

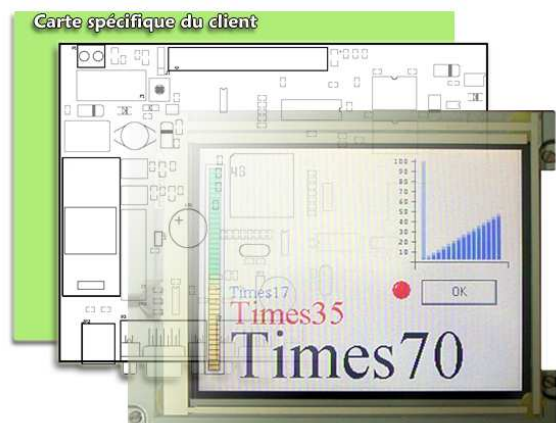
## XB10 – Manuel utilisateur

# Pragmatec

*Produits et services dédiés aux systèmes embarqués*

---

# XB10 – Manuel utilisateur





Bâtiment EARHART  
ZAC Grenoble Air Parc  
38590 St Etienne de St Geoirs - France  
[www.pragmatec.net](http://www.pragmatec.net)

PRAGMATEC

# XB10 – Manuel utilisateur

## TABLE DES MATIERES

<b>1</b>	<b><i>Présentation de l'équipement</i></b> .....	<b>4</b>
	<b>Equipement électronique</b> .....	<b>4</b>
	<b>Mise en œuvre du XB10</b> .....	<b>5</b>
	<b>Connexion au workbench STRATON</b> .....	<b>6</b>
<b>2</b>	<b><i>Tutorial : ajout d'une jauge de température</i></b> .....	<b>9</b>
	<b>Présentation de l'application</b> .....	<b>9</b>
	<b>Ajout de la jauge de température</b> .....	<b>9</b>
	<b>Utilisation d'une image de fond</b> .....	<b>11</b>
	<b>Ajout des boutons « + » et « - »</b> .....	<b>12</b>
	<b>Association d'une LED</b> .....	<b>13</b>
	<b>Manipulation des chaînes de texte</b> .....	<b>14</b>
<b>3</b>	<b><i>Configuration matérielle et logicielle</i></b> .....	<b>16</b>
	<b>Configuration hardware du XB10</b> .....	<b>16</b>
	<b>Configuration logicielle du XB10</b> .....	<b>21</b>
<b>4</b>	<b><i>Utilisation de cartes d'extension</i></b> .....	<b>22</b>
	<b>Carte d'extension sur bus I2C</b> .....	<b>22</b>
	<b>Cartes d'extension sur bus CAN</b> .....	<b>25</b>
	<b>Conversion analogique / numérique</b> .....	<b>28</b>
	<b>Gestionnaire d'entrées</b> .....	<b>29</b>
	<b>Gestionnaire générique d'IO (GPIO)</b> .....	<b>30</b>
	<b>Gestionnaire RS232 / RS485</b> .....	<b>31</b>
	<b>Cartes d'extension sur Ethernet</b> .....	<b>32</b>
<b>5</b>	<b><i>Remarques</i></b> .....	<b>33</b>
	<b>Temps de cycle</b> .....	<b>33</b>
	<b>Cas de dépassement de temps de cycle</b> .....	<b>33</b>
	<b>Mis à jour de l'application</b> .....	<b>34</b>



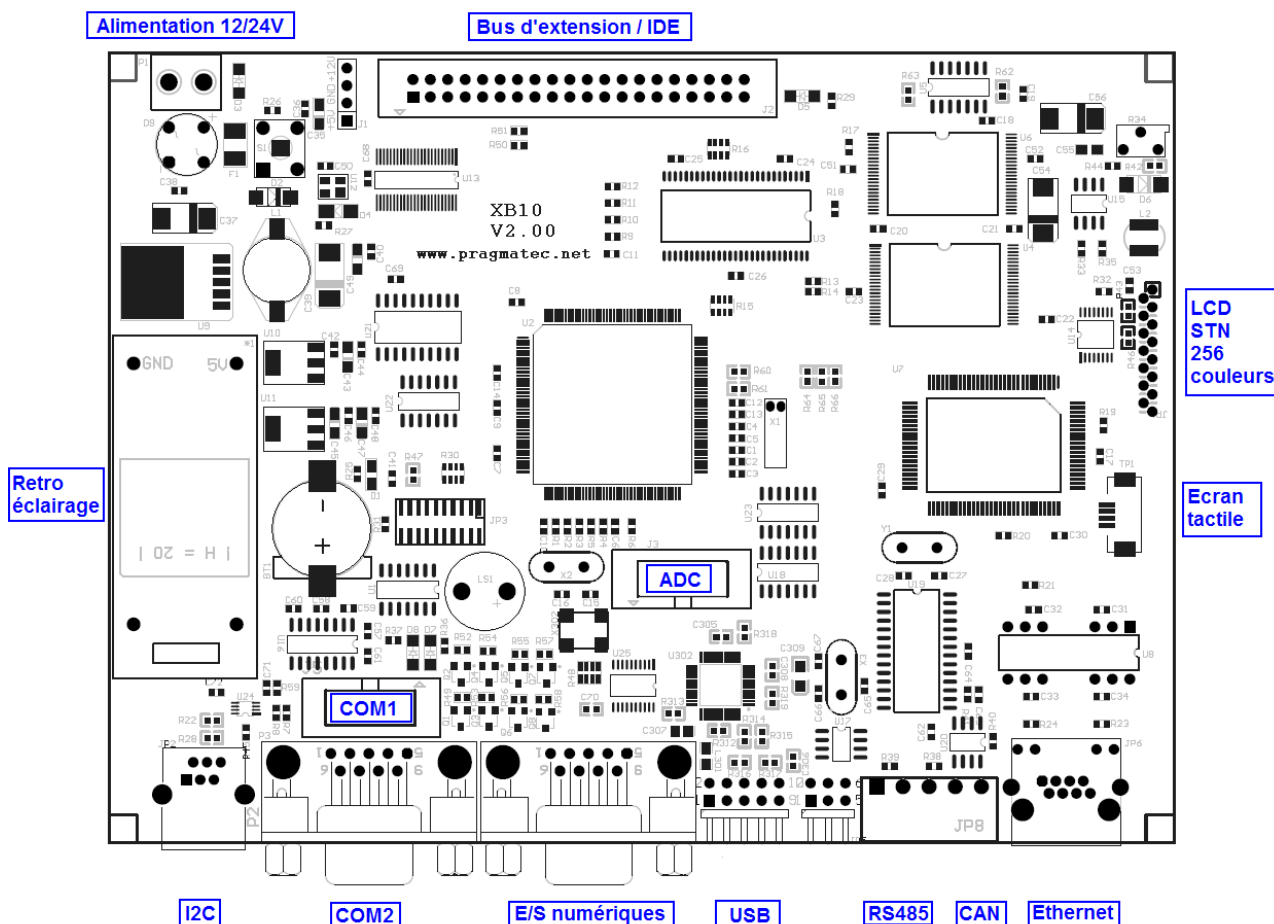
## 1 Présentation de l'équipement

Le but de ce document est de permettre à un utilisateur de mettre en œuvre et d'exploiter l'automate XB10 au travers du runtime STRATON développé par la société COPALP.

Nous commencerons donc par un tutorial destiné à présenter les fonctionnalités de la carte et sa programmation depuis le workbench STRATON. Enfin nous détaillerons la liste des interfaces du XB10 pilotables depuis STRATON ainsi que leur syntaxe.

### Equipement électronique

Le XB10 est basé sur le concept suivant : une carte électronique principale disposant des capacités proche de celles d'un petit PocketPC permet de gérer le programme d'un automate à l'aide du cœur d'automate STRATON. Afin d'élargir les capacités du système, le XB10 peut accueillir une carte spécifique du client avec par exemple des entrées /sorties numériques opto-isolées, des relais, un asservissement moteur par microcontrôleur, ...



## **XB10 – Manuel utilisateur**

Le XB10 est constitué de base des périphériques suivants :

- CPU 60MHz
- 16 Mo de mémoire RAM
- 16 Mo de mémoire flash NAND (stockage des logs, des images, ...)
- une connexion Ethernet 10Mb/sec
- une connexion RS232
- une connexion RS232 ou RS485
- une connexion USB host v1.1
- une connexion DB9 pour 4 entrées numériques et 4 sorties TOR
- une connexion RJ11 pour un bus I2C rapide (400KHz)
- une connexion bus CAN 2.0B jusqu'à 1Mb/s
- une connexion disque dur / CDROM / bus d'extension
- une connexion à un LCD STN 256 couleurs
- un lecteur de SD card
- une alimentation 12V / 24 V DC
- une horloge calendrier sauvegardée par pile

Ces périphériques sont directement gérés par la plate-forme XB10/STRATON pour vous permettre de réaliser votre application automate sans vous soucier aux contraintes liées aux accès de bas niveau.

### **Mise en œuvre du XB10**

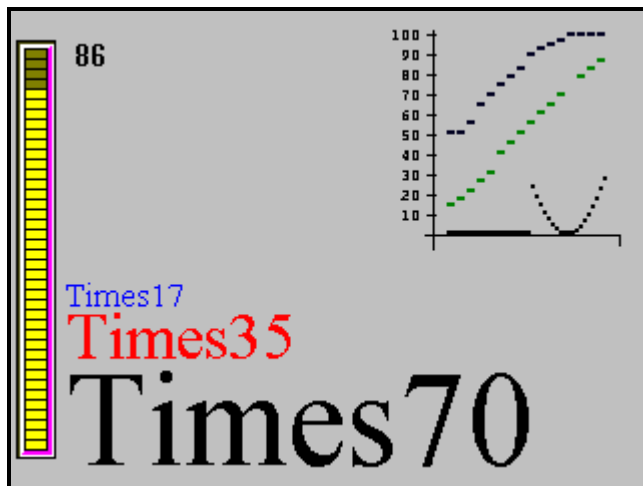
Pour mettre en œuvre le XB10, vous devez l'alimenter par une tension continue de **+12V DC** au minimum et **+30V DC** au maximum. Le XB10 est prévu typiquement pour une tension d'entrée de +24V DC, communément utilisée dans le monde de l'automatisme.

Après l'avoir mis sous tension vous devriez voir apparaître le logo de la société Pragmatec :



Par la suite le XB10 teste la présence d'un fichier t5.cod dans son système de fichier. Ce fichier correspond à votre application interprétable par le runtime STRATON. Si le fichier est bien présent, il est immédiatement exécuté au démarrage du XB10, si bien que lors du premier démarrage, vous devriez voir apparaître l'application de démonstration :

## XB10 – Manuel utilisateur



Le principe de l'application de démonstration est le suivant :

- Une barre de progression verticale à gauche permet d'afficher la valeur d'un angle radian ramené sur une échelle de 0 à 100 (valeur affichée en haut de la barre de progression)
- Un graph situé sur la droite de l'écran présente 3 types de courbes (2 en mode HISTO et 1 en mode GRAPHDOT)
- Affichage de 3 types de texte de couleurs et de taille différente en bas de l'écran
- Un objet jauge de température qui fixe un seuil de contrôle. Ce seuil peut être modifié à l'aide des touches « + » et « - » (utilisation de l'écrantactile).

Si vous venez de mettre sous tension votre XB10 pour la première fois, vous ne verrez pas apparaître la jauge de température ni même les boutons associés. Pour cela il vous faudra suivre le tutorial et modifier le programme de démonstration pour faire apparaître les éléments manquants.

### [Connexion au workbench STRATON](#)

Le XB10 possède une connexion réseau afin de pouvoir s'y connecter depuis Windows. Utilisez un **câble réseau croisé** comme celui fourni avec votre kit de développement.

L'adresse IP de votre système XB10 est par défaut : 192.168.0.30

Paramétrez donc votre PC sous Windows avec une adresse IP compatible avec le masque de sous-réseau du XB10 (255.255.255.0), comme l'adresse IP : 192.168.0.10.

Si vous ne pouvez le faire, il vous faudra modifier l'adresse IP du XB10 pour qu'il ait une adresse appartenant au même sous-réseau que votre PC. Pour cela, consulter le chapitre relatif aux modifications des paramètres du XB10 par TELNET ou FTP.

Ceci étant fait, connectez le XB10 au PC en insérant le cordon Ethernet dans le connecteur RJ45.

Lancez l'application « straton workbench » sur votre PC. Si vous ne disposez pas de version licenciée, vous pouvez utiliser la version de démonstration disponible auprès de la société COPALP. Dans ce cas votre version de démonstration sera limitée de par le nombre d'entrées / sorties du système :

## XB10 – Manuel utilisateur

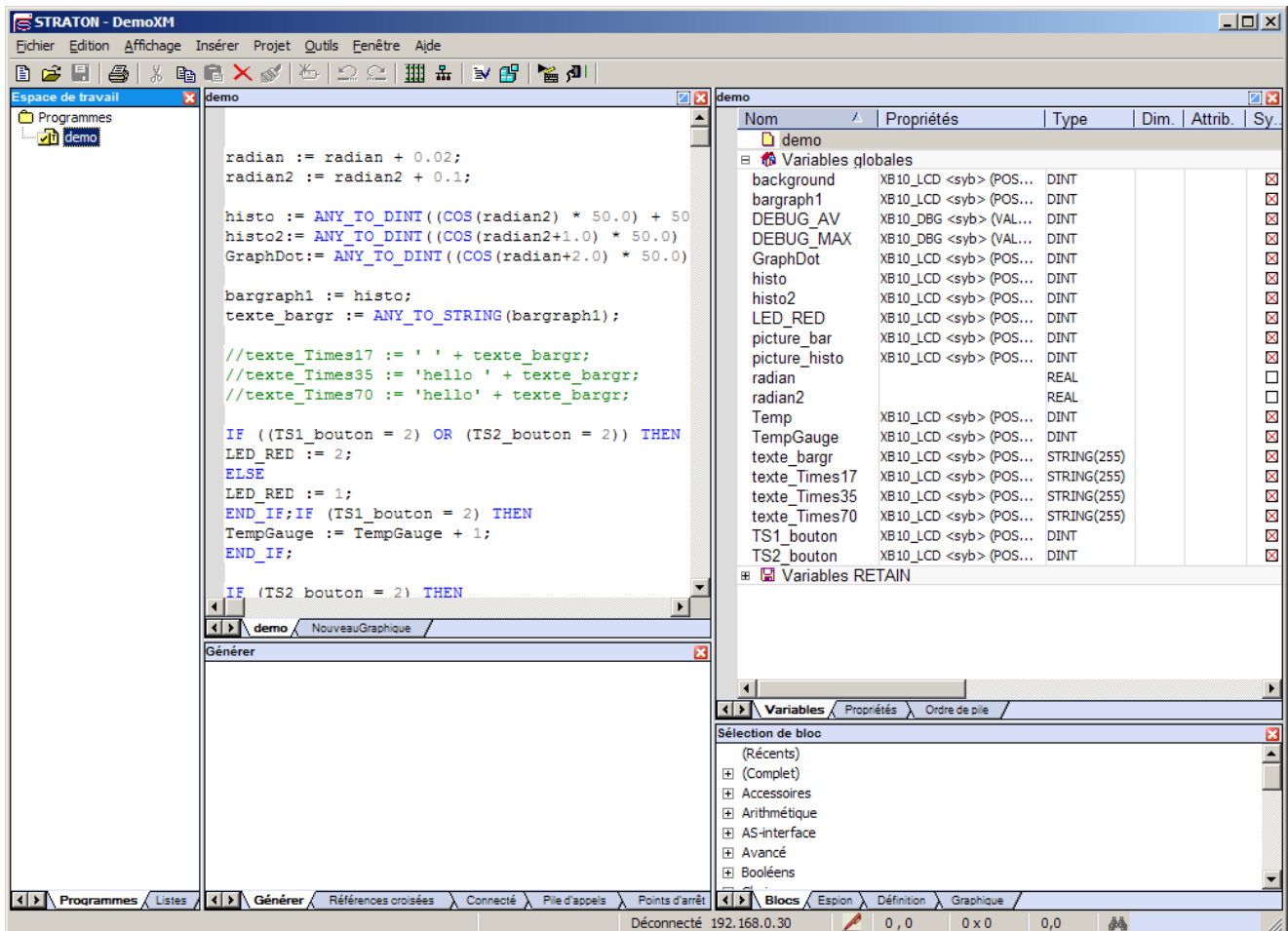


Ensuite il est nécessaire d'ouvrir un projet STRATON. Copiez préalablement le répertoire « DemoXM » du CDROM dans le répertoire de votre choix, par exemple « C:\workspace », puis sélectionnez comme suit l'ouverture d'un programme sous le workbench :



## XB10 – Manuel utilisateur

Une fois le projet sélectionné, le workbench s'ouvre sans tenter de se connecter à la cible. Vous verrez 2 fenêtres essentielles s'ouvrir : celle du centre correspond au code de l'automate, et celle de droite qui contient la liste des variables utilisées par le programme.



Cliquez alors sur le bouton le plus à droite (« Exécution ») afin de vous connecter au XB10. Si vous faites une modification dans le programme et que vous souhaitez recompiler le programme, le workbench vous proposera automatiquement de le recharger dans le XB10 et l'exécution du nouveau programme se fera automatiquement.

## XB10 – Manuel utilisateur

### 2 Tutorial : ajout d'une jauge de température

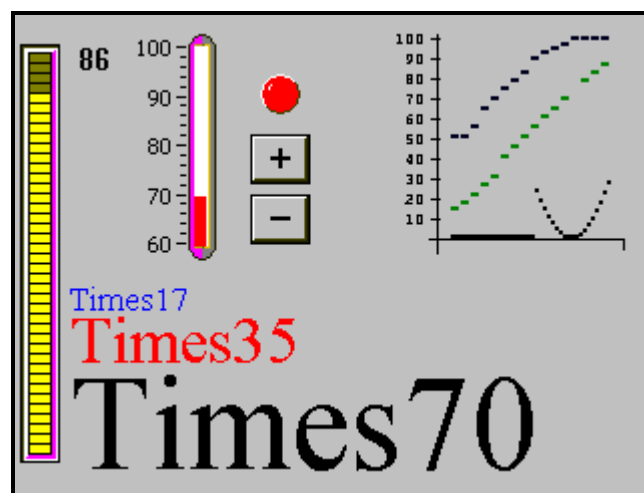


Le but de ce tutorial est de présenter l'interface spécifique du XB10 au travers de l'atelier STRATON WORKBENCH. Le programme de démonstration DemoXM s'affiche pour la première fois comme indiqué dans le précédent chapitre, c'est-à-dire sans la jauge de température. Nous allons ici effectuer quelques modifications afin de faire apparaître la jauge de température et de gérer son paramétrage depuis l'écran tactile.

#### Présentation de l'application

Le but a présent est donc de placer à l'écran une jauge de température. Le projet STRATON du workbench possède déjà les éléments nécessaires à la création de la jauge de température, aussi vous n'aurez qu'à valider certains éléments pour permettre aux objets d'apparaître à l'écran.

Voici à quoi devra ressembler l'écran graphique à la fin du chapitre :



De plus nous allons utiliser l'écran tactile pour paramétrer la valeur de la jauge de température. Celle-ci pourra donc servir de consigne de température.

#### Ajout de la jauge de température

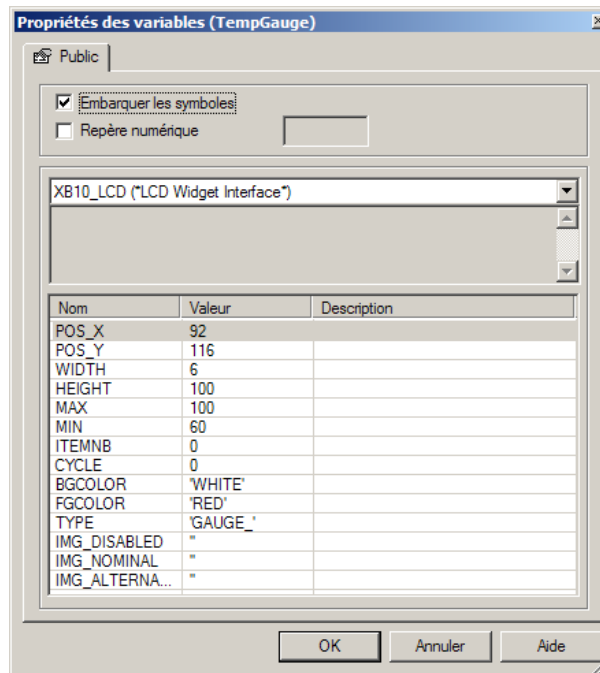
Nous allons donc commencer par ajouter une jauge de température. Pour cela nous allons créer un objet graphique « JAUGE\_V » qui est une jauge verticale. Une JAUGE est en fait un rectangle qui possède une couleur de fond et une couleur principale.

De plus, les JAUGES peuvent être à position verticale ou horizontale.

Ici nous utiliserons donc une JAUGE\_V, c'est-à-dire une jauge verticale, de couleur rouge avec une couleur de fond blanc.

## XB10 – Manuel utilisateur

Lancez le workbench STRATON et déconnectez-vous de la cible.  
Ensuite, double cliquez sur la variable « TempGauge », qui correspond à notre jauge de température. Il doit apparaître la fenêtre suivante :



Vous pouvez constater que la variable est de type XB10\_LCD (objet graphique du XB10), et qu'elle est du type 'GAUGE\_'. Ce type est incorrect car il est incomplet. Le nom complet est « GAUGE\_V » pour indiquer qu'il s'agit bien d'une jauge verticale.

Sa couleur de fond (BGCOLOR) est le blanc (WHITE) et sa couleur d'affichage est le rouge (RED). L'objet est positionné à l'écran aux coordonnées suivantes : POS\_X = 92 et POS\_Y = 116. Le coin supérieur droit du LCD correspond aux coordonnées (X=0, Y=0).

L'objet possède aussi une largeur de 6 pixels (WIDTH) et une hauteur de 100 pixels (HEIGHT).

Enfin il est possible de définir à l'objet des valeurs minimales et maximales : MAX = 100, MIN = 60. Ainsi la variable devra impérativement se situer entre ces 2 bornes, elle ne pourra jamais varier en dehors de la plage [MIN-MAX].

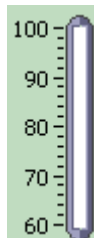
Double cliquez sur le texte '**GAUGE\_**' pour pouvoir le modifier et tapez '**GAUGE\_V**' puis ENTER sur votre clavier. Enfin fermez la fenêtre en cliquant sur OK, recompilez l'application et chargez là sur la cible.

Vous pouvez aussi tenter de vous connecter directement à la cible une fois la modification effectuée, le workbench recompilera automatiquement et vous proposera de charger la nouvelle application.

## XB10 – Manuel utilisateur

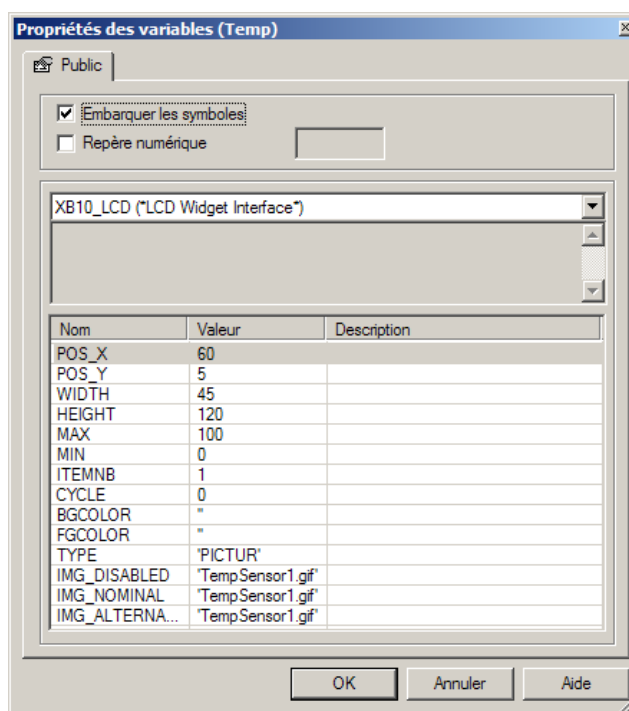
### Utilisation d'une image de fond

Notre jauge de température est un peu triste graphiquement et ne permet pas d'être bien identifiée à l'écran. Nous allons lui associer une image graphique de fond :



Notre image de fond affiche sur la gauche les dizaines de degrés de 60 à 100 °C.

Sous le workbench STRATON, double cliquez sur la variable « TEMP », une fois déconnecté de la cible. Vous verrez apparaître la fenêtre suivante :



Double cliquez sur le champ 'PICTUR' et remplacez la chaîne de texte par 'PICTURE'. Tout comme l'objet 'GAUGE\_V', l'objet 'PICTURE' possède des coordonnées [POS\_X-POS\_Y] ainsi qu'une taille [WIDTH-HEIGHT]. Attention de bien indiquer la taille réelle de l'image sans quoi le XB10 vous redimensionnera votre image à la taille que vous avez spécifié.

Vous remarquerez aussi que l'image possède 3 champs spécifiques :

- **IMG\_DISABLED** : correspondant à l'image qui est affichée lorsque la variable vaut 0
- **IMG\_NOMINAL** : correspond à l'image qui est affichée par défaut lorsque la variable vaut 1
- **IMG\_ALTERNATIVE** : correspond à l'image affichée lorsque la variable vaut 2

## XB10 – Manuel utilisateur

Dans notre cas, nous ne souhaitons pas modifier l'image affichée en fonction de valeur que prendrait la variable « TEMP », aussi nous affectons la même image aux 3 champs.

Vous pouvez préciser n'importe quel nom d'image au format BMP et GIF, en vérifiant toutefois que l'image en question se trouve bel et bien dans le répertoire /home de la cible (si ce n'est pas le cas, faites un transfert de l'image via un logiciel FTP tel que Mozilla).

Vous pouvez désormais recompilier et transférer l'application sur la cible. Une fois connecté, vous pourrez même modifier dynamiquement la valeur de la variable « TempGauge » en l'affectant à 80 par exemple :

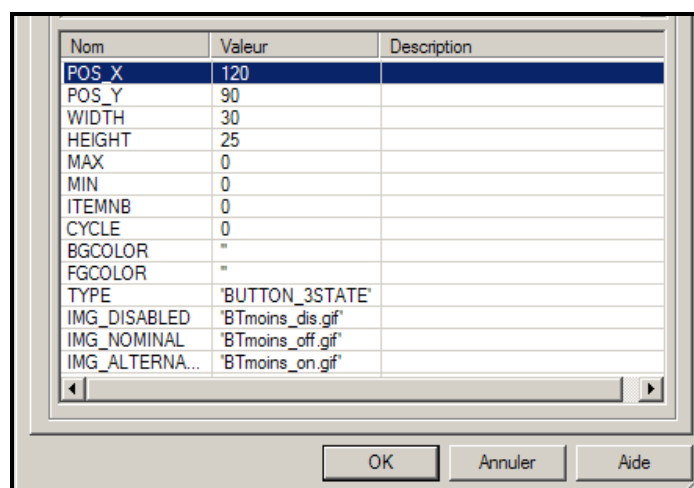
picture_histo	1	XB10_LCD <syb> (POS...	DINT
radian	1.44		REAL
radian2	7.2		REAL
Temp	1	XB10_LCD <syb> (POS...	DINT
<b>TempGauge</b>	<b>80</b>	<b>XB10_LCD &lt;syb&gt; (POS...</b>	<b>DINT</b>
texte_bargr	'84'	XB10_LCD <syb> (POS...	STRING(255)
texte_Times17	'Time[80]	XB10_LCD <syb> (POS...	STRING(255)
texte_Times35	'Times35'	XB10_LCD <syb> (POS...	STRING(255)
texte_Times70	'Times70'	XB10_LCD <syb> (POS...	STRING(255)
TS1_bouton	1	XB10_LCD <syb> (POS...	DINT
TS2_bouton	1	XB10_LCD <syb> (POS...	DINT
Variables RETAIN			

Si vous mettez une valeur en dehors des limites de l'objet TempGauge [60-100], la variable prendra bel et bien la valeur sous le workbench

### Ajout des boutons « + » et « - »

A présent que notre objet « TempGauge » est créé, nous désirons pouvoir modifier sa valeur depuis l'écran tactile. Pour cela nous allons utiliser 2 variables « TS1\_bouton » et « TS2\_bouton » qui seront 2 boutons tactiles permettant d'augmenter et de diminuer la valeur de la variable « TempGauge » :

Double cliquez sur les variables afin de modifier leur champ TYPE en « BUTTON\_TOGGLE » pour TS1\_bouton et « BUTTON\_3STATE » pour TS2\_bouton :



## XB10 – Manuel utilisateur

Le premier bouton TS1\_bouton fait apparaître un bouton « - », basé sur l'image 'Btmoins\_off.gif'. Tout comme pour les images, les objets BUTTON\_xxx permettent d'afficher jusqu'à trois types d'images selon la valeur de la variable.

En fait il s'agit réellement d'un objet 'PICTURE' auquel on a rajouté la capacité de modifier la valeur de la variable au travers de l'écran tactile :

- **l'écran tactile n'est pas touché** : l'image correspondant à 'IMG\_NOMINAL' est affichée
  - **l'écran est touché sur l'image** : l'image correspondant à 'IMG\_ALTERNATIVE' est affichée
  - **la variable vaut 0** : l'image correspondant à 'IMG\_DISABLED' est affichée
- Dans ce dernier cas, le bouton tactile est dé-validé.

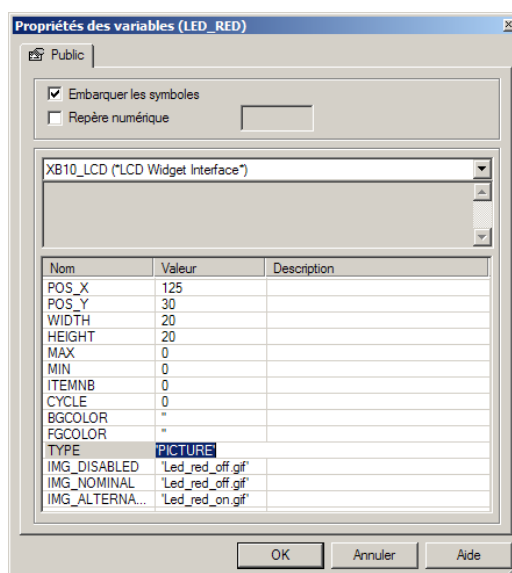
Recompilez et transférez la nouvelle application sur la cible. Vous constaterez alors que les 2 boutons ne réagissent pas de la même façon : le bouton « - » est activé tant qu'on reste appuyé sur le bouton alors que le bouton « + » conserve son action jusqu'à ce qu'on ré-appuie sur le bouton (fonction *TOGGLE*).

### Association d'une LED

L'ajout d'une LED peut s'avérer utile sur un écran LCD : elle permet d'indiquer une alarme ou bien qu'une action a bel et bien été prise en compte.

Nous allons ajouter une LED à notre projet qui s'allumera à chaque fois qu'une action sur les boutons aura lieu.

Double cliquez sur l'objet « LED\_RED » afin d'ouvrir sa fenêtre de propriété :



Modifiez le champ TYPE pour afficher la chaîne 'PICTURE'. En effet notre LED n'est ni plus ni moins qu'une image affichée à l'écran, qui possède 2 états : l'état allumé à l'aide de l'image « Led\_red\_on.gif » et l'état éteint grâce à l'image « Led\_red\_off.gif ».

## XB10 – Manuel utilisateur

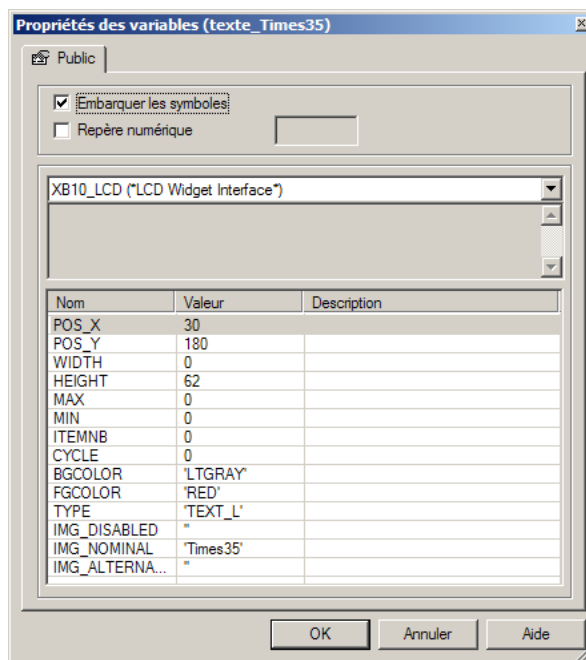
Ensuite il convient d'écrire le code de notre automate de sorte que la LED s'allume lorsqu'un des 2 boutons est appuyé. Ce code est déjà présent dans votre application :

```
IF ((TS1_bouton = 2) OR (TS2_bouton = 2)) THEN
LED_RED := 2;
ELSE
LED_RED := 1;
END_IF;
```

Ainsi lorsque l'une des 2 variables TR1\_bouton ou TS2\_bouton vaut 2 (valeur correspondant au champ IMG\_ALTERNATIVE, donc lorsque l'écran tactile est pressé), la variable LED\_RED prend la valeur 2 aussi (affichant ainsi l'image correspondant à IMG\_ALTERNATIVE soit 'Led\_red\_on.gif').

### Manipulation des chaînes de texte

Enfin nous terminerons ce chapitre en modifiant les chaînes de texte qui s'affichent sur votre écran. Double cliquez sur les propriétés de la variable « texte\_Times35 » pour voir apparaître la fenêtre suivante :



L'objet peut être de 2 types : 'TEXT\_L' pour un texte justifié à gauche ou 'TEXT\_R' pour un texte justifié à droite. Ensuite il convient de lui attribuer une police. Il peut s'agir d'une police de taille fixe (*Terminal, Times35, Times17, Times70, System*), ou bien d'une police TRUETYPE (fichier TTF) qui doit alors être transféré dans la cible via FTP sous le répertoire /home. Dans le cas d'une police TRUETYPE vous lui préciserez une taille de police dans le champ HEIGHT.

Enfin, l'objet possède une couleur d'affichage (RED ici) et une couleur de fond (LTGRAY ici). Il convient d'utiliser la même couleur de fond que l'image de fond afin de permettre l'effacement du texte lors d'une mise à jour.

## XB10 – Manuel utilisateur

Pour tester la modification de texte à l'affichage, dé-commentez une des lignes de code ci-dessous afin de modifier à chaque cycle la valeur de la variable `texte_Times35` :

```
//texte_Times17 := ' ' + texte_bargr;  
texte_Times35 := 'hello ' + texte_bargr;  
//texte_Times70 := 'hello' + texte_bargr;
```

Ainsi vous verrez le texte à l'écran être modifié de façon dynamique.



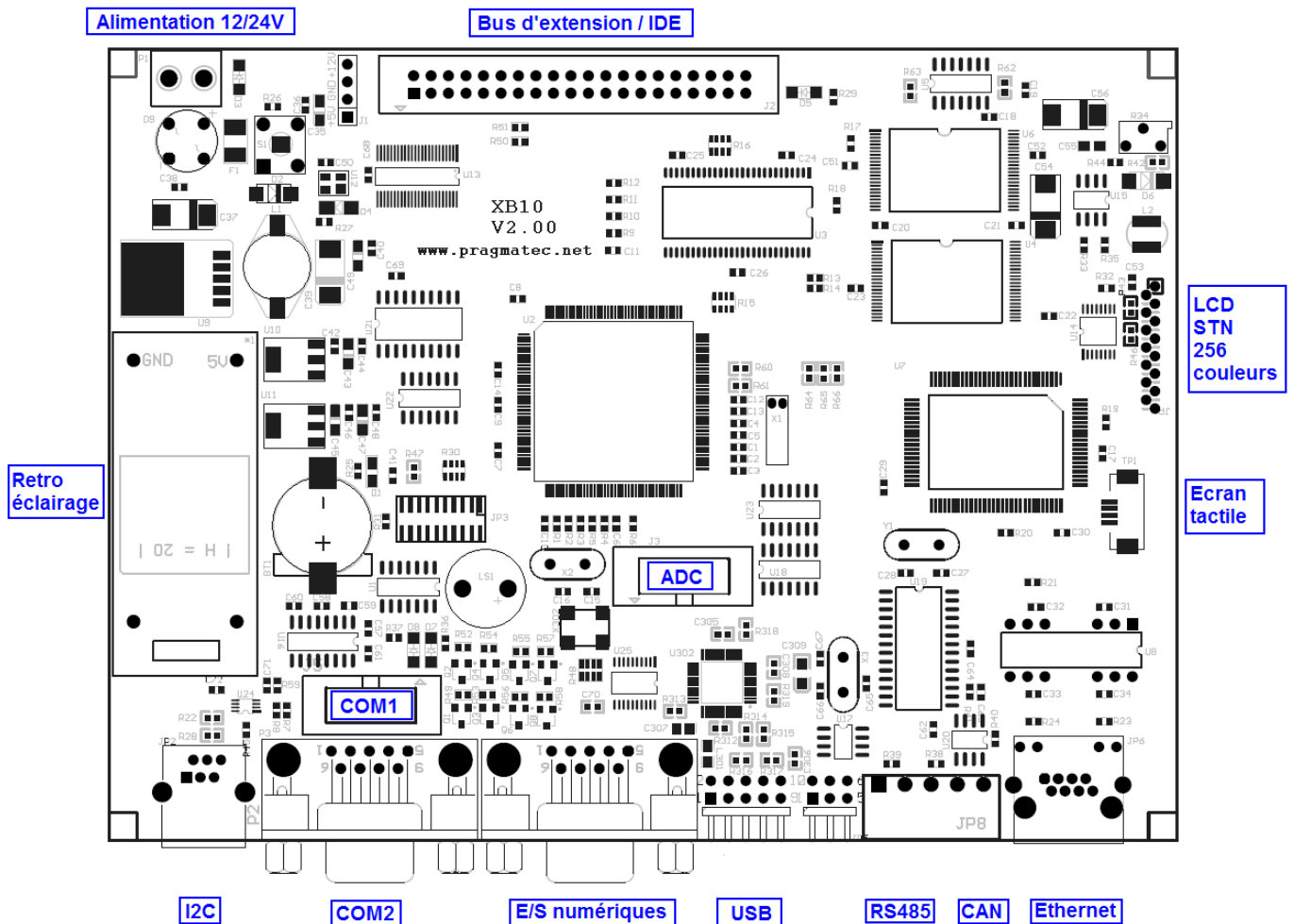
### 3 Configuration matérielle et logicielle

La carte XB10 est constituée d'un circuit imprimé de 115mm par 165mm, réalisé en 4 couches avec cerclage de reprise de masse tout autour de la carte (pour une plus grande immunité aux bruits). L'ensemble des circuits utilisés ainsi que la technique de fabrication sont conformes aux recommandations RoHS.

Le circuit reste toutefois sensible aux décharges électrostatiques comme tout élément constitué d'électronique, il convient donc de le manipuler avec précaution.

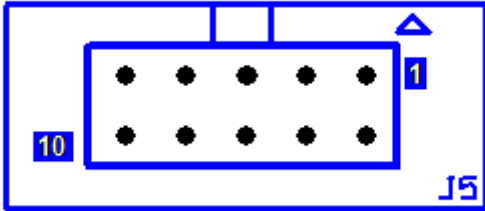
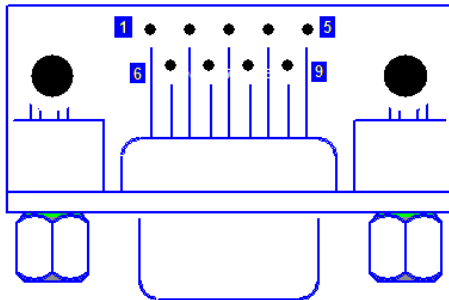
#### Configuration hardware du XB10

Le XB10 présente un certain nombre de connecteurs périphériques que l'on retrouve sur la carte comme sur sa périphérie :

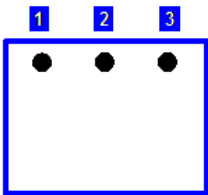

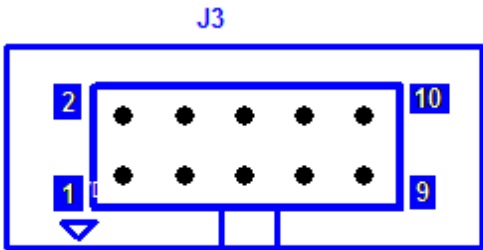


Vous trouverez ci-dessous un tableau vous indiquant les points de connexion des périphériques du XB10.

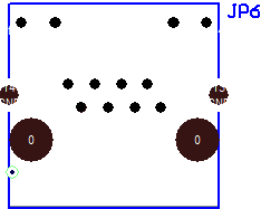
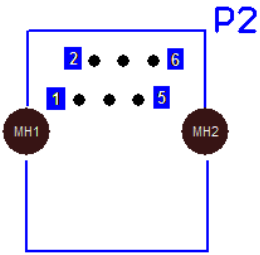
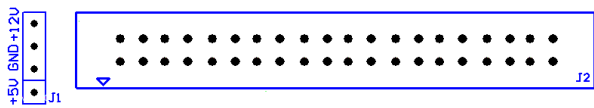
# XB10 – Manuel utilisateur

SCHEMATIQUE	DESCRIPTION
	<p><b>Port COM0</b> Destinée au debug essentiellement, elle est en fait libérée au lancement de linux, et donc libre d'utilisation pour une application quelconque. Attention le niveau électrique des signaux est TTL à 3.3V (5V tolérant). Ne pas y connecter un cordon RS232. Les broches 1 à 10 sont numérotées en quinconce :</p> <ul style="list-style-type: none"> <li>1 : Interruption externe N°7</li> <li>2 : I2C SDA</li> <li>3 : I2C SDL</li> <li>4 : COM0 RX</li> <li>5 : COM0 TX</li> <li>6 : COM0 RTS</li> <li>7 : COM0 CTS</li> <li>8 : 5V</li> <li>9 : GND</li> <li>10 : 3.3V</li> </ul> <p>Au démarrage du XB10, le port COM0 est utilisé pour imprimé des informations d'initialisation de la carte (BIOS) à 115200 bauds, 8 bits, 1 bit de stop et sans parité.</p>
 <p style="text-align: center;">Connecteur mâle</p>	<p><b>Port COM1</b> Elle est libre d'être utilisée par votre application. Les broches 1 à 9 sont numérotées en ligne :</p> <ul style="list-style-type: none"> <li>1 : NC</li> <li>2 : COM1 RX</li> <li>3 : COM1 TX</li> <li>4 : NC</li> <li>5 : GND</li> <li>6 : NC</li> <li>7 : COM1 RTS</li> <li>8 : COM1 CTS</li> <li>9 : GND</li> </ul> <p>Les signaux sont compatibles RS232 (connexion à un PC possible à l'aide d'un câble croisé). Le numéro des broches est indiqué à l'avant du connecteur, sur la partie plastique (convention sur connecteur DB9).</p>

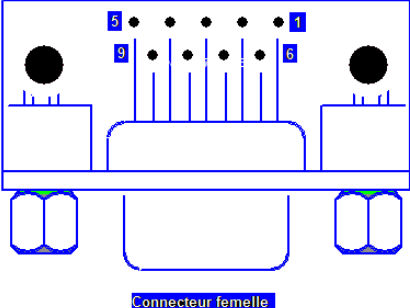
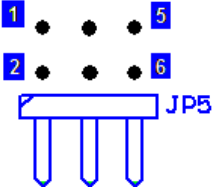
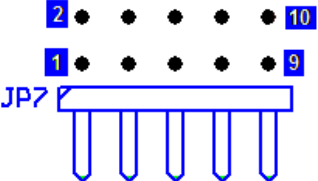
## XB10 – Manuel utilisateur

	<p><b>Bus CAN v2.0B</b> Connecteur pour bus CAN, les broches sont numérotées de gauche à droite (face composant) : 1 : GND 2 : CANL 3 : CANH Une résistance de terminaison peut être mise à l'aide d'un strap sur le connecteur « Jumper ». La carte possède un transceiver CAN MCP2551 de Microchip en interface avec le bus.</p>
	<p><b>Port RS485</b> Ce port est libre d'utilisation lorsque le strap adéquat a été placé sur le connecteur « Jumper ». Ce port est en fait multiplexé avec le port COM0 vu plus haut. Le port COM0 peut donc être du type</p> <ul style="list-style-type: none"> <li>• TTL 3.3V sur connecteur HE10 J5</li> <li>• RS485 sur connecteur débrochable</li> </ul> <p>Les points de connexion sont les suivants : 1 : RS485B 2 : RS485A</p>
	<p><b>Port Analogique</b> Port de conversion analogique/numérique de 8 voies 10 bits. Attention, le port n'est pas protégé en entrée, ceci doit être réalisé sur la carte d'extension. Le niveau de tension maximum est de 2.5V, le minimum étant de 0V. Les points de connexion sont les suivants : 1 : 2.5V 2 : AIN0 3 : AIN1 4 : AIN2 5 : AIN3 6 : AIN4 7 : AIN5 8 : AIN6 9 : AIN7 10 : GND</p>

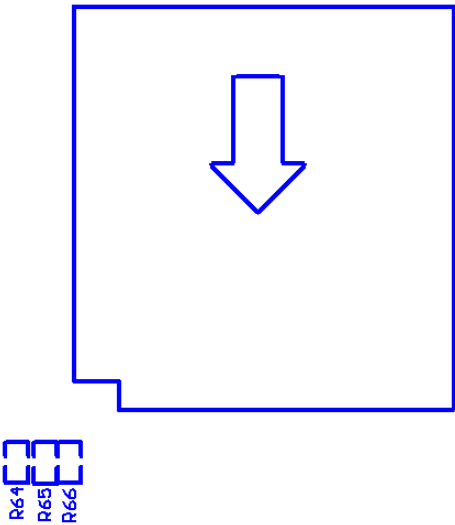
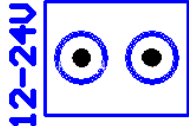
## XB10 – Manuel utilisateur

 <p>The diagram shows a square connector labeled JP6. It has two circular terminals at the bottom, each marked with a '0'. There are several pins arranged in a grid within the square frame.</p>	<p><b>Port Ethernet</b> Port Ethernet 10Mb/s qui nécessite un câble croisé. Vous pouvez un câble droit si le système connecté détecte automatiquement l'équipement XB10 et croise en interne ses signaux.</p> <p>Le XB10 bénéficie de serveurs TELNET et FTP intégré qui peuvent être accédés depuis un PC Windows ou Linux :</p> <ul style="list-style-type: none"> <li>• Login : root</li> <li>• Password : root</li> </ul>
 <p>The diagram shows a rectangular connector labeled P2. It has two circular terminals at the bottom, labeled MH1 and MH2. There are six pins in the top row, numbered 1 through 6. Pins 1 and 2 are connected to MH1, and pins 5 and 6 are connected to MH2.</p>	<p><b>Port I2C</b> Extension destinée à des distances inter-cartes courtes. Le XB10 possède les résistances de pull-up de 470 ohms montées en sorties d'un transceiver PCA9306 (valeurs recommandées pour ce transceiver).</p> <p>Les points de connexion sont les suivants :</p> <ol style="list-style-type: none"> <li>1 : VIN (tension d'alimentation)</li> <li>2 : VIN (tension d'alimentation)</li> <li>3 : SCL</li> <li>4 : SDA</li> <li>5 : GND</li> <li>6 : GND</li> </ol> <p>Les résistances de pull-up sont reliées à une tension de 5V. Pensez à connecter une électronique d'interface compatible 5V.</p>
 <p>The diagram shows two connectors. On the left is a 4-pin connector labeled J1 with pins labeled +5V, GND, +12V, and GND. On the right is a 40-pin connector labeled J2.</p>	<p><b>Port IDE</b> Extension IDE permettant la connexion d'un DiskOnModule PQI par exemple (testé jusqu'à 256Mo).</p> <p>L'interface est compatible TTL 5V. Un connecteur d'alimentation J1 situé à côté du connecteur 40 points permet l'alimentation du module :</p> <ol style="list-style-type: none"> <li>1 : 5V</li> <li>2 : GND</li> <li>3 : GND</li> <li>4 : VIN (alimentation d'entrée du XB10)</li> </ol>

## XB10 – Manuel utilisateur

 <p style="text-align: center;">Connecteur femelle</p>	<p><b>Port d'entrées/sorties (IO)</b> Gère à la fois 4 entrées et 4 sorties numériques. Les broches 1 à 9 sont numérotées en ligne :</p> <ul style="list-style-type: none"> <li>1 : GND</li> <li>2 : OUTPUT4</li> <li>3 : OUTPUT1</li> <li>4 : INPUT3</li> <li>5 : INPUT1</li> <li>6 : OUTPUT3</li> <li>7 : OUTPUT2</li> <li>8 : INPUT4</li> <li>9 : INPUT2</li> </ul> <p>Les entrées sont inactives au niveau haut (pull-up de 4.7Kohm au 3.3V). Elles sont actives à l'état bas. Utilisez un contact entre l'une des entrées INPUT et la masse.</p> <p>Les sorties ne sont pas opérationnelles (futur product)</p>
 <p style="text-align: center;">JP5</p>	<p><b>Connecteur « Jumper »</b> Permet de configurer l'emploi de la COM0 ainsi que la présence de la résistance de terminaison du bus CAN :</p> <ul style="list-style-type: none"> <li>1-2 ON : COM0 en RS232</li> <li>3-4 ON : COM0 en RS485</li> <li>5-6 ON : impédance de terminaison CAN</li> </ul> <p>L'impédance de terminaison du bus CAN est de 120 Ohms.</p>
 <p style="text-align: center;">JP7</p>	<p><b>Port USB</b> Le connecteur JP7 est un connecteur HE10 de 10 points symétrique, c'est-à-dire qu'il permet de connecter un adaptateur quel que soit son sens. Seuls 4 signaux sont nécessaires à la gestion d'un périphériques USB, mais on les retrouve de part et d'autre du connecteur :</p> <ul style="list-style-type: none"> <li>1 : 5V (Vusb)</li> <li>2 : GND (Shield USB)</li> <li>3 : USB D-</li> <li>4 : GND</li> <li>5 : USB D+</li> <li>6 : USB D+</li> <li>7 : GND</li> <li>8 : USB D-</li> <li>9 : GND (Shield USB)</li> <li>10 : 5V (Vusb)</li> </ul> <p>Vous pouvez y connecter une clef USB, un adaptateur USB/RS, ou encore un récepteur GPS.</p>

## XB10 – Manuel utilisateur

	<p><b>Connecteur SDCard</b></p> <p>Le XB10 permet de lire le contenu d'une Sdcard formatée VFAT (format Windows). Le connecteur est équipé d'un système d'extraction assisté : appuyer sur la SDcard pour l'éjecter.</p> <p>L'utilisation de la SDcard et du Touchscreen n'est pas compatible avec les drivers fournis avec le XB10 :</p> <ul style="list-style-type: none"> <li>• Pour utiliser un écran tactile, soudez la résistance R65 et retirez la R64.</li> <li>• Dans le cas d'une utilisation de la SDcard, soudez la résistance R64 et retirez la R65.</li> </ul>
	<p><b>Connecteur d'alimentation</b></p> <p>Le XB10 peut être alimenté par une tension comprise entre 12V et 30V. Son alimentation nominale est de 24V.</p> <p>Il peut être alimenté par une tension négative (inverseur de polarité intégré).</p> <p>L'alimentation est équipée d'un fusible Polyswitch réarmable automatiquement.</p>

### Configuration logicielle du XB10

Le XB10 est équipé d'une version allégée de Linux, adaptée au processeur sans mémoire virtuelle (NO\_MMU) ni unité de calcul à virgule flottante câblée (NO\_FPU). Le noyau utilisé est ainsi un noyau uClinux 2.4.24, adapté par Pragmatec afin d'accroître les performances du XB10 tout en garantissant sa stabilité et sa fiabilité.

De plus il est fourni avec un ensemble de logiciels :

- 2 serveurs embarqués TELNET et FTP (login : root / password : root)
- 2 partitions embarquées de 8Mo chacune (/home et /usr)
- Le serveur graphique nano-X
- Le runtime STRATON et les ressources du programme de démonstrations
- Tous les drivers nécessaires pour piloter les périphériques (USB, Ethernet, RS, CAN, ...)
- Un lancement automatique de l'application à l'aide du script modifiable /home/rc.sh

Quant aux développements logiciels de votre programme, ils sont de 2 types :

- En C à l'aide de l'interface de développement Eclipse (codage et debug)
- En langage normalisé d'automatisme à l'aide des outils Straton de la société Copalp

Référez-vous aux documentations relatives aux outils de développement choisis.



## 4 Utilisation de cartes d'extension

Le système XB10 de base comprend un ensemble de contrôleur d'interfaces tels l'Ethernet, la RS232 ou l'I2C par exemple. Il convient alors de connecter les périphériques adéquats afin de permettre leur gestion depuis le Workbench straton.

Les paragraphes qui suivent permettent d'exploiter les différents périphériques de la carte.

### Carte d'extension sur bus I2C

Le bus I2C est un bus série synchrone, c'est-à-dire composé de très peu de fils (4) dont un est l'horloge qui permet de cadencer la vitesse du bus. Le processeur du XB10 est le maître sur le bus, il interroge donc les différents périphériques I2C qui sont connectés au bus en tant qu'esclave.

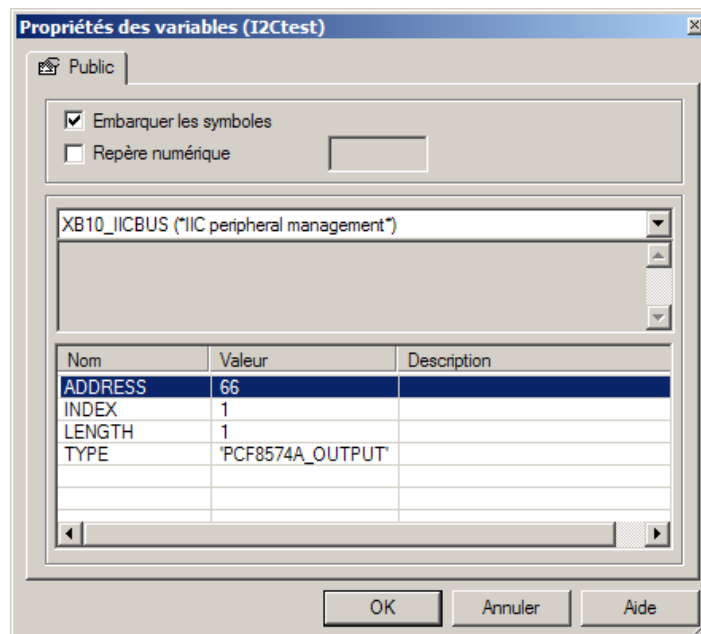
L'I2C est en général utilisé pour piloter des entrées / sorties distantes de quelques dizaines de centimètres et pour des événements relativement lent et peu fréquent : pilotage d'électrovannes, lecture d'alarmes, pilotage de LEDs d'état de fonctionnement, lecteur de badges, ...).

Le XB10 permet, dans sa version actuelle, de gérer un seul périphérique : le PCF8574A. Ce composant créé par Philips (concepteur du bus I2C) permet de gérer 8 entrées / sorties numériques au travers du bus I2C. De nombreux autres périphériques seront gérés par le XB10 prochainement tels les MCP23016 de Microchip.

Pour le bon fonctionnement de votre système nous vous recommandons d'utiliser un PCF8574A pour les entrées et un autre pour les sorties.

**Sous STRATON, voici comment procéder pour créer une variable de sortie I2C:**

Commencez par créer une variable I2Ctest par exemple, de type DINT et utilisant le profile XB10\_IICBUS :



## XB10 – Manuel utilisateur

Chaque carte et composant I2C présent sur le bus doit posséder sa propre adresse I2C. Les composants de la famille PCF8574A sont tous obligatoirement dotés d'une adresse de base 0x40. Ensuite, 3 pins du composant permettent de créer des variantes pour chaque puce de 0x40 à 0x4E. Le composant choisi pour notre exemple se situe à l'adresse 0x42, soit 66 en décimal.

Il importe ensuite de préciser le type de périphérique I2C qui est concerné par la variable, à savoir « PCF8574A\_OUTPUT » dans notre exemple. Nous verrons par la suite qu'il existe 2 autres types.

Enfin, souvenez-vous que le PCF8574A est un périphérique 8 bits, soit de longueur 1 octet (LENGTH = 1). Dans notre exemple nous avons un relais qui se trouve connecté sur le premier bit de poids faible de cet octet (index 1 de la sortie, l'index étant compris entre 1 et 8 inclus).

Vous pouvez alors créer une seconde variable pour cet exemple, afin de piloter un second relais. Une fois l'application compilée sous STRATON WORKBENCH et transférée sur la cible, vous pourrez piloter directement vos relais en mettant à 1 et à 0 les 2 variables « I2Ctest » et « I2Ctest2 » :

Nom	Valeur	Propriétés	Type
demo			
Variables globales			
background	1	XB10_LCD <syb> (POS...	DINT
bargraph1	5	XB10_LCD <syb> (POS...	DINT
DEBUG_AV	0	XB10_DBG <syb> (VAL...	DINT
DEBUG_MAX	40	XB10_DBG <syb> (VAL...	DINT
GraphDot	64	XB10_LCD <syb> (POS...	DINT
histo	5	XB10_LCD <syb> (POS...	DINT
histo2	6	XB10_LCD <syb> (POS...	DINT
I2Ctest	1	XB10_IICBUS <syb>(A...	DINT
I2Ctest2	0	XB10_IICBUS <syb> (A...	DINT
LED_RED	1	XB10_LCD <syb> (POS...	DINT

### **Sous STRATON, voici comment procéder pour créer une variable d'entrée I2C:**

Commencez par créer une variable I2Cinput par exemple, de type DINT et utilisant le profile XB10\_IICBUS :

Public

Embarquer les symboles  
 Repère numérique

XB10\_IICBUS (\*IIC peripheral management\*)

Nom	Valeur	Description
ADDRESS	68	
INDEX	2	
LENGTH	1	
TYPE	'PCF8574A_INPUT'	

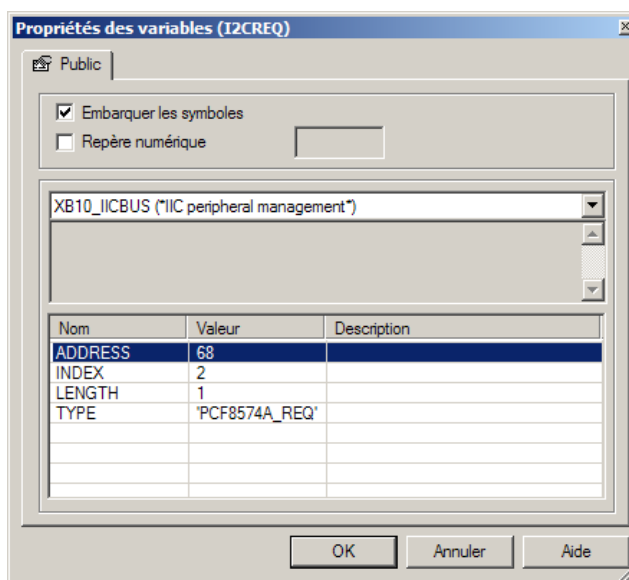
OK Annuler Aide

## XB10 – Manuel utilisateur

Nous utilisons à présent un second composant PCF8574A pour les entrées. Celui-ci possède l'adresse hexadécimale 0x44, soit 68 en décimal. Son type a été changé en « PCF8574A\_INPUT » afin de stipuler à STRATON que nous souhaitons renseigner la variable selon l'état d'un bit sur l'octet du PCF8574A (en l'occurrence la seconde entrée).

Comme nous l'avons vu en introduction, le bus I2C est plutôt destiné à des actions peu fréquentes, et ceci pour des raisons liées à la charge du bus I2C. En effet une scrutation trop systématique du bus (pour lire les entrées d'un clavier) va considérablement freiner le système et ceci même si aucune entrée n'a changé.

Il importe donc sous STRATON de créer une seconde variable de type « PCF8574A\_REQ » qui sera en charge de demander à l'automate d'accéder à un périphérique en lecture sur le bus I2C :



Nous avons ici créé une variable I2CREQ en charge de demander une lecture de toutes les entrées du périphérique PCF8574A dont l'adresse est 0x44 (68 en décimal). Par la suite, toutes les variables de type PCF8574A\_INPUT correspondant à cette adresse seront automatiquement mises à jour. Vous pourrez ainsi par exemple connaître l'état de 8 entrées alarmes chaque seconde en mettant simplement à « 1 » toutes les secondes la valeur de la variable I2CREQ.

Si l'opération s'est bien déroulée (le périphérique répond à l'adresse 68) alors la variable sera remise à la valeur 0, sinon à la valeur -1 (65535 pour le type DINT).

Au final, nous ajoutons du code à notre application afin de fermer et d'ouvrir alternativement les relais de notre carte de sortie :

```
IF ((bargraph1 > 25) AND (bargraph1 < 75)) THEN
I2Ctest2 := 1;
ELSE
I2Ctest2 := 0;
END_IF;

IF ((bargraph1 > 0) AND (bargraph1 < 50)) THEN
I2Ctest := 1;
ELSE
I2Ctest := 0;
END_IF;
```

## Cartes d'extension sur bus CAN

Le bus CAN est un bus série asynchrone développé par la société BOSCH, qui, tout comme le bus RS485, transmet uniquement les données sur une paire de fils en mode différentiel, sans transmettre un signal d'horloge associé. Ceci implique que toutes les interfaces connectées à un même bus CAN soient impérativement synchronisée sur la même vitesse de communication (baudrate). Une trame CAN contient 8 octets de données utiles et un certain nombre d'octets contenant un identifiant spécifique au message (CANID).

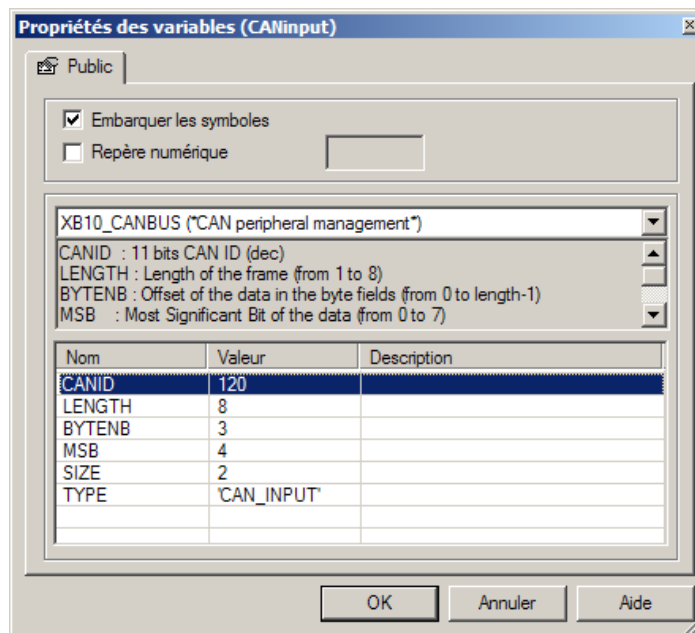
Contrairement au bus I2C que nous avons pu voir précédemment, les périphériques sur le bus CAN peuvent à tout moment « prendre la main » sur le bus et émettre des informations. Le bus CAN est donc un bus multi-maître ou chacun reçoit et émet des informations à sa convenance.

Si vous connectez des périphériques CAN sur le XB10, ce dernier recevra donc des informations issues des différents capteurs et mettra à jour les variables STRATON associée instantanément.

De même le XB10 peut émettre des trames CAN, afin, par exemple, d'ouvrir ou de fermer des relais d'une carte d'extension sur bus CAN.

### **Sous STRATON, voici comment procéder pour créer une variable d'entrée CAN:**

Commencez par créer une variable CANinput par exemple, de type DINT et utilisant le profile XB10\_CANBUS :



Le paramétrage d'une entrée CAN peut sembler complexe de prime abord mais elle permet en réalité d'utiliser des variables CAN dans n'importe quelle circonstance. Tout d'abord il faut considérer que le bus CAN est utilisé pour s'interfacer avec des périphériques bien plus complexes qu'avec un bus I2C par exemple. En effet vous pouvez interfacer des cartes d'extension CAN pilotées par processeur et capable d'envoyer plusieurs informations (température, état des entrées/sorties numériques, alarmes, consommation instantanée, ...) soit au travers d'une seule trame ou bien au travers d'autant de trames qu'il existe d'information à fournir.

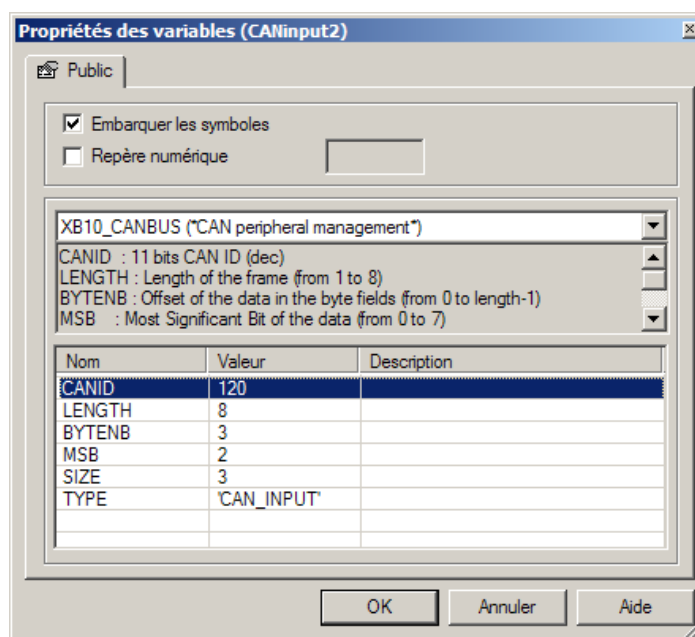
## XB10 – Manuel utilisateur

Pour réduire la charge du bus c'est bien souvent la première solution qui est choisie : transmettre toutes les informations au sein d'une seule et même trame dont l'identifiant sera le numéro spécifique de la trame. Avec un analyseur de bus CAN il sera alors simple d'établir la provenance de chaque trame sur le bus.

C'est aussi la topologie qui a été retenue pour transmettre en permanence les informations de sorties dans les véhicules (voitures, camions, véhicules agricoles, ...). Du coup il devient impératif de pouvoir stocker toutes ces informations dans une seule trame CAN de 8 octets de données utiles, et donc des positionner les données sur quelques bits seulement dans un octet et de spécifier son emplacement à l'aide de masque, d'indice d'octet dans la trame, ...

Dans l'exemple précédent nous avons considéré une trame dont l'identifiant est 0x078 (120), et qui est composée de 8 octets. Au sein de cette trame, nous recherchons une information située dans l'octet N°3, d'une largeur de 2 bits et dont le bit de poids fort est le bit N°4. Les masques seront automatiquement appliqués à la trame CAN afin de positionner la variable STRATON avec la bonne valeur.

Nous pouvons à présent créer une seconde variable CANinput2, qui sera positionnée sur le même octet de la trame 120, mais cette fois-ci en occupant les bits N°0, 1 et 2 (MSB=2, SIZE=3) :



### ***Sous STRATON, voici comment procéder pour créer une variable de sortie CAN:***

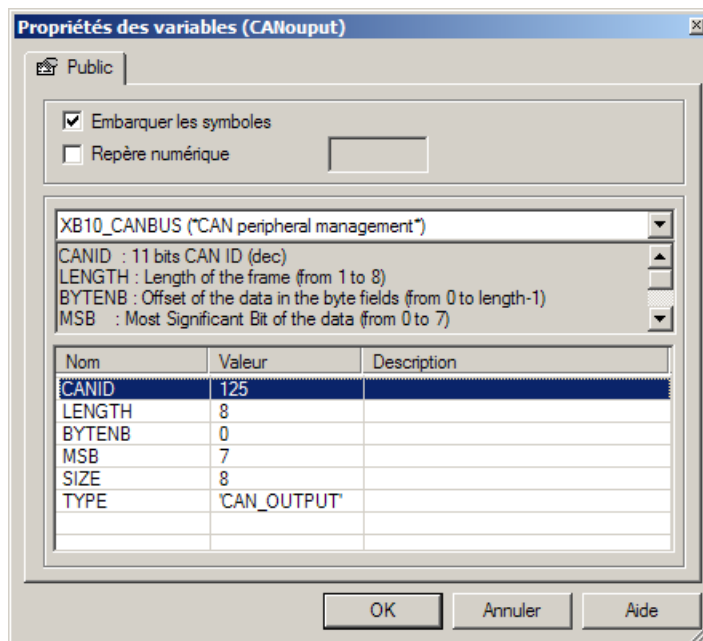
Le principe d'avoir plusieurs informations réparties au sein d'une trame de 8 octets est fort bénéfique quant à la réduction de la charge du bus CAN, mais cela possède un inconvénient majeur lors de l'émission de données sur ce bus.

En effet votre automate risque de devoir aussi envoyer des trames de consigne (fermeture de relais, ordre à envoyer à un calculateur, requête afin d'obtenir des informations, ...), et ces informations peuvent elles-aussi être contenues au sein d'une seule et même trame CAN. Par exemple il arrive souvent que certaines trames CAN soient des trames de requêtes contenant à la fois le numéro de la requête sur le premier octet, un sous-indice sur le seconde octet, la longueur de la requête sur le

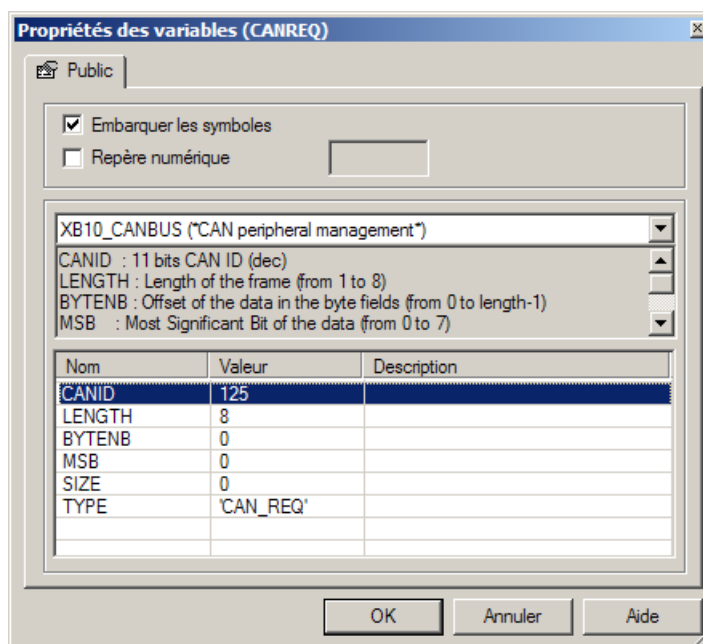
## XB10 – Manuel utilisateur

troisième, puis des données de consigne par exemple. Or, ces données sont chacune des variables STRATON et peuvent être amenées à changer à chaque cycle. La trame CAN ne doit donc pas être transmise à chaque changement d'une des variables mais bel et bien lors de l'envoi d'un ordre de transfert.

Commencez par créer une variable CANouput par exemple, de type DINT et utilisant le profile XB10\_CANBUS :



La variable CANouput tiendra sur un octet (MSB=7, SIZE=8) et occupera la place du premier octet de la trame 125 (BYTENB=0). Notez bien que toute modification de la variable n'aura aucun effet sur le fait d'émettre ou non une trame CAN d'identifiant 125. Pour cela il vous faudra créer une variable de type « CAN\_REQ » comme suit :



## XB10 – Manuel utilisateur

Seuls les champs « CANID », « LENGTH » et « TYPE » sont importants pour ce type de variable CAN. Ils signifient au XB10 d'envoyer la trame d'identifiant 125 lorsque la variable passe de 0 à 1. Si la trame CAN a été envoyée sans rencontrer d'erreur, la variable CANreq aura été automatiquement remise à 0, dans le cas contraire à -1 (65535 pour un DINT).

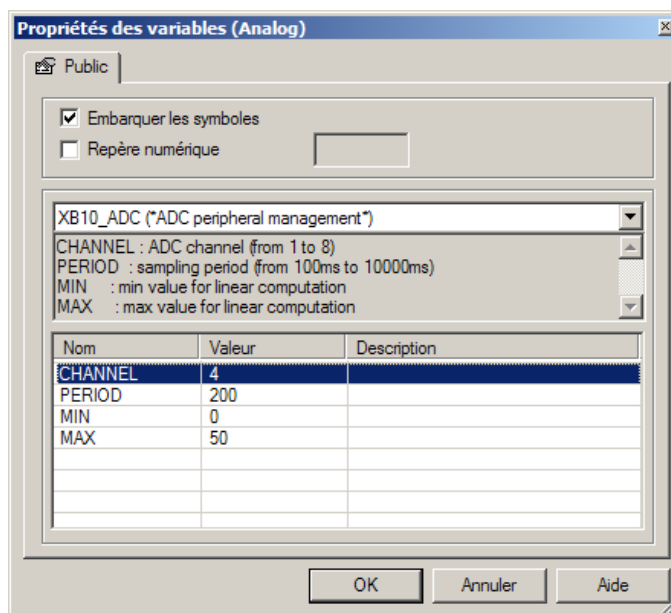
### Conversion analogique / numérique

Le XB10 dispose de 8 ports ADC (Analog to Digital Converter). Ces 8 entrées ne sont pas protégées en tension, elles doivent donc être manipulées avec beaucoup de précaution. Si vous souhaitez créer votre propre carte d'interface en utilisant ces entrées analogiques, pensez à adapter une électronique adéquate avec des écrêteurs de tension (par diode par exemple).

Le convertisseur ne permet pas la mesure de tension négative et ne peut accepter une tension d'entrée de plus de 2,5V DC.

#### **Sous STRATON, voici comment procéder pour créer une variable d'entrée ADC :**

Commencez par créer une variable Analog par exemple, de type DINT et utilisant le profile XB10\_ADC :



Parmi les différents paramètres à renseigner, se trouve bien évidemment le numéro du canal utilisé, pouvant aller de 1 à 8 inclus. La valeur électrique mesurée sera comprise entre 0V et 2,5V, mais le profile XB10\_ADC permet de faire une mise à l'échelle automatique.

Pour une valeur à mini-tension (1,127V) et des seuils MIN et MAX de 0 et 50 respectivement, vous obtiendrez une valeur de variable de 25 (la moitié). Si vous souhaitez ajouter un offset, modifiez le seuil MIN à 10 par exemple et le seuil MAX à 60. La valeur de la variable vaudra alors 35. Ceci vous permet de vous interfacer directement avec des capteurs comme des sondes de températures.

## XB10 – Manuel utilisateur

Enfin vous devez préciser une période de rafraîchissement en nombre de cycle. Ici nous avons rentré une période de 200 cycles, avec un cycle à 10 ms. Cela veut donc dire que la variable sera mise à jour toutes les 2 secondes.

### Gestionnaire d'entrées

Le XB10 dispose de 4 entrées numériques. Ces entrées sont accessibles depuis le connecteur DB9 femelle de bord de carte :

- 1 : GND
- 4 : INPUT3
- 5 : INPUT1
- 8 : INPUT4
- 9 : INPUT2

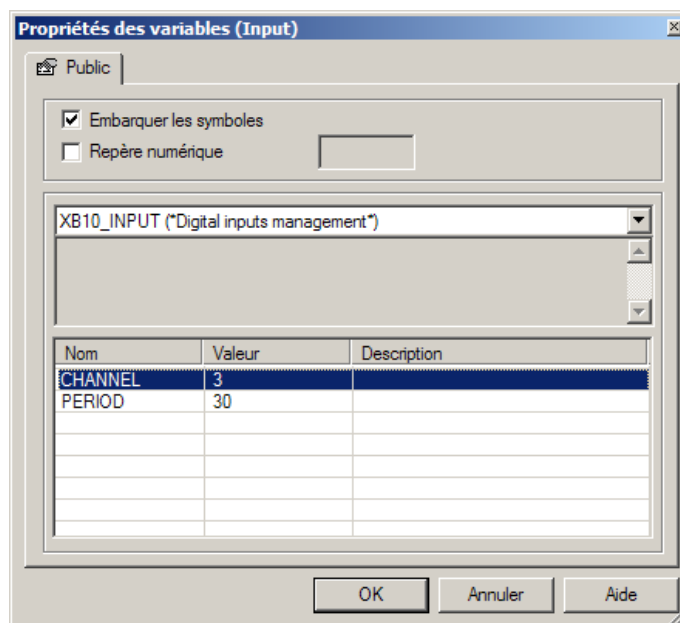
Ces entrées numériques sont directement connectées au bus du processeur, et sont lues instantanément sans nécessiter un contrôleur d'entrées/sorties. Elles ne sont pas latchées mais bufferisées, ce qui signifie que le bus du processeur est protégé mais que l'état n'est pas mémorisé : si tôt le signal relâché, si tôt sa valeur change sans que son état précédent soit conservé.

Les entrées possèdent une résistance de pull-up de 4,7Ko tirée au 5V. Ceci correspond à l'état logique « 0 » : ainsi une variable straton vaudra « 0 » si rien n'est connecté aux entrées numériques. Si vous tirer l'entrée à la masse (disponible sur la DB9 en broche 1), vous verrez la valeur électrique passer à 0V et la variable straton passer à l'état « 1 », indiquant qu'un signal a été détecté.

Ceci vous permet typiquement de connecter sur les 4 entrées 4 interrupteurs à la masse. Lorsque les contacts seront positionnés, la variable associée passera automatiquement à 1.

### **Sous STRATON, voici comment procéder pour créer une variable d'entrée INPUT :**

Commencez par créer une variable Input par exemple, de type DINT et utilisant le profile XB10\_INPUT :



## XB10 – Manuel utilisateur

Parmi les différents paramètres à renseigner, se trouve bien évidemment le numéro de l'entrée utilisée, pouvant aller de 1 à 4 inclus. La valeur électrique mesurée sera comprise entre 0V et 5V (5V par défaut si rien n'est connecté), mais la variable associée pourra prendre les valeurs 0 ou 1. Vous devez ensuite fournir une période en nombre de cycle. Ici nous avons choisi une période 30 avec un cycle à 10ms, ce qui signifie que l'entrée 3 sera inspectées et la variable rafraîchie 3 fois pas seconde.

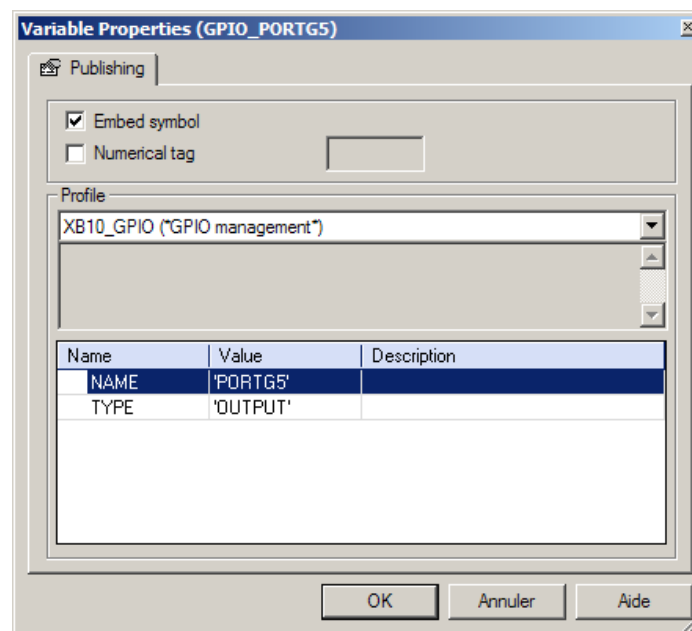
### Gestionnaire générique d'IO (GPIO)

Le XB10 vous permet de manipuler n'importe quel IO du CPU. Il est donc de la responsabilité du développeur de ne manipuler que les IO libre sur la cible dont il dispose. Les GPIO issus du CPU ont un potentiel de 3.3V. Nous vous recommandons de les protéger par une électronique adaptée telle qu'un latch ou un simple buffer qui fera office de fusible en cas d'anomalie.

Les GPIO doivent physiquement être protégée contre les ESD, les court-circuits et les surtensions.

#### **Sous STRATON, voici comment procéder pour créer une variable de type GPIO :**

Commencez par créer une variable GPIO\_PORTG5 par exemple, de type DINT et utilisant le profile XB10\_GPIO :



Parmi les différents paramètres à renseigner, se trouvent le nom de la broche du processeur ainsi que le type de la broche (« INPUT » ou « OUTPUT »). Le nom du signal doit toujours commencer par « PORT », puis comporter une lettre qui va de « A » à « H » pour un processeur S3C2410 par exemple, et enfin un numéro correspondant au numéro de la broche sur le port.

Référez vous à la documentation du processeur ainsi qu'au schéma électrique de votre carte afin de n'utiliser que les GPIO libre de votre carte. La manipulation d'un GPIO non libre peut entraîner des disfonctionnement de votre carte voir un reset de la cible.

## XB10 – Manuel utilisateur

### Gestionnaire RS232 / RS485

Vous souhaitez peut-être transmettre des données via un bus RS232 ou RS485. Bien plus encore, vous pouvez utiliser un protocole par dessus la liaison RS232 ou RS485, tel que le protocole MODBUS. Straton vous permet très facilement de paramétrer vos requêtes MODBUS afin de récupérer toutes les données contenus dans vos périphériques MODBUS distants.

Pour transmettre des données au travers d'une liaison RS232 ou RS485, il vous faut créer une variable « Serlo » et brancher sur cet objet des variables qui vous permettront de le piloter (RUN, SND, ...).

**Sous STRATON, voici comment procéder pour créer une variable de type Serlo :**

Créez comme suit les variables MySer, RUN, SND, CONF, OPEN, DATASND :

MySer		SerIO		<input type="checkbox"/>	
RUN		BOOL		<input type="checkbox"/>	FALSE
SND		BOOL		<input type="checkbox"/>	
CONF		STRING(25...		<input type="checkbox"/>	'com1,9600,N,8,1,485'
OPEN		BOOL		<input type="checkbox"/>	
DATASND		STRING(25...		<input type="checkbox"/>	
Tab		USINT	[0..3]	<input type="checkbox"/>	USINT#116,USINT#65,U

La variable la plus importante est la variable CONF, une chaîne de caractères qui décrit la façon dont est paramétrée le port COM choisi. Le premier port COM est appelé « com1 », le second « com2 », ... Le dernier paramètre permet de spécifier si le mode de communication est le mode 485 ou RS232, en d'autres termes, si le contrôleur du processeur s'interface avec un transceiver de type MAX232 ou MAX485. Dans le cas d'un MAX485, il vous faudra un signal additionnel, le signal RTS du port sélectionné, qui mettra le MAX485 en mode émission ou réception (la réception est le mode par défaut).

Nous vous conseillons d'utiliser le premier port COM (com1) en mode RS485. Toutefois, la cible Linux que vous utilisez se sert déjà de ce port COM comme console par défaut. **Pour la libérer, il vous faut paramétrer le démarrage de Linux depuis le BIOS.** Contacter Pragmatec pour le faire si vous connaissez pas la procédure.

Enfin, il ne vous reste qu'à piloter le port COM depuis Straton en commençant par ouvrir le port :

```
MySer (RUN, SND, CONF, DATASND);
Tab[1] := 65 + ANY_TO_USINT(TempGauge);
ArrayToStringU(Tab, DATASND, 2);
OPEN := MySer.OPEN;
```

Passer la variable RUN à 1 pour ouvrir le port COM. Si l'ouverture s'effectue sans encombre, la variable OPEN passera elle-aussi à 1. Passer à présent la variable SND à 1 pour commencer d'émettre le contenu du buffer DATASND. Ce transfert s'effectuera ensuite cycliquement.



Bâtiment EARHART  
ZAC Grenoble Air Parc  
38590 St Etienne de St Geoirs - France  
[www.pragmatec.net](http://www.pragmatec.net)

## XB10 – Manuel utilisateur

### Cartes d'extension sur Ethernet

Différentes cartes d'extension sur bus Ethernet existent sur le marché. Le Workbench STRATON permet de gérer une sur couche MODBUS sur Ethernet qui vous permettra de piloter tout un ensemble de cartes d'entrées / sorties à l'aide d'une passerelle Ethernet connectée au XB10.

Contactez la société COPALP pour connaître la liste des fabricants et des revendeurs français qui proposent ce type de périphériques compatibles avec l'automate STRATON.

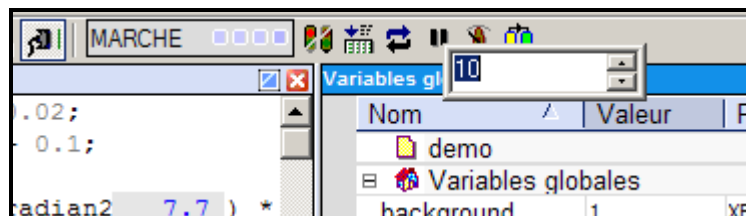


## 5 Remarques

Ce dernier chapitre a pour but de présenter un certain nombre d'information à toutes fins utiles.

### Temps de cycle

Sous le Workbench STRATON il est possible de modifier dynamiquement le temps de cycle de votre automate. Ce temps est primordial dans la conception de votre automate, car il stipule le temps imparti au processeur pour exécuter la totalité du code de votre automate et aussi de parcourir l'ensemble de vos entrées / sorties afin d'effectuer des mises à jour en cas de besoin.

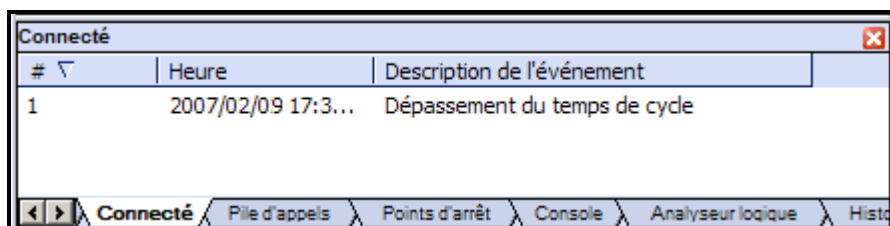


Nous avons choisi pour nos exemples un temps de cycle de 10ms. Même si le code est en totalité exécuté toutes les 10ms, les variables ne changent pas pour autant à chaque cycle. Nous avons pu voir précédemment que certaines variables du XB10 sont caractérisées par une période, et que la plupart des variables de sorties sont mises à jour lorsque leur valeur change uniquement.

Plus votre programme sera volumineux et plus il sera difficile pour le XB10 se tenir des temps de cycles de 10ms, qui sont tout de même les temps de cycle les plus courts qu'il soit possible d'atteindre sur une telle plate-forme. Lorsque le XB10 n'est plus capable de respecter un temps de cycle particulier, du à une charge ponctuelle, il le fait savoir via l'interface du Workbench en indiquant un dépassement de temps de cycle.

### Cas de dépassement de temps de cycle

Les dépassements de temps de cycle sont rares dans une configuration standard du XB10. Toutefois lorsque vous utiliserez un LCD graphique avec des basculements de pages (pour circuler dans des menus par exemple) vous risquez d'obtenir parfois un dépassement de temps de cycle ponctuel.

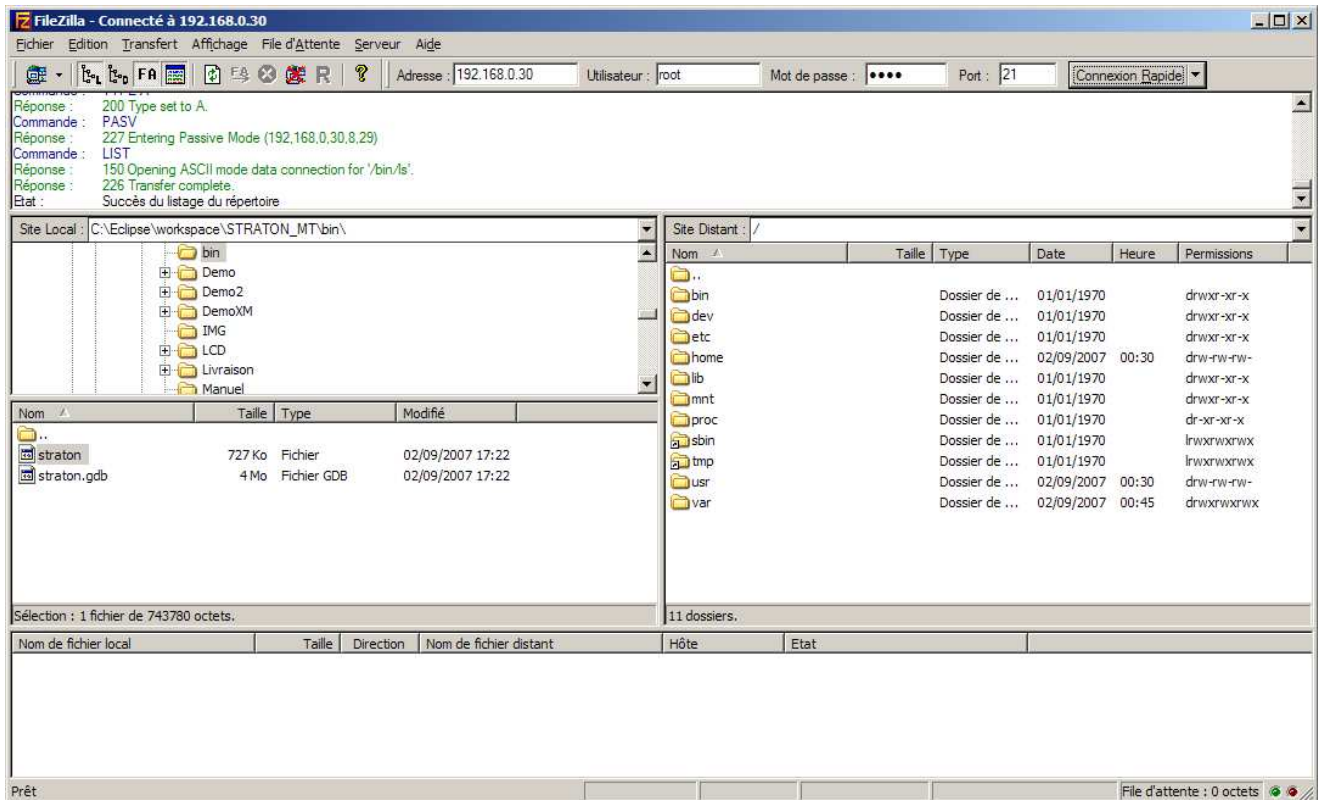


Pour cela un mécanisme spécial a été conçu pour le XB10, pour lequel l'affichage des objets graphiques se fait tant que le cycle le permet. Dans le cas contraire certains objets ne seront pas affichés ou partiellement : faites des essais à différents temps de cycle.

## XB10 – Manuel utilisateur

### Mis à jour de l'application

Le XB10 peut être mis à jour à l'aide d'un outil de transfert FTP tel que FileZilla :



Il vous suffit de connecter votre ordinateur au XB10 à l'aide d'un cordon RJ45 croisé et de rentrer les paramètres de connexion suivants :

**Adresse :** 192.168.0.30 (par défaut)  
**Utilisateur :** root  
**Mot de passe :** root  
**Port :** 21

Cliquez ensuite sur « Connexion rapide » pour obtenir l'affichage ci-dessus ou l'ensemble des répertoires du XB10 seront vu comme au travers d'un exploreur. Double cliquez sur le répertoire « home », puis placez-vous sur la fenêtre de gauche à l'endroit de votre disque dur ou se trouve votre runtime STRATON. Double cliquez cette fois sur votre runtime de mise à jour, le transfert va s'effectuer automatiquement.

Lorsque la mise à jour est terminée, vous pourrez redémarrer le XB10 pour que le nouveau runtime prenne effet immédiatement.