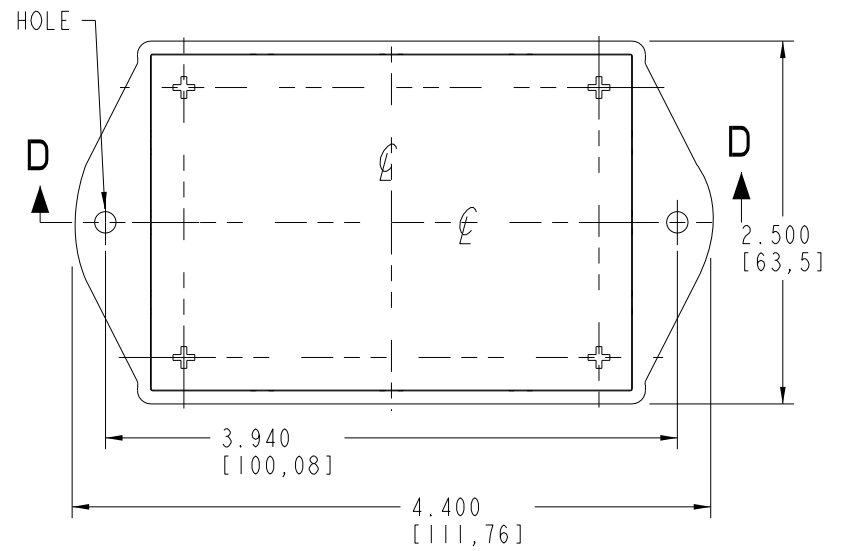
**SECTION B-B**



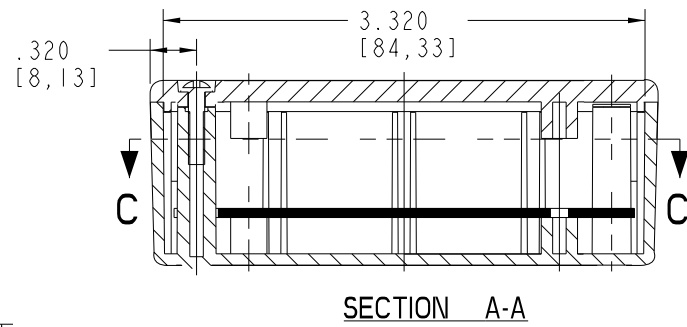
**SECTION C-C**



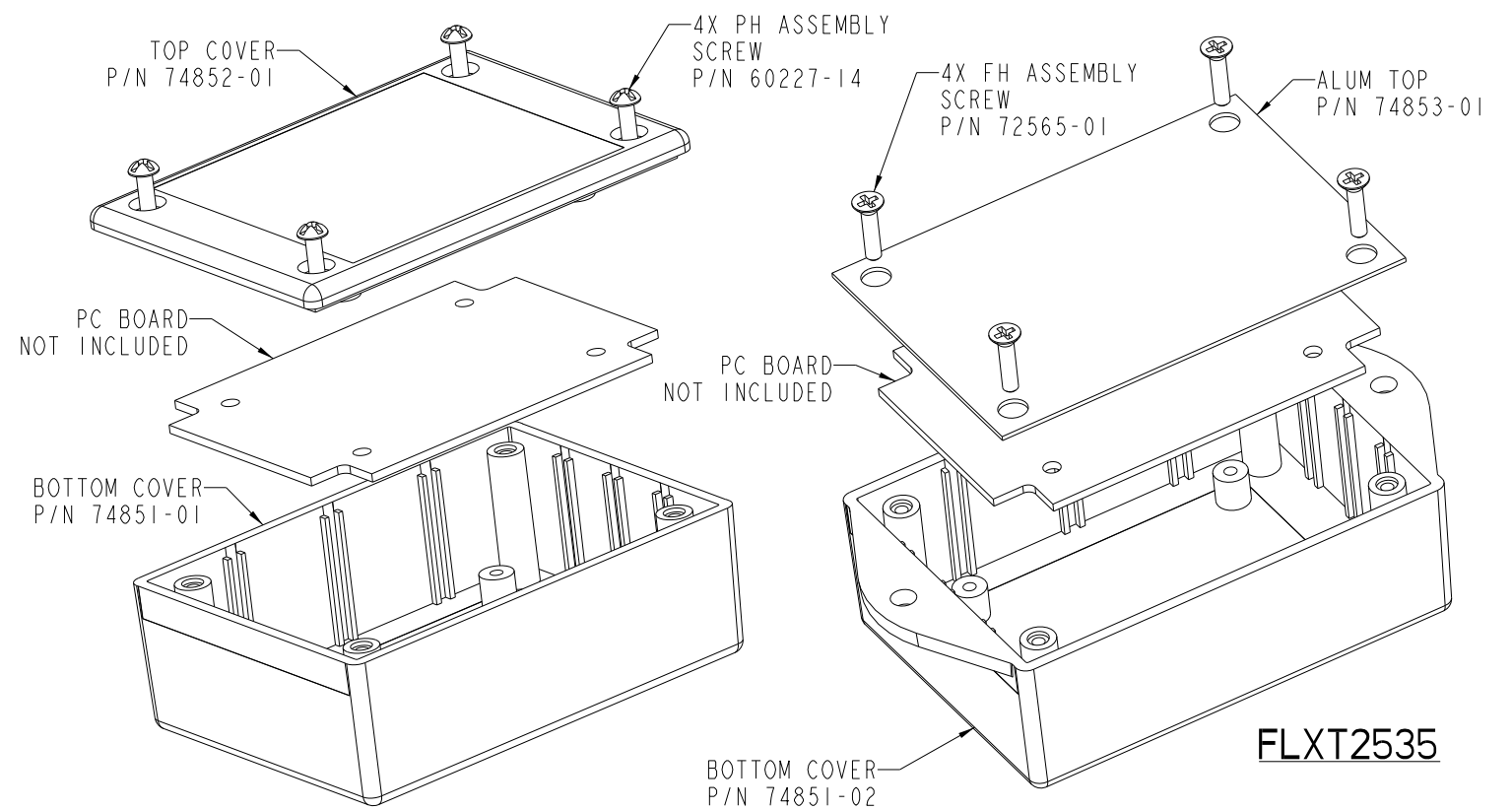
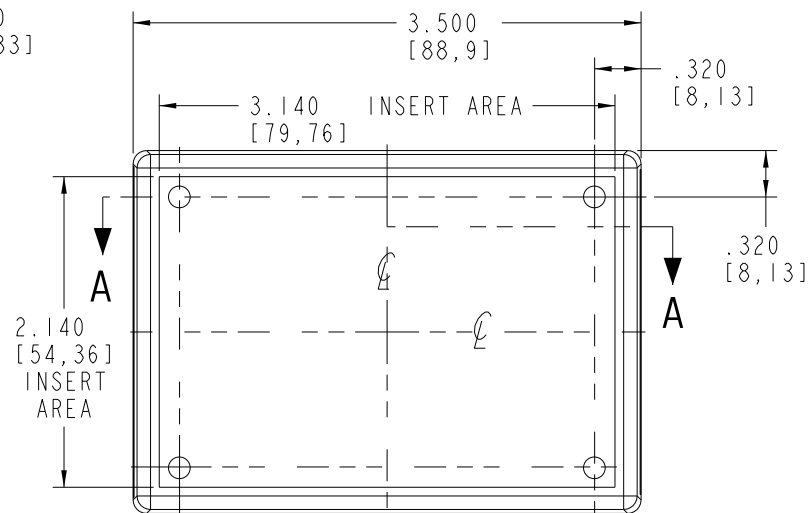
**SECTION D-D**



**RECOMMENDED PCB LAYOUT**



**SECTION A-A**



**FLX2535**

**FLXT2535**

**PACINTEC®**  
1-610-361-4200

**FLX2535**  
**FLXT2535**

VISIT **PACINTEC®**  
ON THE WORLD WIDE WEB  
[www.pactecenclosures.com](http://www.pactecenclosures.com)

COLOR / MATERIAL CODE		
COLOR	94HB	94VO
BLACK	000	028
P.C.BONE	039	060

DRAWING NUMBER	REV
74978	B

### 2.3. Caractéristiques et fonctionnement du PK4PWM

#### Alimentation :

Le PK4PWM s'alimente indifféremment de **10 à 35V DC** par les bornes à vis **DCIN** (borne +) et **GND** (borne -) (Cf. 2.1 Equipement face bottom)

Deux régulateurs (5V et 3.3V) alimentent le processeur et les composants internes.

Enfin, une diode de protection évitera les inversions de polarité.

#### Sorties PWM :

Le PK4PWM utilise les 4 sorties PWM du RCM3110 (Cf. 1.1) : à savoir PF4, PF5, PF6 et PF7. Celles-ci sont amplifiées grâce à des composants de puissance (2 ULN2803A). Comme chaque ULN2803A peut fournir 500mA, **chaque sortie pourra ainsi fournir  $500/2 = 250$  mA.**

On retrouve physiquement ces sorties sur les bornes à vis **PWM0 à PWM3** (Cf. 2.1 Equipement face bottom).

1/ Chaque sortie PWM est complètement indépendante.

Doté d'un registre unique de commande PWM, le TIMER A9 (TAT9R du Timer A) du Rabbit3000 permet de fractionner le "ratio PWM" en 255 parties égales. Cependant, l'utilisateur pourra utiliser une plage variant de 1 à 1024 mais le timer se calera sur la valeur de rationnement la plus proche pour effectuer sa modulation. (Cf. 2.4.1. Fonctions du Dynamic C Premier pour réaliser du PWM)

2/ Si on utilise le RCM3110 à **plein régime**, sa "plage de rationnement" varie de **50 Hz à ( $50*255$  =) 12,75 kHz avec un pas de 50 Hz.**

3/ Il est possible de diminuer ce pas en modifiant la variable **CLOCK\_DOUBLED** du *Rabbitbios.c* en la fixant à "0". Dans ce cas précis, le Rabbit3000 voit sa fréquence divisée par deux et la plage de rationnement PWM varie de **28 Hz à ( $28*255$  =) 7.1 kHz avec un pas de 25 à 28 Hz.**

4/ De manière plus software, il est possible d'abaisser le rationnement à une précision de quelques Hz (**environ 8Hz**).

#### Liaison série

Le PK4PWM est prévu pour dialoguer directement par liaison RS232. Pour cela les pins **PC0, PC1, PC2 et PC3** (correspondant au port D et au port C) du RCM3110 ont été utilisés et ainsi configurés en niveau TTL (via un MAX232N).

2 liaisons série 3 fils sont ainsi disponibles sur les bornes à vis du PK4PWM : **RXC, TXC, RXD, TXD et une masse commune GND.** (Cf. 2.1 Equipement face bottom)

#### Autres caractéristiques hardware

Le PK4PWM possède un cœur RCM3110 doté d'un microprocesseur Rabbit3000 à 29.4MHz, de 256k de Flash mémoire et 128K de SRAM. Ce module est également équipé d'une RTC et d'un watchdog.

Ces caractéristiques sont largement suffisantes pour une application utilisant port série et générant 4 sorties PWM.

## Environnement

Le PK4PWM peut être livré avec ou sans boîtier. Dans les deux cas, il ne bénéficie d'aucune norme IP. Il est prévu pour être renfermé dans un milieu "relativement" sain (ni trop humide, ni trop poussiéreux).

Hors boîtier, ses caractéristiques sont celles du RCM3110 (-40° à +85°C), 5 à 95 % d'humidité saine. Ne pas oublier d'exclure tout appareil pouvant créer des interférences à moins de 2 cm au dessus du RCM3110. (dû aux radiations du Rabbit3000)

## 2.4. Programmation du PK4PWM

### 2.4.1. Prise en main du PK4PWM

Pour programmer votre PK4PWM, ouvrez votre boîtier en retirant les 4 vis au dessus du boîtier.

Dans le cas ou votre module n'a pas de boîtier, connectez directement votre cordon de programmation (petit connecteur PROG) sur le module interne RCM3110 à l'endroit prévu (*attention de respecter le sens de la nappe en faisant correspondre la bande rouge sur la pin 1 du connecteur*). Connecter l'autre extrémité de votre cordon de programmation sur le port série de votre PC.

N'oubliez pas d'alimenter votre PK4PWM sur les bornes DCIN et GND.(10 à 35 VDC)

### 2.4.2. Fonctions utiles du Dynamic C Premier

#### POUR LES PWM

**pwm\_init** in <R3000.LIB>

SYNTAX: unsigned long pwm\_init(unsigned long frequency);

DESCRIPTION: Sets the base frequency for the PWM pulses and enables the PWM driver on all four channels. The base frequency is the frequency without pulse spreading. Pulse spreading (see pwm\_set) will increase the frequency by a factor of 4.

PARAMETER1: frequency(in Hz)

RETURN VALUE: actual frequency set. This will be the closest possible match to the requested frequency.

**pwm\_set** in <R3000.LIB>

SYNTAX: int pwm\_set(int channel, int duty\_cycle, int options);

DESCRIPTION: Sets a duty cycle for one of the PWM channels.

The duty cycle can be a value from 0 to 1024, where 0 is logic low the whole time, and 1024 is logic high the whole time. Option flags are used to enable features on an individual PWM channel.

Bit masks for these are:

PWM\_SPREAD - sets pulse spreading. The duty cycle is spread over four separate pulses to increase the pulse frequency.

PWM\_OPENDRAIN - sets the PWM output pin to be open-drain instead of a normal push-pull logic output.

PARAMETER1: channel - PWM channel(0 to 3)

PARAMETER2: duty\_cycle - value from 0 to 1024

PARAMETER3: options - combination of optional flags(see above)

RETURN VALUE: 0 if ok  
-1 if an invalid channel number is used  
-2 an invalid duty\_cycle was requested

## **POUR LA RS232**

**serXopen** in <RS232.LIB> (**X = port C or D**)  
SYNTAX: int serDopen(long baud);  
DESCRIPTION: Opens the D serial port. This function is non-reentrant.  
PARAMETER1: baud : bits per second of data transfer  
RETURN VALUE: 1: The baud set on the rabbit is the same as the input baud  
0: The baud set on the rabbit does not match the input baud

**serXwrFlush** in <RS232.LIB> (**X = port C or D**)  
SYNTAX: void serXwrFlush();  
DESCRIPTION: Flushes the serial port X transmit buffer. This function is non-reentrant.  
PARAMETER1: None  
RETURN VALUE: None

**serDwrFlush** in <RS232.LIB> (**X = port C or D**)  
SYNTAX: void serXwrFlush();  
DESCRIPTION: Flushes the serial port X input buffer. This function is non-reentrant.  
PARAMETER1: None  
RETURN VALUE: None

**serCread** in <RS232.LIB> (**X = port C or D**)  
SYNTAX: int serCread(void \*data, int length, unsigned long tmout);  
DESCRIPTION: Reads length bytes from serial port C or until tmout milliseconds transpires between bytes. If there is no data available when the function is called it will return immediately. This function is non-reentrant.  
PARAMETER1: data : data structure to read from serial port C  
PARAMETER2: length: number of bytes to read  
PARAMETER3: tmout : milliseconds max wait for any byte from previous one.  
RETURN VALUE: The number of bytes read from serial port C

**serDwrite** in <RS232.LIB> (**X = port C or D**)  
SYNTAX: int serDwrite(void \*data, int length);  
DESCRIPTION: Transmits length bytes to serial port D. This function is non-reentrant.  
PARAMETER1: data : data structure to write to serial port D  
length: number of bytes to write  
RETURN VALUE: The number of bytes successfully written to serial port D.

### 2.4.3. Exemple typique de programme pour faire fonctionner le PK4PWM

```
/* PROGRAMME PROTOCOLAIRE DE COMMUNICATION
   PK4PWM / RCM3110 pour PWM - J.BOUVIER
   Programme typique pour fonctionnement du PK4PWM
*/

/***** DESCRIPTIF : SUR LE PORT D *****/
On attend continuellement la réception de 3 caractères provenant de la liaison D du
PK4PWM.
- Soit pour démarrer le pwm X : 3 octets : "0xb(numéro de pwm)" + "durée d'activation en
1/10e de sec" + "ration de fonctionnement en %"
- Soit pour communiquer entre la liaison C et D : 3 octets "0xbb" + "nbre de car à recevoir"
+ "1 octet fixé à 0x55"
Après avoir reçu les caractères reçus sur le port D , on les envoie sur la liaison série C.
*/

/***** DESCRIPTIF : SUR LE PORT C *****/
Après avoir reçu les caractères provenant de la liaison série D , on les envoie sur la liaison
série C. Puis on active la réception des caractères sur le port C (costate liaison_port_C).
Après avoir tout reçu venant de la liaison série C, on les ré-envoie sur la liaison D.
*/

#define DINBUFSIZE 63
#define DOUTBUFSIZE 63
#define CINBUFSIZE 63
#define COUTBUFSIZE 63

char recD[3]; // buffer de réception sur le port D pour commande
char dataD[300]; // buffer de réception sur le port D pour les données
char recC[1]; // buffer de réception sur le port C pour commande
char dataC[3000]; // buffer de réception sur le port C pour les données
char _dataC[1];

CoData pwmX;
CoData liaison_port_C;

void main()
{
    int pwm_options, done, n1, i, j, delay;
    int rpwm0, rpwm1, rpwm2, rpwm3;
    long tpwm0, tpwm1, tpwm2, tpwm3;
    rpwm0=rpwm1=rpwm2=rpwm3=0;
    tpwm0=tpwm1=tpwm2=tpwm3=0;

    serDopen(57600);
    serCopen(57600);
    pwm_init(1000);
    pwm_options=0;
    CoReset(&pwmX);
    CoReset(&liaison_port_C);
}
```

```

while (1)
{
    costate
    {
        serDrdFlush();
        waitfor( serDread(recD,3,100) );
        switch (recD[0])
        {
            case 0x30 : tpwmX=(long)(recD[1]); rpwmX=(int)(recD[2]); CoBegin(&pwmX);
                        break;          // Démarrage du pwmX
            case 0x39 : waitfor( serDread(dataD,recD[1],80)||DelayMs(500) );
                        // Réception des n1=recD[1] data
                        serCwrFlush(); serCwrite(dataD,recD[1]);
                        // Transmission de ces n1 data vers le port C
                        CoBegin(&liaison_port_C); break;
            default : break;
        }
    }
}

costate liaison_port_C
{
    serCrdFlush(); done=0;
    waitfor ( serCread(recC,1,40)|| DelaySec(3));
        // Attente réception d'un caractère ou 3sec d'écoulé
    dataC[0]=recC[0];

    for (j=0;j<(i-1);j++)    // =(i-1) boucles
    {
        _dataC[0]=dataC[j];
        waitfor (DelayMs(2));    // temps d'attente de réception (pas obligatoire)
        serDwrFlush(); serDwrite(_dataC,1);
    }
}

costate pwmX
{
    pwm_set(0, (int)(0.01 * 1024 * rpwmX), pwm_options);
        // Active la sortie PWM0 pendant la durée demandée
    waitfor (DelayMs(tpwm0*100));
    pwm_set(0, 0, pwm_options);
}
}
}

```

## 2.5. Schéma électronique du PK4PWM

