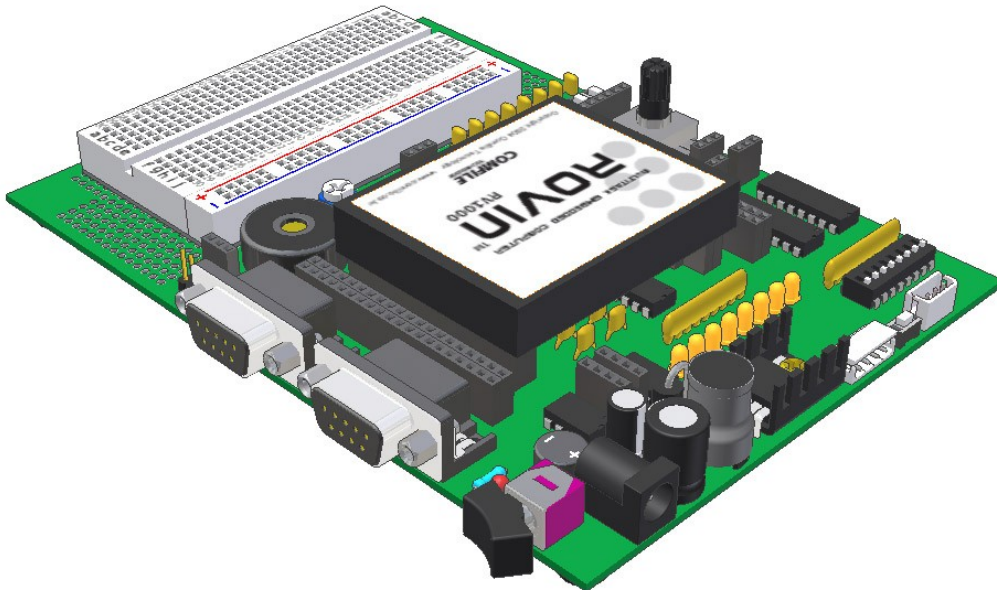


# Formation M.I.M.E

Master Informatique

**Spécialité : Informatique des Systèmes Embarqués**

## Travaux pratiques de programmation du $\mu$ C ARM7 Pour applications Embarqués



Année 2010

A.A.C

### Présentation

Présentation

Raccordement du câble

### Description générale

Les DIP-SWITCHs

Exécuter un programme

Créer un PROJET

Utiliser le mode DEBUG

### Démarrage rapide

### Travaux pratiques : programmation du $\mu$ C Arm7

Programme 1 : Gestion interruptions

Programme 2 : Conversion analogique/numérique

Programme 3 : Piloter un afficheur

Programme 4 : Gestion compteur

Programme 5 : Comparateur analogique.

Programme 6 : Ports PPI en "E/S"

Programme 7 : Ports EXP en "E/S"

Programme 8 : Gestion UART (RS232)

Programme 9 : Pilotage EEprom I2C

Programme 10 : Réaliser une horloge

Programme 11 : 74HC165(Entrées) / 74HC595(Sorties)

Programme 12 : Application multitâche

Programme 12 : Pilotage sonar

Programme 13 : Pilotage Accéléromètre

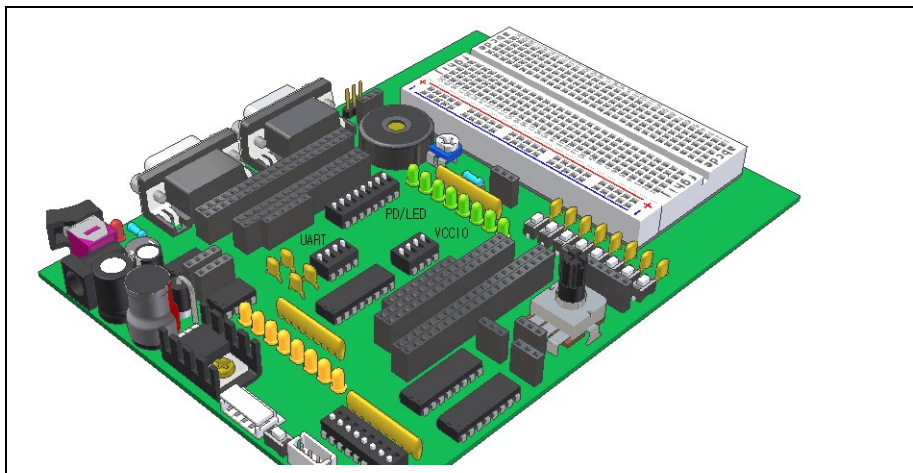
## Présentation de la carte

La platine "Quick.Start.Board" est spécialement conçue pour faciliter le développement et les expérimentations à l'aide du module ROVIN™. Bien que d'apparence simple, elle vous permettra de tester la plupart des fonctionnalités et possibilités du ROVIN™.

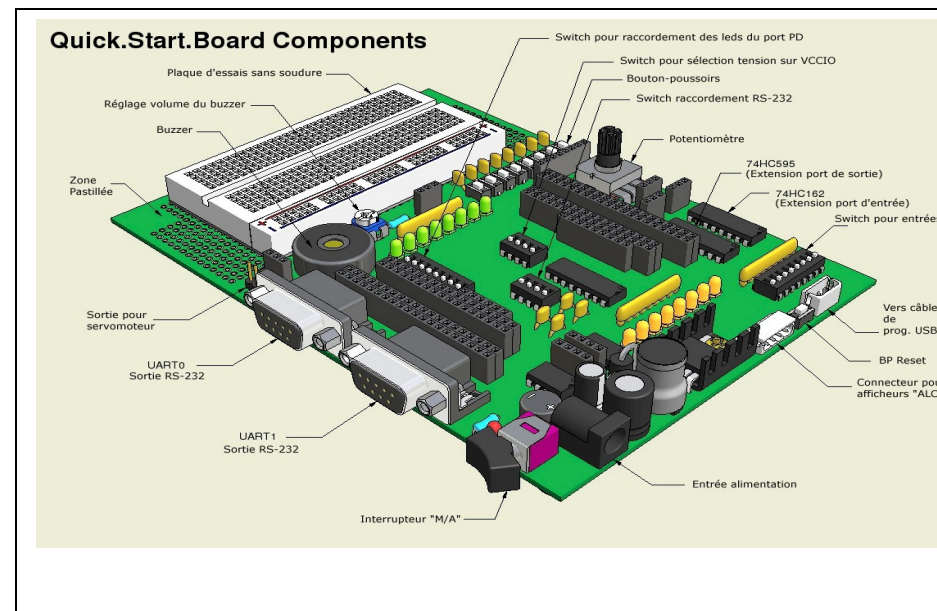
La platine "Quick.Start.Board" est livrée avec une plaque de connexion sans soudure qu'il vous sera possible d'apposer ou non sur le côté du circuit selon vos préférences de travail. Si vous n'utilisez pas la plaque de connexion sans soudure, vous disposerez alors d'une large zone de développement pastillée qui pourra recevoir des composants additionnels.

La platine est équipée d'un étage de régulation qui pourra fournir jusqu'à 1 A. Il vous faudra juste acquérir une alimentation externe de 8 à 9 V (non livrée) - Nous proposons le modèle "PSU10R" par exemple. Il n'est pas nécessaire de vous soucier de la polarité de cette alimentation (la platine intègre en effet un pont diode). La plupart des périphériques de la carte disposent de connecteurs de raccordement. L'utilisateur pourra ainsi au grès de ses réalisations utiliser de simples fils pour relier ces derniers aux ports sur module ROVIN™.

### La platine "Quick.Start.Board" (sans la platine d'essai)



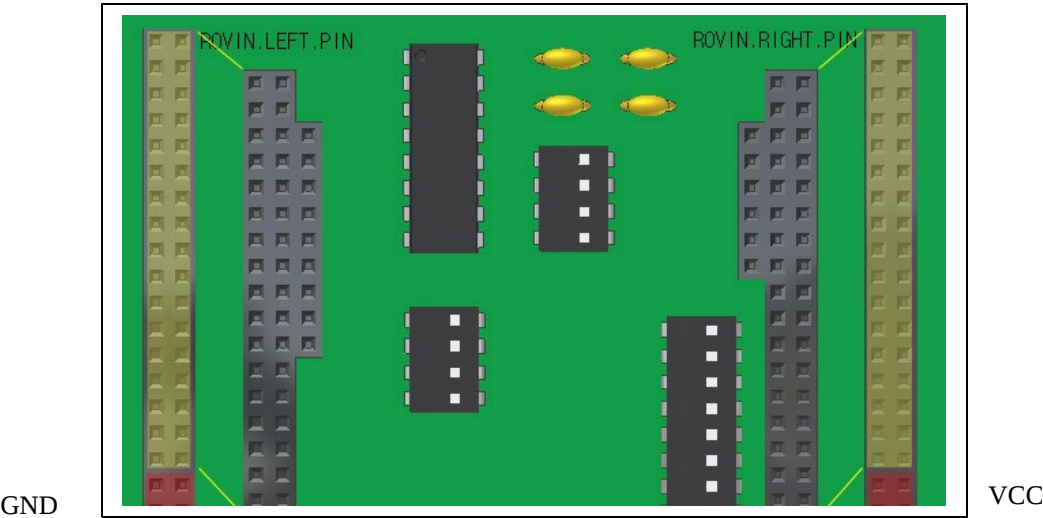
La platine de connexion sans soudure dispose de 2 rangées pour la reprise des alimentations: + et - (masse). Ces signaux pourront être récupérés sur certains connecteurs de la platine "Quick.Start.Board".



Comme vous pouvez le constater la platine "Quick.Start.Board" dispose de très nombreux accessoires et composants spécifiques. Les Leds près du buzzer pourront être reliées au port PD en utilisant le DIP Switch prévu à cet effet. Ces dernières seront très utiles pour tester et expérimenter vos différents programmes. Les autres LEDs (à côté du régulateur) servent à tester les sorties du circuit d'extension de port 74HC595.

Note: les entrées logiques de commande du circuit d'extension de port 74HC595 étant initialement en l'air, il se peut que sous l'influence de signaux parasites, les Leds de visualisation associées à ce dernier s'allument et s'éteignent toute seules de façon intempestive. Si vous n'utilisez pas le 74HC595, il est conseillé de "forcer" sa broche "SCK" au niveau logique 0 en la reliant à la masse.

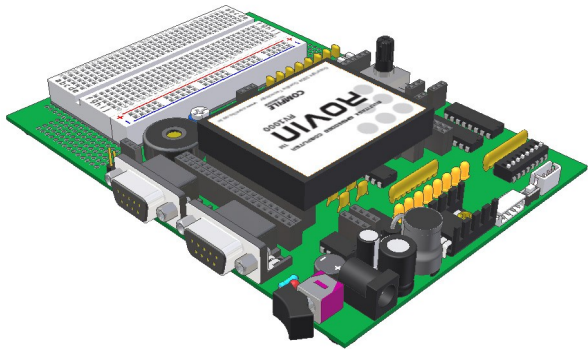
Relation entre les connecteurs femelles et les broches sur le module ROVIN



Comme vous pouvez le voir ci-dessus, les connecteurs femelles de "reprise" des broches du ROVIN™ reprennent exactement la même implantation que celui-ci (broche à broche). Les connecteurs en jaune vous permettrons ainsi de vous connecter sur n'importe quelle broche du ROVIN™ conformément au brochage de ce dernier (voir détail dans le manuel du ROVIN™ ou l'aide en ligne).

ATTENTION: Du fait qu'aucun marquage ne soit disponible sur la platine, il vous faut impérativement vérifier avec attention le brochage du ROVIN™ avant de réaliser vos connexions afin d'éviter toute erreur pouvant être préjudiciable à ce dernier. Les connecteurs femelles rouges vous permettrons de récupérer les tensions d'alimentation 0 V et 5 V afin de pouvoir par exemple les rapporter sur la plaque de connexion sans soudure

La carte équipée du module ROVIN



La représentation ci-dessus montre le module ROVIN™ inséré sur le support de la platine "Quick.Start.Board". Coupez impérativement l'alimentation de la platine "Quick.Start.Board" avant d'insérer ou de retirer le module ROVIN™ de son support sous peine de DESTRUCTION de ce dernier. Dans tous les cas insérez le module avec précaution en exerçant une force "modérée" sur les 2 côtés du ROVIN™. Dans le même ordre d'idée, retirez ce dernier avec précaution en évitant de forcer sur un seul des côtés.

Le brochage du "ROVIN"

Patte N°

1	18	50	67		
2	19	51	68		
3	20	35	44	52	69
4	21	36	45	53	70
5	22	37	46	54	71
6	23	38	47	55	72
7	24	39	48	56	73
8	25	40	49	57	74
9	26	41	58	75	
10	27	42	59	76	
11	28	43	60	77	
12	29	61	78		
13	30	62	79		
14	31	63	80		
15	32	64	81		
16	33	65	82		
17	34	66	83		

**Tableau des entrées/sorties**

N°	Nom	Fonc. sup	Désignation
1	EXPA7	ADC7	E/S générale ou conversion " A/N ".
2	EXPA6	ADC6	E/S générale ou conversion " A/N ".
3	EXPA5	ADC5	E/S générale ou conversion " A/N ".
4	EXPA4	ADC4	E/S générale ou conversion " A/N ".
5	EXPA3	ADC3	E/S générale ou conversion " A/N ".
6	EXPA2	ADC2	E/S générale ou conversion " A/N ".
7	EXPA1	ADC1	E/S générale ou conversion " A/N ".
8	EXPA0	ADC0	E/S générale ou conversion " A/N ".
9	VCCIO-PA		Tension de PULL-UP du port PA. (Moins de 5V )
10	PA7		E/S générale.
11	PA6		E/S générale.
12	PA5		E/S générale.
13	PA4		E/S générale.
14	PA3		E/S générale.
15	PA2		E/S générale.
16	PA1		E/S générale.
17	PA0		E/S générale.
18	EXPB7	INT7, CP1	E/S générale, CP1, ou interruption externe.
19	EXPB6	INT6, CT0	E/S générale, CT0, ou Interruption externe.
20	EXPB5	PWM/INT5	E/S générale, PWM3C, ou Interruption externe.
21	EXPB4	PWM/INT4	E/S générale, PWM3B, ou Interruption externe.
22	EXPB3	PWM/AIN1 (-)	E/S générale, PWM3A, Tension entrée ANCOMP (-), ou Interruption externe.
23	EXPB2	AIN0(+)	E/S générale ou Tension entrée ANCOMP (+).
24	EXPB1	TXD0	E/S générale ou TXD0. (UART)
25	EXPB0	RXD0	E/S générale ou RXD0. (UART)
26	VCCIO-PB		Tension de PULL-UP du port PB. (Moins de 5V )
27	PB7		E/S générale.
28	PB6		E/S générale.
29	PB5		E/S générale.
30	PB4		E/S générale.
31	PB3		E/S générale.
32	PB2		E/S générale.
33	PB1		E/S générale.
34	PB0		E/S générale.
35	GND(0.0V)		GND (A connecter au 0 V)
36	XRESET		RESET DU MODULE ROVIN™.
37	PC-TXD		TXD pour connexion du ROVIN™ au PC

38	PC-RXD		RXD pour connexion du ROVIN™ au PC
39	SYSTEM		Laissez libre ( Ne rien connecter sur cette broche !)
40	SYSTEM		Laissez libre ( Ne rien connecter sur cette broche !)
41	SYSTEM		Laissez libre ( Ne rien connecter sur cette broche !)
42	SYSTEM		Laissez libre ( Ne rien connecter sur cette broche !)
43	SYSTEM		Laissez libre ( Ne rien connecter sur cette broche !)
44	VCC(5.0V)		A connecter au 5.0 V.
45	3.3V OUT		Sortie 3.3 V. A n'utiliser que pour alimenter des systèmes très faible consommation.
46	SYSTEM		Laissez libre ( Ne rien connecter sur cette broche !)
47	SYSTEM		Laissez libre ( Ne rien connecter sur cette broche !)
48	SYSTEM		Laissez libre ( Ne rien connecter sur cette broche !)
49	SYSTEM		Laissez libre ( Ne rien connecter sur cette broche !)
50	PWM1C	PWM2	Broche PWM1C/PWM2 uniquement de sortie.
51	PWM1B		Broche de sortie PWM1B uniquement.
52	PWM1A		Broche de sortie PWM1A uniquement.
53	PWM0		Broche de sortie uniquement.
54	XBUS		Laissez libre ( Ne rien connecter sur cette broche !)
55	XBUS		Laissez libre ( Ne rien connecter sur cette broche !)
56	XBUS		Laissez libre ( Ne rien connecter sur cette broche !)
57	XBUS		Laissez libre ( Ne rien connecter sur cette broche !)
58	VCCIO-PC		Tension de PULL-UP du port PC. (Moins de 5V )
59	PC7		E/S générale.
60	PC6		E/S générale.
61	PC5		E/S générale.
62	PC4		E/S générale.
63	PC3		E/S générale.
64	PC2		E/S générale.
65	PC1		E/S générale.
66	PC0		E/S générale.
67	EXPD7	CT2	E/S générale ou CT2.
68	EXPD6	CT0	E/S générale ou CT0.
69	EXPD5		E/S générale.
70	EXPD4	CP0	E/S générale ou CP0.
71	EXPD3	TXD1/INT3	E/S générale, TXD1, broche Interruption externe.
72	EXPD2	RXD1/INT2	E/S générale, RXD1, broche Interruption externe.
73	EXPD1	INT1	E/S générale ou broche Interruption externe.
74	EXPD0	INT0	E/S générale ou broche Interruption externe.
75	VCCIO-PD		Tension de PULL-UP du port PD. (Moins de 5V )
76	PD7		E/S générale.
77	PD6		E/S générale.
78	PD5		E/S générale.



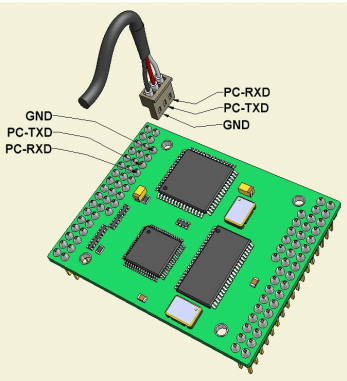
79	PD4		E/S générale.
80	PD3		E/S générale.
81	PD2		E/S générale.
82	PD1		E/S générale.
83	PD0		E/S générale.

Le câble USB connexion vers le PC



Ce câble vous permettra de télécharger vos programmes (tâches) au sein du module ROVIN™ et de déboguer ces derniers.

Avant de pouvoir l'utiliser, quittez tous les programmes en cours puis insérez le câble sur le port USB de votre PC afin que votre machine le détecte et procède à l'installation de son driver (sélectionnez l'emplacement où le programme d'installation pourra trouver le driver: généralement X:/Program Files/ROVIN/ROVIN-IDE/CABLE DRIVER).



Si votre système d'exploitation est WindowsXP™, il se peut que la fenêtre d'installation du driver vous indique que Si vous réalisez vous-même votre propre circuit imprimé, vous pouvez acquérir le connecteur mâle associé afin que vous puissiez réaliser la connexion ci-dessous.

Dans tous les cas, ne rallongez jamais la longueur du câble USB de programmation du module ROVIN™.

Les DIP-SWITCHS

Explications

La platine "Quick.Start.Board" dispose de 4 DIP switchs. Ces derniers permettent de connecter ou de déconnecter certaines broches du module ROVIN™ des composants présents sur la platine. En utilisant ces derniers, vous pourrez isoler totalement les broches du module ROVIN™.

Dip Switch	Explications
PD/LED	Lorsque ces derniers sont sur la position "ON", les entrées du port PD sont connectées sur les Leds près du buzzer. A l'inverse lorsque les Dip switchs sont dans l'autre position ,les Leds sont déconnectés du port PD.
VCCIO	Lorsque ces derniers sont sur la position "ON", les tensions VCCIO des ports PA/PB/PC/PD sont reliées au +5 V. A l'inverse lorsque les Dip switchs sont dans l'autre position, les ports sont en "Drain ouvert" et vous pourrez appliquer vous-même la tension de référence voulue (évitiez impérativement de placer les Dip Switchs sur la position "ON" si vous appliquez votre propre tension de référence sous peine de destruction possible du module ROVIN™.
UART	Lorsque ces derniers sont sur la position "ON", les broches correspondantes du ROVIN™ sont physiquement reliés aux circuits de mise à niveau des UART. Les 2 dip Switchs du haut sont dédiés à l'UART1, les 2 du bas, à l'UART0. A l'inverse lorsque les Dip switchs sont dans l'autre position, les broches correspondantes du ROVIN™ peuvent être utilisée comme des ports "E/S" à usage générale.
Entrées 74HC165	Lorsque ces derniers sont sur la position "ON", un niveau "1" est appliqué sur les entrées du circuit 74HC165. A l'inverse lorsque les Dip switchs sont dans l'autre position, un niveau "0" est appliqué sur les entrées du circuit 74HC165.

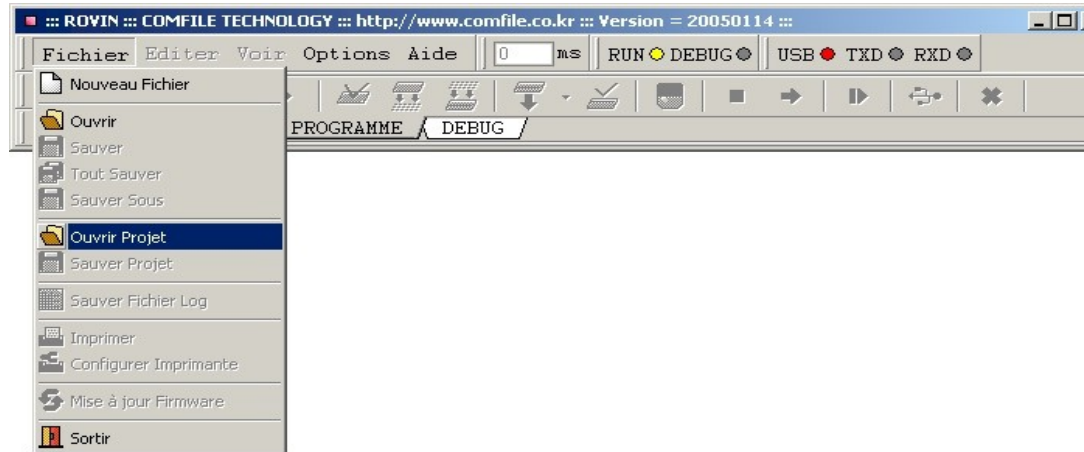
## Comment faire exécuter un programme au module ROVIN

Après avoir procédé à l'installation du logiciel de l'environnement de développement "ROVIN-IDE" et du driver du câble USB (conformément à la notice livrée avec le CD-ROM), reliez le câble USB au module ROVIN™, puis alimentez votre application. Suivez alors les instructions ci-dessous pour pouvoir apprendre à faire exécuter un programme au module ROVIN

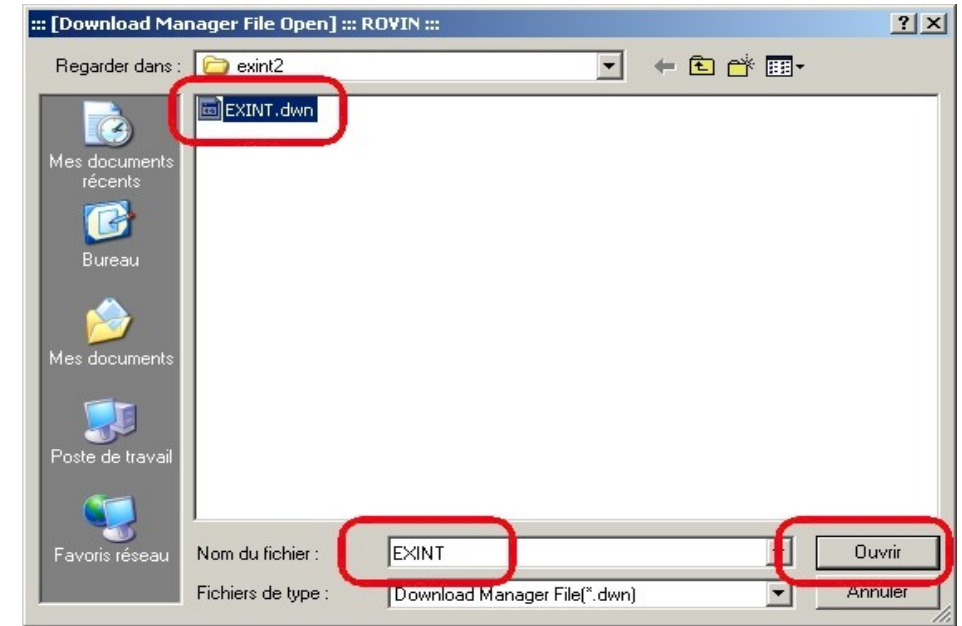
### 1. Démarrez l'environnement de développement "ROVIN-IDE"



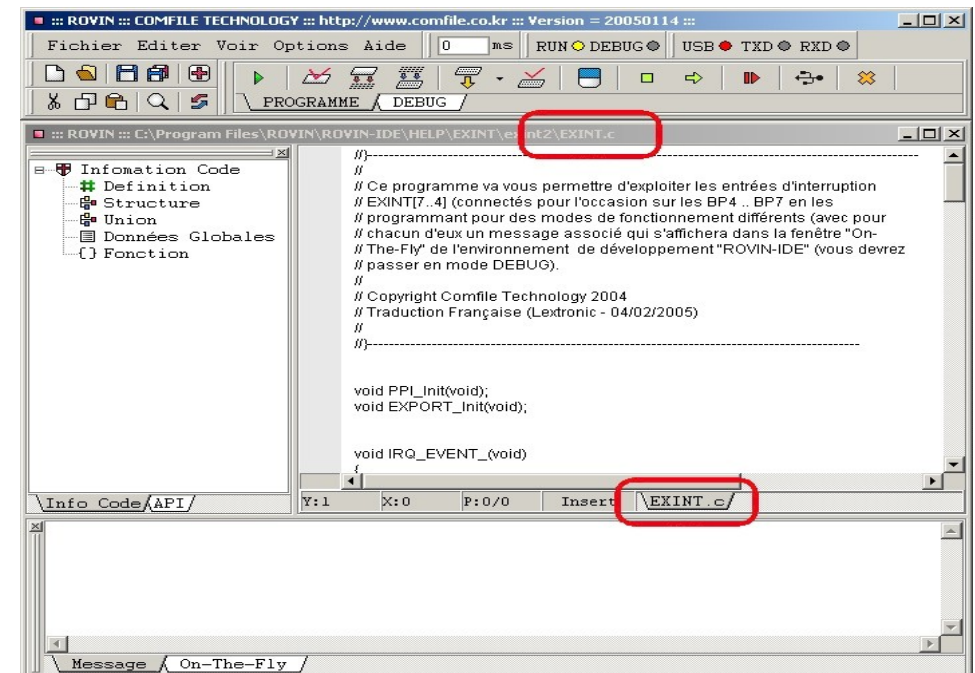
### 2. Cliquez sur "Fichier"->"Ouvrir Projet".



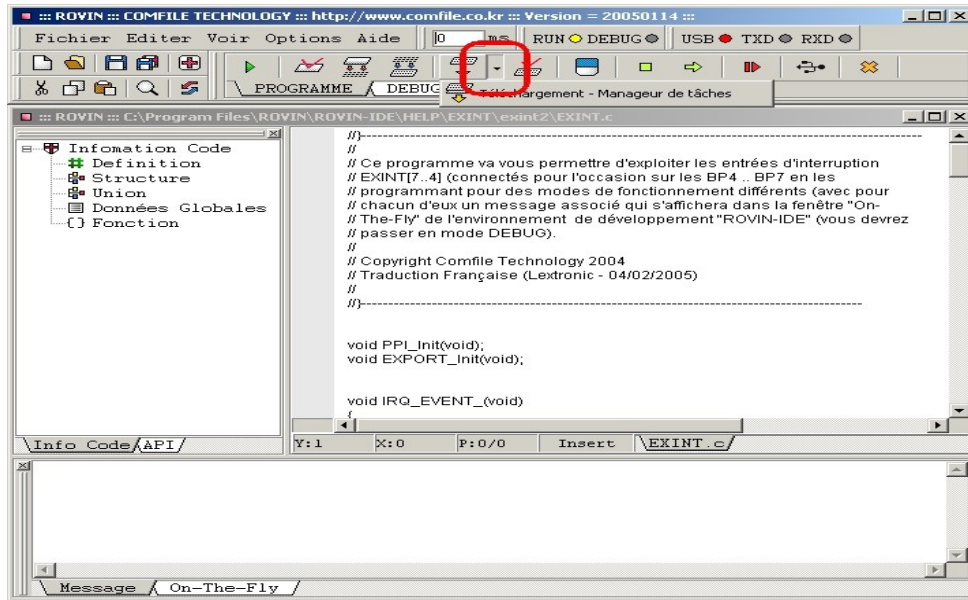
### 3. Sélectionnez un fichier "Projet" (\*.dwn) et ouvrez ce dernier



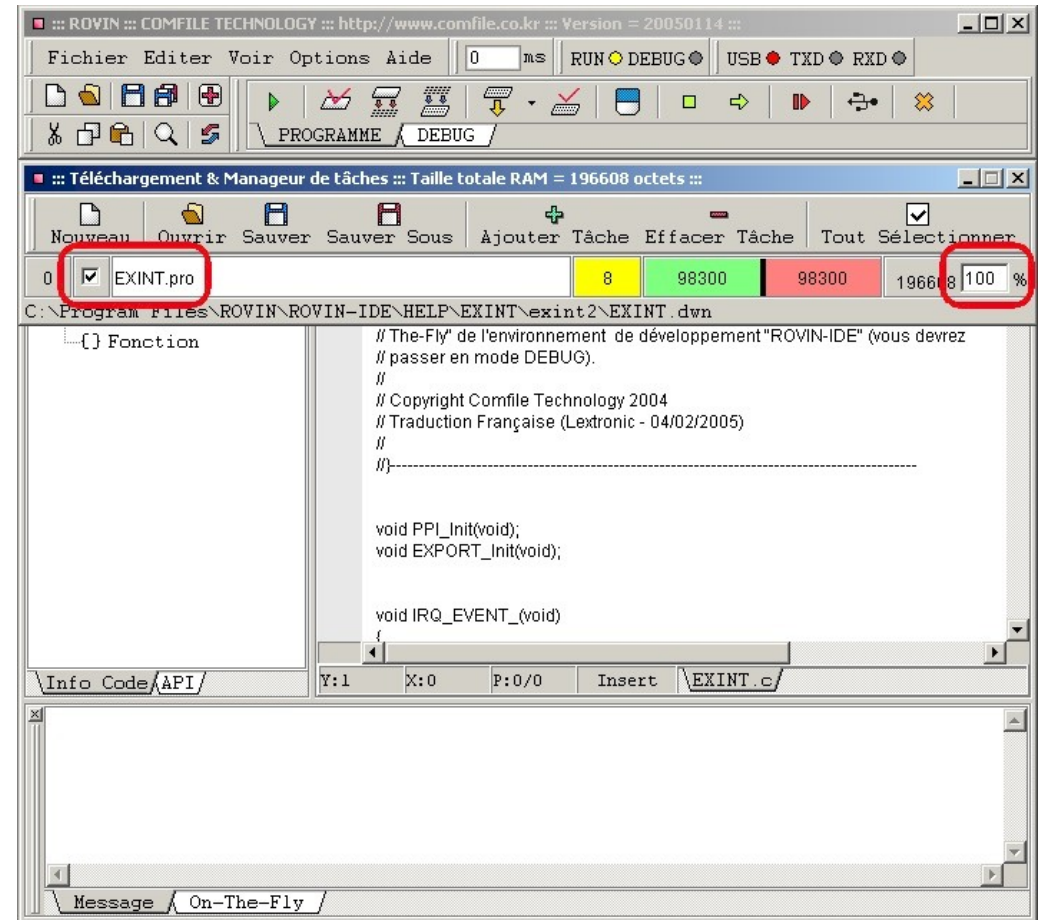
### 4. L'écran de l'environnement "ROVIN-IDE" doit alors ressembler à ci-dessus



5. Cliquez sur la petite flèche à droite du bouton "Télécharger programme(s)"

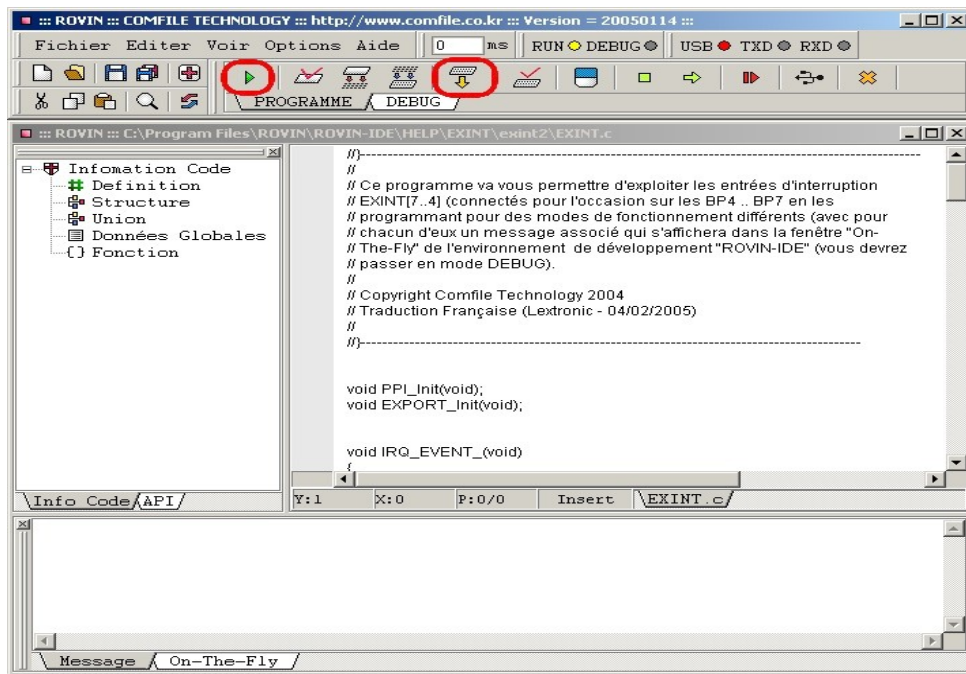



6. Vérifiez que le fichier de téléchargement est bien "coché" et que la case de droite correspondante à la répartition mémoire soit bien à 100 % (auquel cas modifiez ces dernières comme sur l'écran ci-dessous). Fermez ensuite la fenêtre de téléchargement à l'aide de la petite croix en haut à droite



7. Recompilez et téléchargez le programme au sein du ROVIN™ à l'aide du bouton .  
Si il n'y a eu aucun changement réalisé dans le code source. Transférez simplement le programme dans le module ROVIN™ à l'aide du bouton .





8. Cliquez sur le bouton Reset , afin que le module ROVIN™ exécute le programme. Si le module ROVIN™ ne réagit pas correctement, consultez la rubrique "Problèmes et solutions" disponible dans l'aide en ligne ou à la fin du manuel du ROVIN

## Créer un "Projet"

La création d'un projet vous permettra d'enregistrer tous les paramètres de votre application dans un seul et même fichier. Ce dernier contiendra: Le nom des fichiers sources de vos programmes (si vous développez une application monotâche, il ne mémorisera que le nom du fichier source principal). Le nom des fichiers exécutables (générés après une compilation) qui pourront être téléchargés dans la mémoire du ROVIN™. Les valeurs des répartitions mémoires que vous aurez définies pour chaque tâche.

- Le nom du fichier "de paramètres" utilisé pour le mode DEBUG.

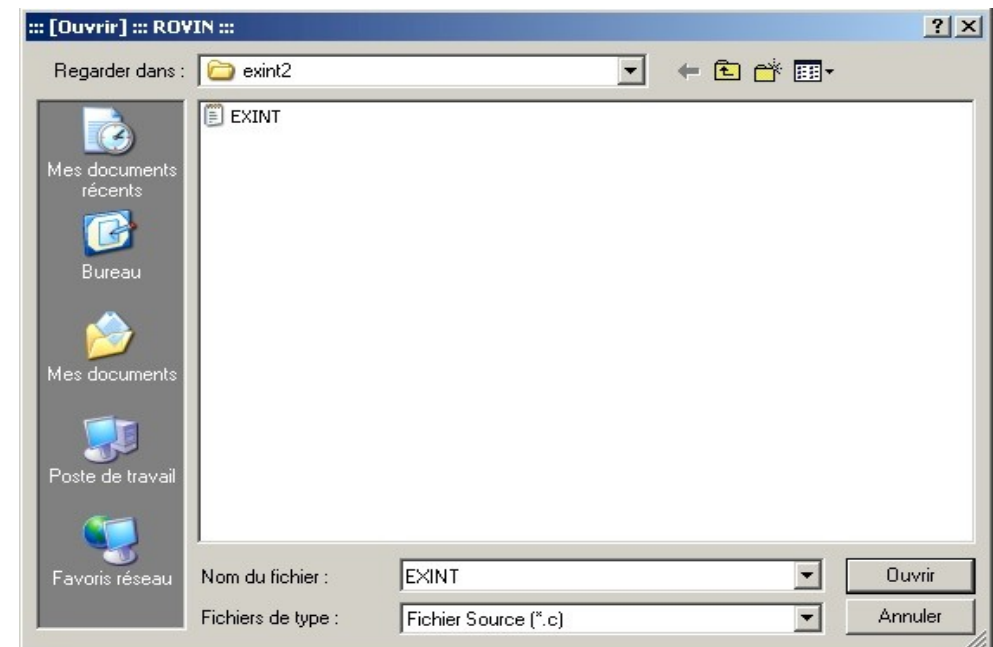
Ainsi lorsque vous chargez un projet en mémoire, l'environnement de développement "ROVIN-IDE", récupérera toutes les données relatives à votre application (l'éditeur mettra à votre disposition tous les fichiers source de votre projet, le manageur de téléchargement connaîtra quels seront les programmes potentiellement téléchargeables dans la mémoire du ROVIN™ et quelle est leur répartition mémoire associée, Le mode DEBUG pourra également être activé à volonté).

Avant de pouvoir commencer à créer le projet proprement dit, il vous faut en priorité commencer à écrire votre programme en langage "C". Le plus simple est de récupérer un fichier source existant.

Une fois sous l'environnement de développement "ROVIN-IDE", ouvrez un fichier source (ces derniers ont une extension '.c')

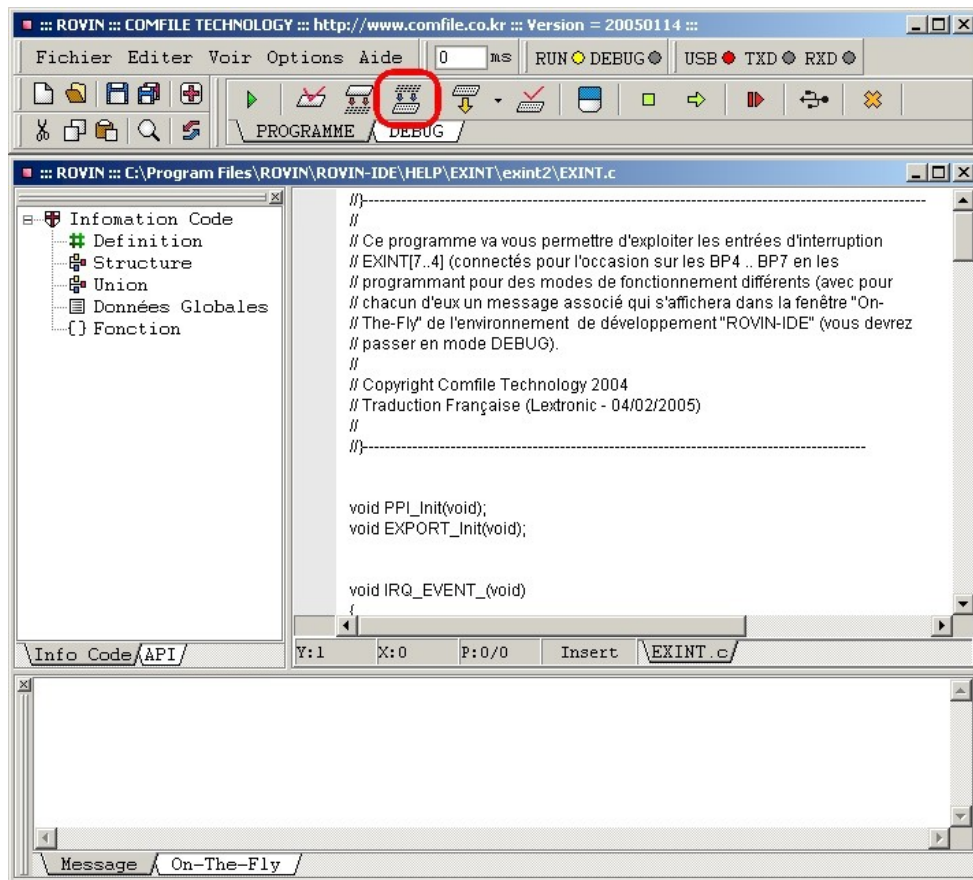


2. Sauvegardez le fichier sous un nouveau répertoire (en lui donnant si nécessaire un nouveau nom). Modifiez et adaptez alors le fichier selon les besoins de votre application (si vous en êtes à votre première utilisation, laissez le fichier en état afin d'éviter toute erreur de compilation).

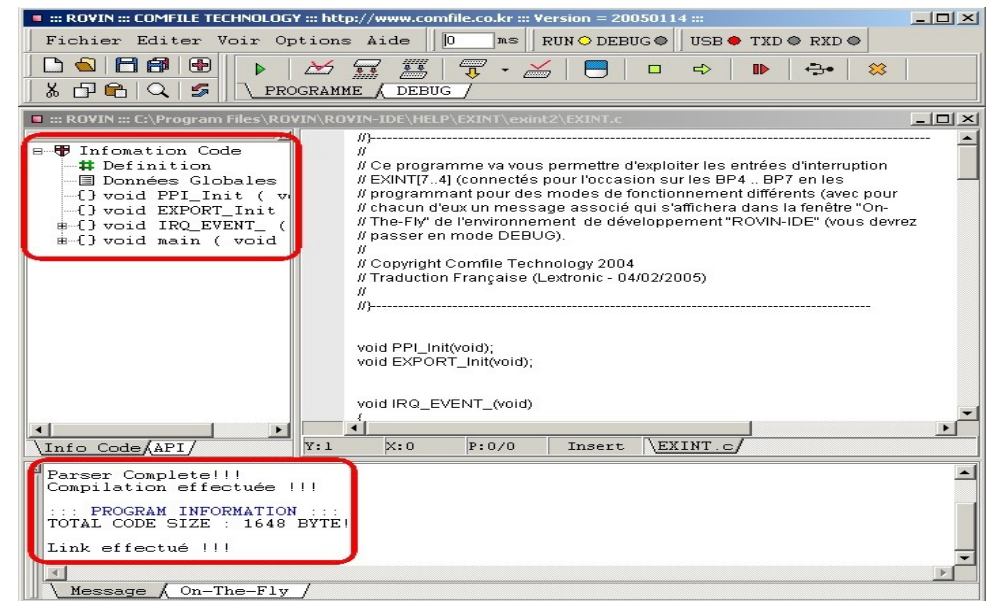


3. Cliquez alors sur le bouton "Linker". Au bout de quelques secondes la fenêtre de message au bas de l'écran doit vous signaler que la compilation s'est correctement déroulée. Le "ROVIN-IDE" vous aura alors créé un fichier exécutable portant le même nom que celui de votre source, mais avec l'extension (.pro)

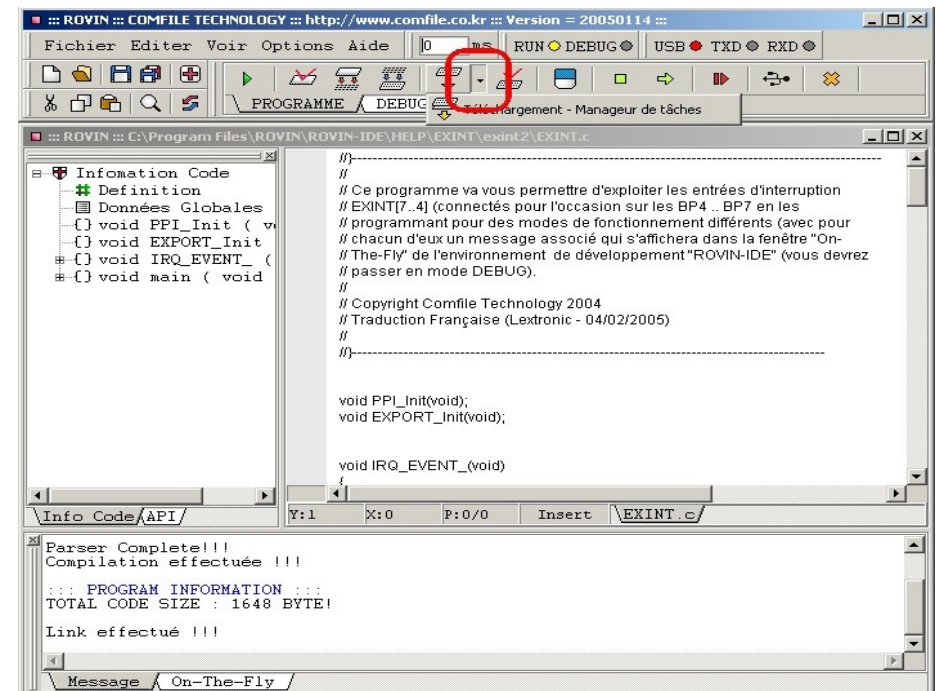




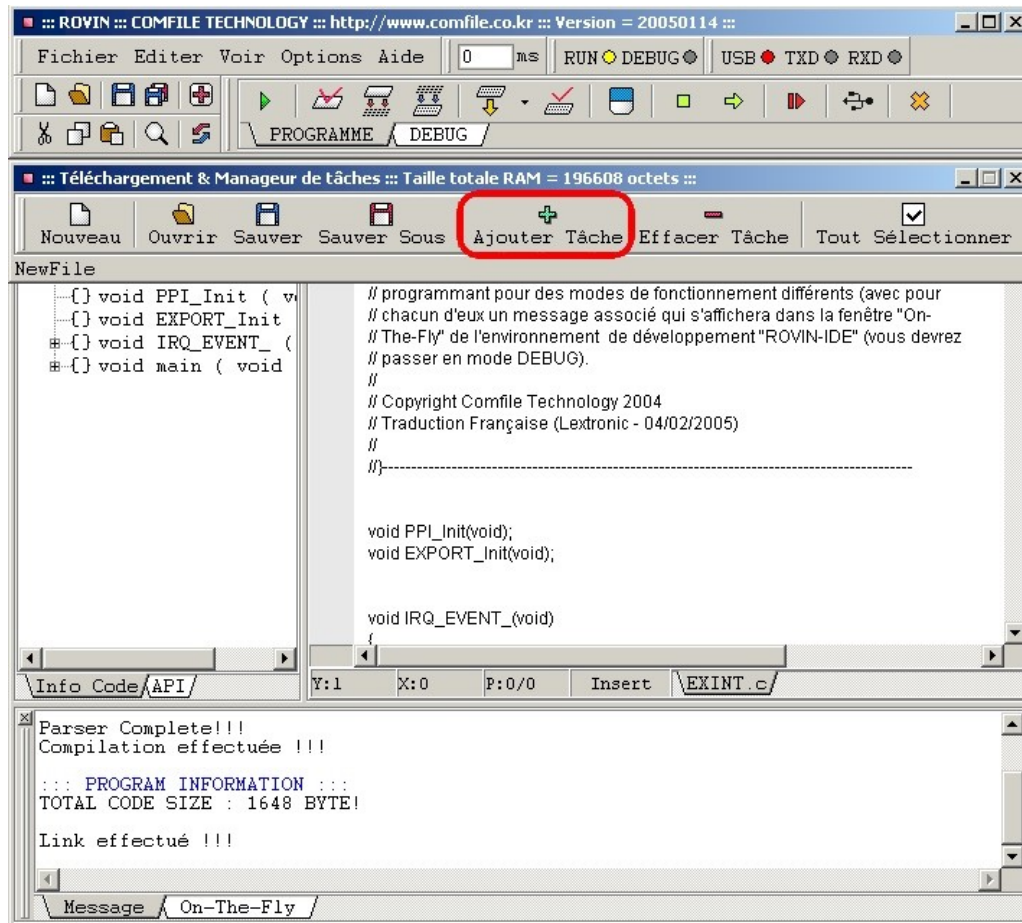
4. Vous pourrez également voir apparaître le nom des variables et fonctions utilisées dans votre source dans la fenêtre "Code Explorateur".



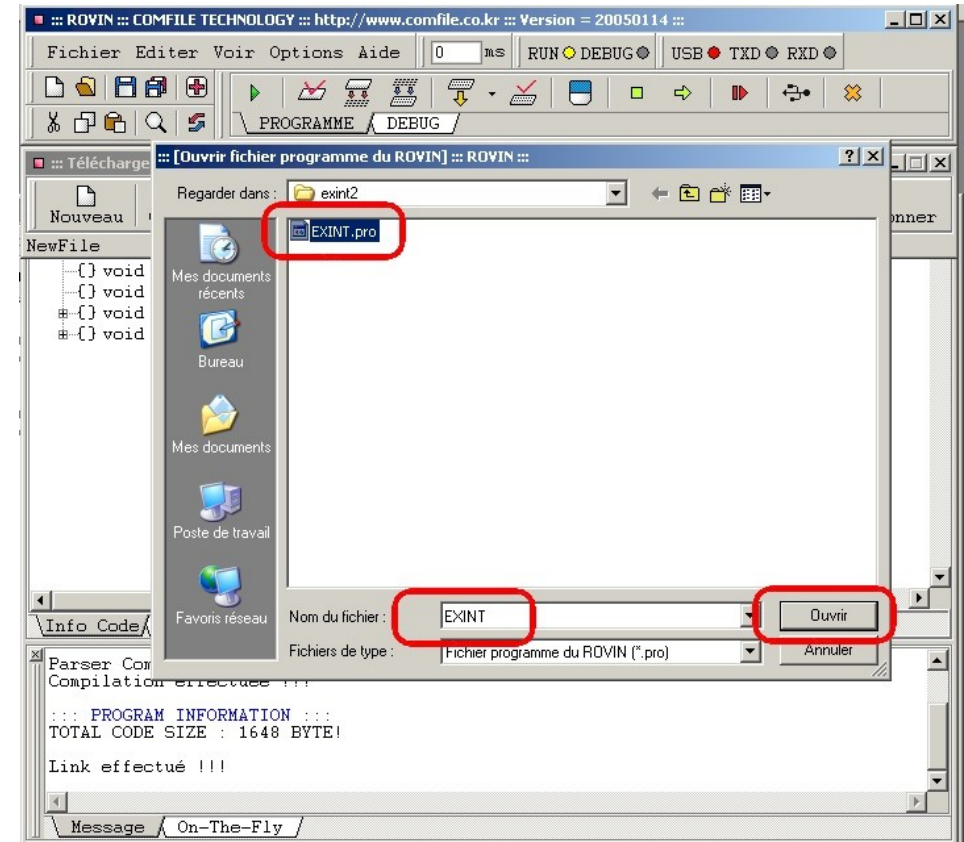
5. Cliquez sur la petite flèche à droite du bouton "Télécharger Programme(s)"



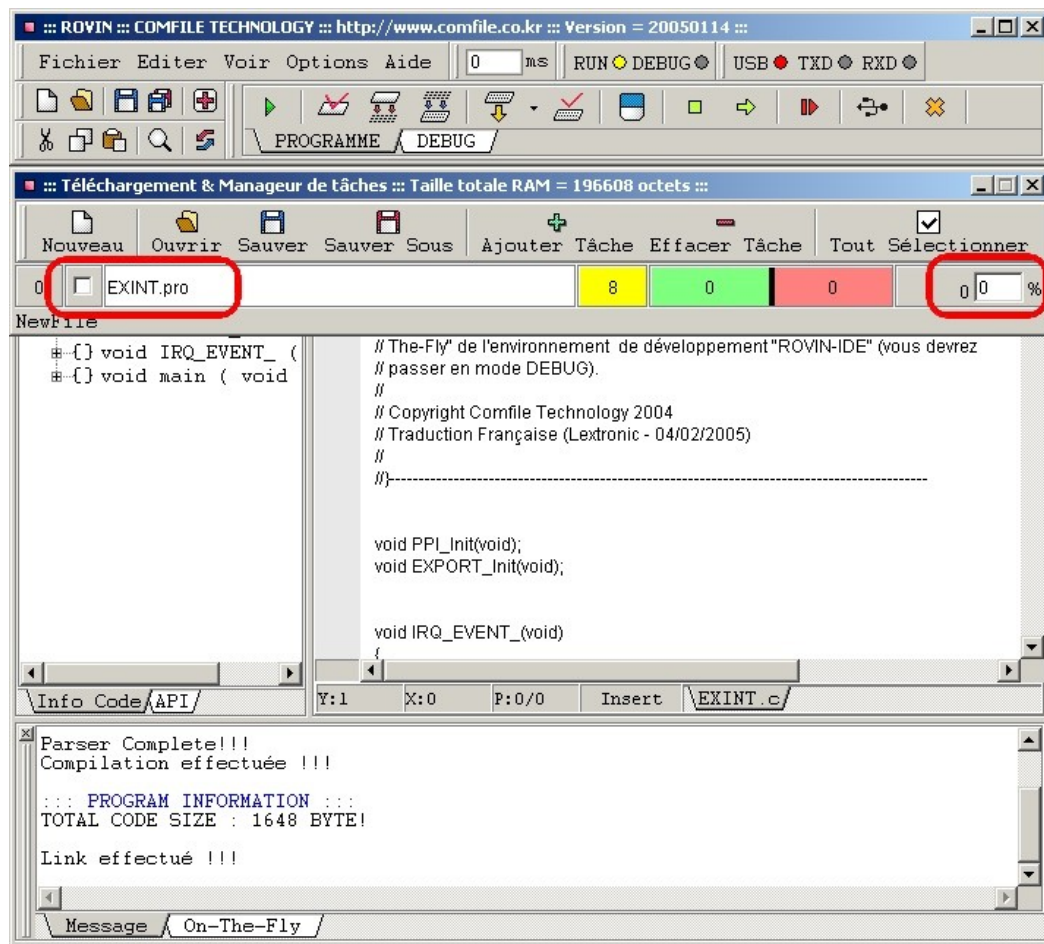
6. La fenêtre "téléchargement - Manager de tâches" s'ouvre alors. Cliquez sur le bouton "Ajouter Tâche" afin d'indiquer à l'environnement de développement "ROVIN-IDE" quel est le ou les programmes à télécharger dans la mémoire du "ROVIN™"



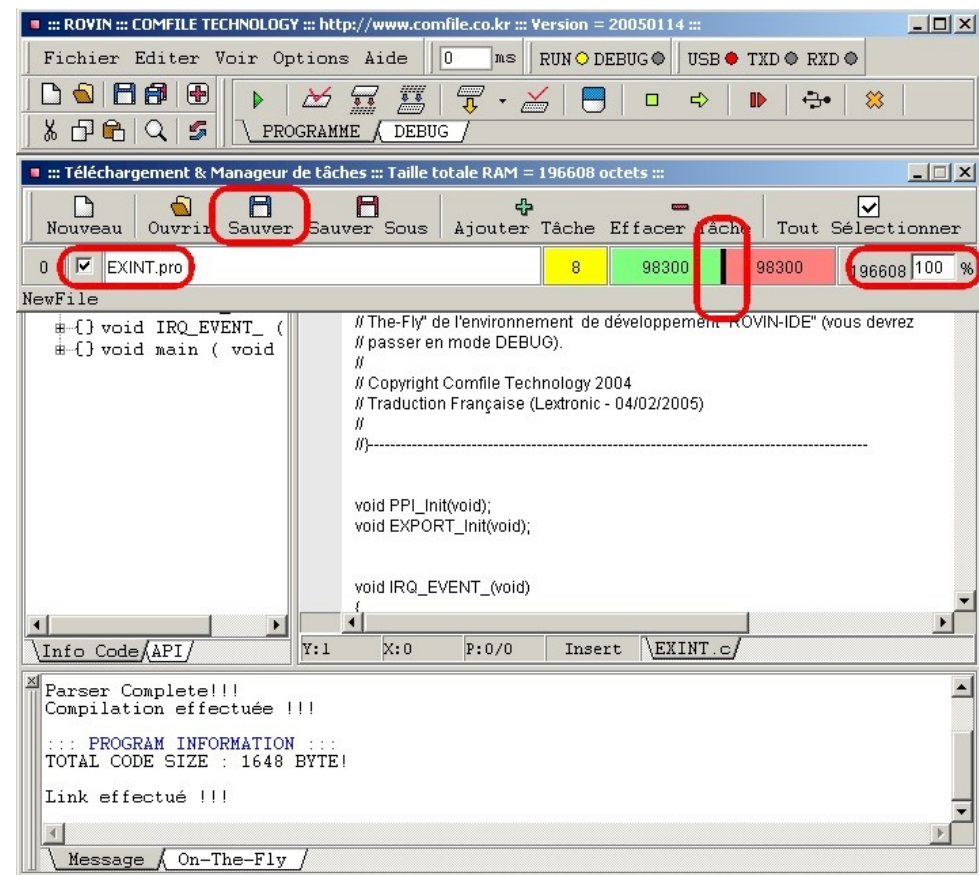
7. Sélectionnez alors le fichier avec l'extension (\*.pro) précédemment généré



. 8. La fenêtre "téléchargement Manager de tâches" doit alors afficher ceci.

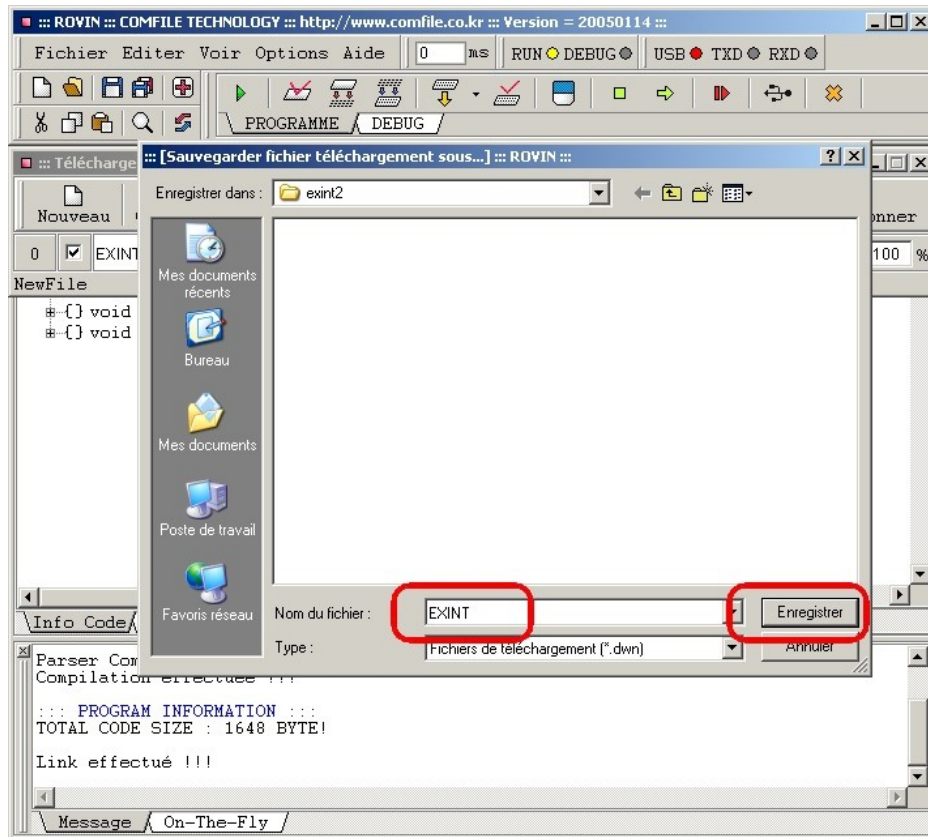




9. Cochez la case de gauche et indiquez à l'environnement de développement "ROVIN-IDE" que le fichier exécutable à transférer peut prendre 100 % de la capacité mémoire du ROVIN™ (pas de problème puisqu'il n'y a qu'une seule tâche). Il n'est pas non plus nécessaire de modifier la répartition mémoire à l'intérieur de la tâche: de ce fait, la barre noire qui sépare les bargraphs vert et rouge ne nécessite aucun réglage. Cliquez enfin sur le bouton "Sauver".

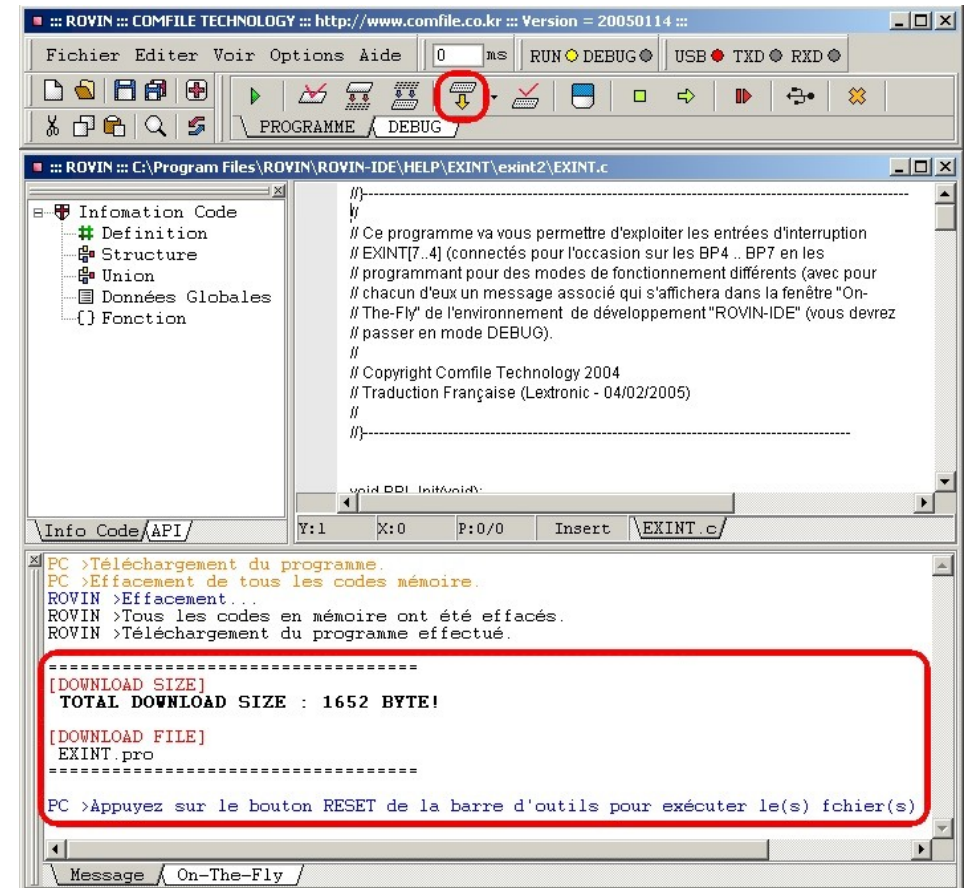



10. Saisissez le nom de votre PROJET. Ce dernier peut avoir le même nom que celui de votre fichier source. Ne mettez aucune extension au nom de votre PROJET. L'environnement de développement se chargera tout seul de lui attribuer une extension (.dwn). De même veuillez impérativement à ne pas changer de répertoire (tous les fichiers d'un Projet doivent être absolument présents dans le même répertoire).

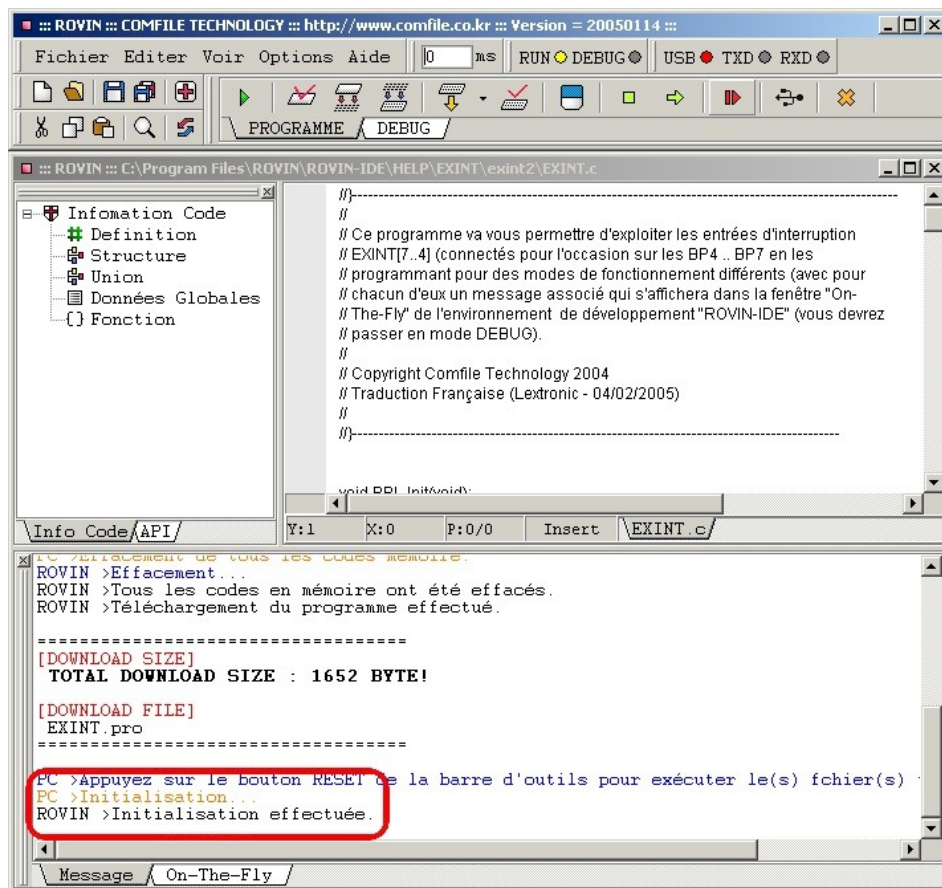




11. Téléchargez alors le programme au sein du ROVIN™ l'aide du bouton  ou .



12. Cliquez sur le bouton Reset , afin que le module ROVIN™ exécute le programme (vous pouvez également solliciter le bouton-poussoir Reset de la platine du ROVIN™ afin d'obtenir le même résultat).

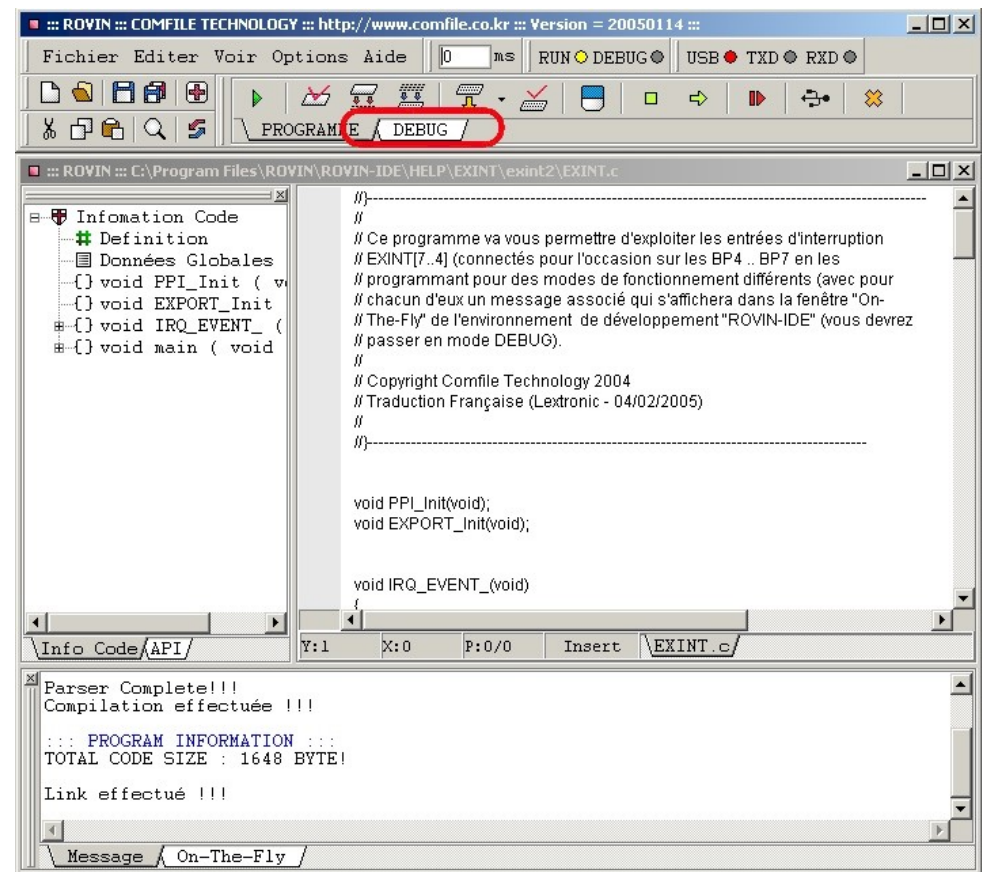


Si le module ROVIN™ ne réagit pas correctement, consultez la rubrique "Problèmes et solutions" donnée à la fin du manuel du ROVIN™. Consultez la documentation du ROVIN™ pour apprendre comment développer des applications multitâches

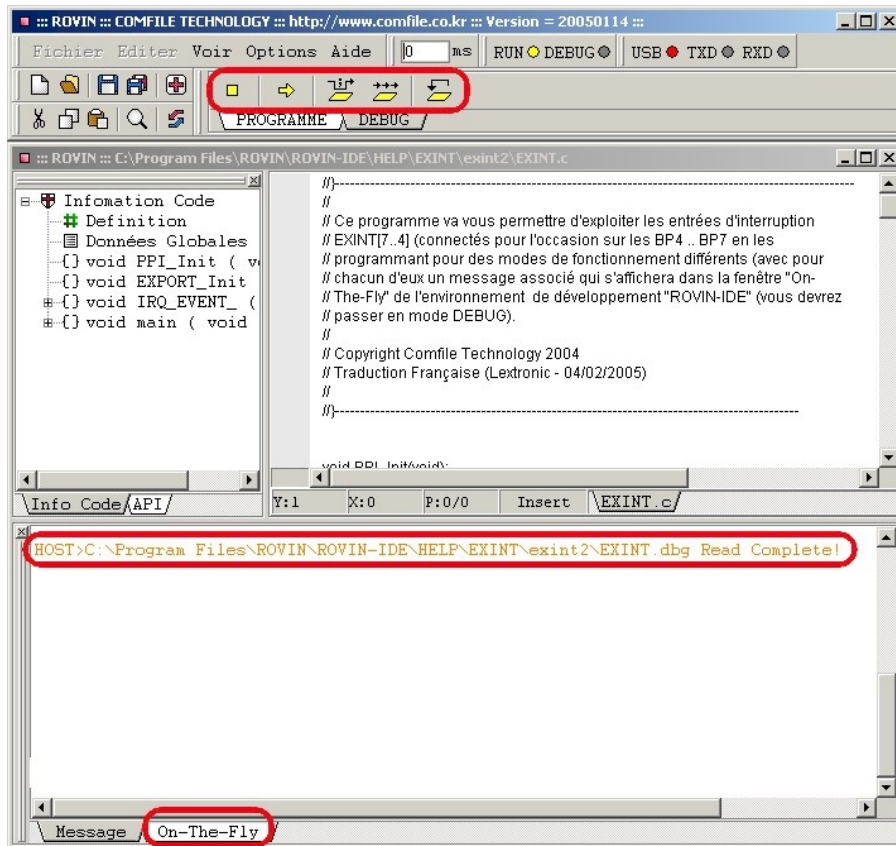
### Utiliser le mode "DEBUG"

Afin de pouvoir activer le mode DEBUG, il vous faut avoir préalablement chargé et compilé un programme. Puis avoir complété la fenêtre "Téléchargement - Manager de tâches" et téléchargé l'application au sein du module ROVIN™.

1. Cliquez alors sur l'onglet "DEBUG" au bas de la boîte à outils du haut de l'écran



2. Si tout se passe normalement l'environnement de développement "ROVIN-IDE" vous signalera dans la fenêtre "On-The-Fly" que vous êtes effectivement passé en mode DEBUG et de nouveaux boutons apparaissent dans la boîte à outils du haut de l'écran.



Si le module ROVIN™ ne réagit pas correctement, consultez la rubrique "Problèmes et solutions" données à la fin du manuel du ROVIN™.

Consultez la documentation du ROVIN™ pour apprendre comment utiliser le mode DEBUG et recevoir des informations de la part du module ROVIN™ dans la fenêtre "On-The-Fly"



**Programme1 : Gestion des Interruptions**

//Ce programme va vous permettre d'exploiter les entrées d'interruption EXINT[7..4] (connectés sur les BP4 .. BP7 en les //programmant pour des modes de fonctionnement différents (avec pour chacun d'eux un message associé qui s'affichera //dans la fenêtre "On- The-Fly"

```
void PPI_Init(void);
void EXPORT_Init(void);
```

```
void IRQ_EVENT_(void)
{
```

```
    -----
    // Test la nature des événements et affichage le nom de celui qui a été détecté
    // dans la fenêtre "On-The-Fly" du PC en mode DEBUG.
```

```
    unsigned char iMSG;
    iMSG=GetMsg();
```

```
    switch(iMSG)
    {
        case MSG_EXINT4 :
            //[Test cas EVENEMENT EXINT4]{-----
            DebugPrint("EVENEMENT 'EXINT4' détecté ! -> Front montant (relâchement de la touche).\n");
            break;
            //}-----
```

```
        case MSG_EXINT5 :
            //[Test cas EVENEMENT EXINT5]{-----
            DebugPrint("EVENEMENT 'EXINT5' détecté ! -> Front descendant (Appui sur la touche).\n");
            break;
            //}-----
```

```
        case MSG_EXINT6 :
            //[Test cas EVENEMENT EXINT6]{-----
            DebugPrint("EVENEMENT 'EXINT6' détecté ! -> Niveau bas (tant qu'on reste appuyé sur la
touche).\n");
```

```
            // Réactive l'autorisation de l'interruption
            // car on fonctionne sur un niveau et non pas sur une transition
            ExintOn(6);
            break;
            //}-----
```

```
        case MSG_EXINT7 :
            //[Test cas EVENEMENT EXINT7]{-----
            DebugPrint("EVENEMENT 'EXINT7' détecté ! -> Réagit à un changement d'état
(appui ou relâchement touche).\n");
            break;
            //}-----
    }
}
```

```
void main (void)
```

```
{
    unsigned short nowVMTIME;
    unsigned short beforeVMTIME;
```

```
    //[Initialisation des PORTS du ROVIN]{-----
    PPI_Init();
    EXPORT_Init();
    //}-----
```

```
    //[Initialisation des entrées d'interruption EXINT]{-----
    //Les broches EXINT[7..4] sont en entrées.
```

```
    PortSetMode(EXPB,0xf0);
```

```
    // Les entrées sont configurées en PULL-UP !
    PortOut(EXPB,0xff);
```

```
    //*****
    // Configure l'entrée d'interruption EXINT.7 pour générer un événement sur un changement d'état
    // (lorsqu'on appuie et qu'on relâche la touche) !
```

```
    ExintSet(7,EXINT_CHANGE);
```

```
    // Autorise la génération des événements par EXINT.7
    ExintOn(7);
```

```
    // Configure l'entrée d'interruption EXINT.6 pour générer un événement tant que l'entrée est au niveau
bas
```

```
    ExintSet(6,EXINT_LOW);
```

```
    // Autorise la génération des événements sur EXINT.6
```

```
    ExintOn(6);
```

```
    // Configure l'entrée d'interruption EXINT.5 pour générer un événement sur la détection d'un front
descendant
    // lorsqu'on appuie sur la touche
    ExintSet(5,EXINT_LEDGE);
```

```
    // Autorise la génération des événements sur EXINT.5
    ExintOn(5);
```

```
    //*****
    // Configure l'entrée d'interruption EXINT.4 pour générer un événement sur la
détection d'un front montant
    // lorsqu'on relâche la touche
    ExintSet(4,EXINT_HEDGE);
```

```
    //Autorise la génération des événements sur EXINT.4
```

```
ExintOn(4);
//}-----
//{-----
//Autorise la génération d'événements
EventOn();

// Boucle principale... Attend les événements !
while(1);
//}-----
}

void PPI_Init(void)
{ // Initialisation des ports PA, PB et PC en entrées.
  // 0 : Broche en sortie.  // 1 : Broche en entrée.

  PPI_SetMode(PA,0xff); PPI_Out  (PA,0xff);
  PPI_SetMode(PB,0xff); PPI_Out  (PB,0xff);
  PPI_SetMode(PC,0xff); PPI_Out  (PC,0xff);

  // Initialisation du port PD en sortie et extinction des Leds de la platine

  PPI_SetMode(PD,0x00);
  PPI_Out  (PD,0xff);
}

void EXPORT_Init(void)
{ // Initialisation des ports EXPA, EXPB et EXPD en entrées (haute impédance).

  PortSetMode(EXPA,0xff);
  PortSetMode(EXPB,0xff);
  PortSetMode(EXPC,0xff);
  PortSetMode(EXPD,0xff);
}
```

Travail à faire : Tester ce programme en mode debug, puis afficher sur les leds jaunes l'activation des boutons poussoirs.

**Programme 2 : conversion analogique numérique**

// Ce programme va vous permettre d'apprendre à utiliser les entrées du comparateur analogique. Les 2 entrées devront être reliée au curseur de résistance ajustable (ou potentiomètre) comme indiqué sur le schéma. A chaque fois que la valeur de la tension des 2 entrées sera identique. Une interruption sera générée (EVENEMENT) et un message associé s'affichera dans la fenêtre "On-The-Fly" du mode DEBUG. Pour les tests, placez la résistance ajustable de la plaque de connexion sans soudure à mi-course, puis tournez le potentiomètre, lorsque les 2 tensions seront identiques, le message s'affichera dans la fenêtre. Si vous arrivez à trouver l'équilibre parfait, le message s'affichera en permanence (puisque l'interruption est réactivée à chaque fois dans le programme).

```
void PPI_Init(void);
void EXPORT_Init(void);
```

```
void Delay(int pTIME)
{
    while(pTIME--);
}
```

```
void IRQ_EVENT_(void)
{
    // [Récupère valeur de l'événement
    -
    switch(GetMsg())
    {
        case MSG_ANCOMP : // Est-ce un EVENEMENT causé par le comparateur analogique ?

            DebugPrint("Les 2 tensions sont identiques !\n");    // Oui -> Affiche message dans fenêtre
            "On-The-Fly"
            AncompOn();           // Autorise à nouveau génération EVENEMENTS par le
            comparateur
            break;
        }
    }
}
```

```
void main(void)
{
    // Initialisation des ports du ROVIN
    PPI_Init();
    EXPORT_Init();

    // Configure le comparateur analogique et autorise la génération d'EVENEMENT par ce dernier !
    AncompSet(ANCOMP_TOGGLE);
    AncompOn();
}
```

```
// Active l'horloge RTC car le comparateur dépend de cette dernière.
RtcOn();
```

```
// Autorise la génération des EVENEMENTS !
EventOn();
```

```
// Attend un EVENEMENT !
while(1);
}
```

```
void PPI_Init(void)
{ // Initialisation des ports PA, PB et PC en entrées.
    // 0 : Broche en sortie.
    // 1 : Broche en entrée.
```

```
PPI_SetMode(PA,0xff);  PPI_Out  (PA,0xff);
PPI_SetMode(PB,0xff);  PPI_Out  (PB,0xff);
PPI_SetMode(PC,0xff);  PPI_Out  (PC,0xff);
```

```
// Initialisation du port PD en sortie et extinction des Leds de la platine
PPI_SetMode(PD,0x00);
PPI_Out  (PD,0xff);
}
```

```
void EXPORT_Init(void)
{ // Initialisation des ports EXPA, EXPB et EXPD en entrées (haute impédance)
    PortSetMode(EXPA,0xff);
    PortSetMode(EXPB,0xff);
    PortSetMode(EXPC,0xff);
    PortSetMode(EXPD,0xff);
}
```

**Travail à faire :**

- Câbler correctement sur les entrées du microcontrôleur le potentiomètre figurant sur la carte.
- Installer un deuxième potentiomètre sur les entrées du microcontrôleur et modifier le code ci-dessus puis Afficher la valeur des deux potentiomètres.

**Programme 3 : Gestion des UARTS (RS232)**



// Ce programme va vous permettre d'utiliser le port série UART1 du ROVIN. Ce dernier est configuré pour générer un EVENEMENT à chaque fois qu'il reçoit 1 octet. Dès lors, le programme récupérera la valeur de cet octet et le renverra aussitôt sur le port série. Si vous utilisez un terminal de saisie connecté à la platine du ROVIN (comme hyper terminal (TM) par exemple), à chaque fois que vous taperez des caractères dans la fenêtre de saisie du terminal, ces mêmes caractères vous seront retournés par le ROVIN et ils s'afficheront sur l'écran du PC.

// le terminal de saisie devra être configuré en 9600 bds !

```
void PPI_Init(void)
```

```
{ // Initialisation des ports PA, PB et PC en entrées.
```

```
  // 0 : Broche en sortie.
```

```
  // 1 : Broche en entrée.
```

```
  PPI_SetMode(PA,0xff);
```

```
  PPI_Out  (PA,0xff);
```

```
  PPI_SetMode(PB,0xff);
```

```
  PPI_Out  (PB,0xff);
```

```
  PPI_SetMode(PC,0xff);
```

```
  PPI_Out  (PC,0xff);
```

```
  // Initialisation du port PD en sortie et extinction des Leds de la platine
```

```
  PPI_SetMode(PD,0x00);
```

```
  PPI_Out  (PD,0xff);
```

```
}
```

```
void EXPORT_Init(void)
```

```
{
// Initialisation des ports EXPA, EXPB et EXPD en entrées (haute impédance)
```

```
  PortSetMode(EXPA,0xff);
```

```
  PortSetMode(EXPB,0xff);
```

```
  PortSetMode(EXPC,0xff);
```

```
  PortSetMode(EXPD,0xff);
```

```
}
```

```
void Delay(int pTIME)
```

```
{
```

```
  while(pTIME--);
```

```
}
```

```
#define uartPORTnum    1
```

```
#define rxPACKETsize    1
```

```
void main(void)
```

```
{
```

```
//Initialise les ports
```

```
PPI_Init();
```

```
EXPORT_Init();
```

```
//Initialise l'UART1 - UART1 9600 BPS
```

```
UartSetBaud(uartPORTnum,9600);
```

```
// Active l'UART1 !
```

```
UartOn(uartPORTnum);
```

```
// Configure l'UART
```

```
UartSetPacketSize(uartPORTnum,rxPACKETsize);
```

```
// Active les EVENEMENTs de l'UART
```

```
UartEventOn(uartPORTnum);
```

```
//}-----
```

```
RtcOn();
```

```
// Autorise les EVENEMENTS
```

```
EventOn();
```

```
// Attend des EVENEMENTS
```

```
while(1);
```

```
}
```

```
void IRQ_EVENT_(void)
```

```
{
```

```
//[Gestion des EVENEMENTS
```

```
  unsigned char iSTR[255];          // Taille du buffer
```

```
  GetMsg();
```

```
  // Récupère le contenu du buffer de l'UART.
```

```
  UartRxBufRead(uartPORTnum,iSTR);
```

```
  // Renvoi l'octet reçu sur l'UART
```

```
  UartWrite(uartPORTnum,iSTR[0]);
```

```
}
```

## 1) Configurer Hyper Terminal

- Lancer Hyper Terminal
- Donnez un nom à la communication (par exemple ROVIN9600), puis validez.

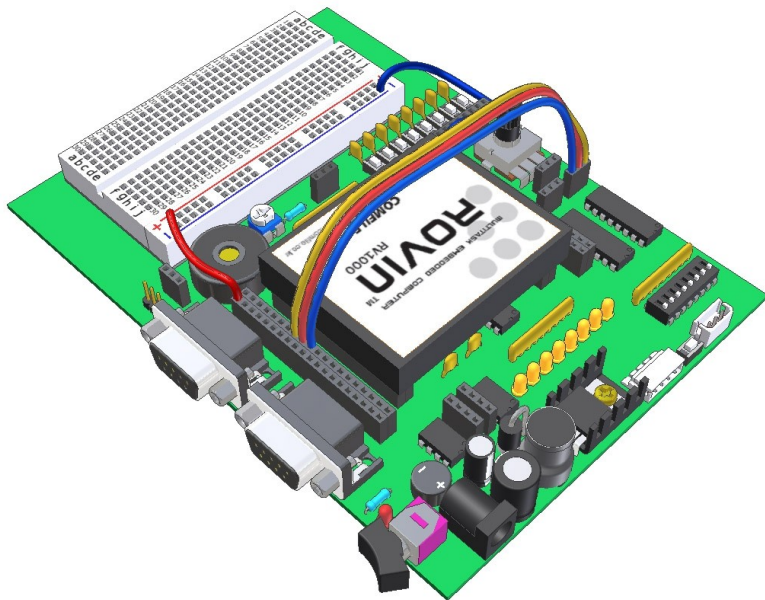
### Travail à faire :

- Tester ce programme.
- Modifier ce programme pour qu'il génère un événement à chaque fois qu'il reçoit 255 octets. Dès lors, le programme récupérera le contenu du buffer et le renverra aussitôt sur le port série.

Sélectionnez la vitesse de communication (9600 ou 115200 bds (ou une autre valeur) selon la vitesse configurée sur le ROVIN™. Recopiez également les autres paramètres comme sur la fenêtre ci-dessus.

Dès lors la communication est opérationnelle et vous pouvez commencer les échanges de données entre le PC et la platine du ROVIN™.



**Travail à faire :**

- Tester le programme, puis brancher toutes les entrées du composant sur le microcontrôleur.
- Brancher un potentiomètre sur le 74HC165 puis lisez sa valeur à partir du microcontrôleur.

**Programme 5 : Port PPI en E/S**

// Ce programme va vous permettre de réaliser un petit chenillard avec les Leds du port PD (configuré en sortie).

```
void PPI_Init(void)
{ // Initialisation des ports PA, PB et PC en entrées.
  // 0 : Broche en sortie.
  // 1 : Broche en entrée.
```

```
  PPI_SetMode(PA,0xff);
  PPI_Out (PA,0xff);
  PPI_SetMode(PB,0xff);
  PPI_Out (PB,0xff);
  PPI_SetMode(PC,0xff);
  PPI_Out (PC,0xff);
```

```
// Initialisation du port PD en sortie et extinction des Leds de la platine
```

```
  PPI_SetMode(PD,0x00);
  PPI_Out (PD,0xff);
}
```

```
void EXPORT_Init(void)
{ //Initialisation des ports EXPA, EXPB et EXPD en entrées (haute impédance).
```

```
  PortSetMode(EXPA,0xff);
  PortSetMode(EXPB,0xff);
  PortSetMode(EXPC,0xff);
  PortSetMode(EXPD,0xff);
}
```

```
void Delay(int pTIME)
{
  while(pTIME--);
}
```

```
void main(void)
{
  unsigned int iINTERVAL;
  iINTERVAL=0xf0;
  // Vous pouvez changer ici la valeur de la tempo iINTERVAL !
```

```
  PPI_Init();
  EXPORT_Init();
```

```
// Création ligne par ligne de l'effet "chenillard"
```

```
// Afin de vous perfectionnez utilisez une entrée de conversion "A/N" pour
// modifier la vitesse de défilement à l'aide du potentiomètre de la carte.
```

```
while(1)
{
  PPI_Out(PD,~0b00000000); Delay(iINTERVAL);
  PPI_Out(PD,~0b00000001); Delay(iINTERVAL);
  PPI_Out(PD,~0b00000010); Delay(iINTERVAL);
  PPI_Out(PD,~0b00000100); Delay(iINTERVAL);
  PPI_Out(PD,~0b00001000); Delay(iINTERVAL);
  PPI_Out(PD,~0b00010000); Delay(iINTERVAL);
  PPI_Out(PD,~0b00100000); Delay(iINTERVAL);
  PPI_Out(PD,~0b01000000); Delay(iINTERVAL);
  PPI_Out(PD,~0b10000000); Delay(iINTERVAL);
  PPI_Out(PD,~0b00000000); Delay(iINTERVAL);
  PPI_Out(PD,~0b10000000); Delay(iINTERVAL);
  PPI_Out(PD,~0b01000000); Delay(iINTERVAL);
  PPI_Out(PD,~0b10000000); Delay(iINTERVAL);
```

```

PPI_Out(PD,~0b00100000); Delay(iINTERVAL);
PPI_Out(PD,~0b00010000); Delay(iINTERVAL);
PPI_Out(PD,~0b00001000); Delay(iINTERVAL);
PPI_Out(PD,~0b00000100); Delay(iINTERVAL);
PPI_Out(PD,~0b00000010); Delay(iINTERVAL);
PPI_Out(PD,~0b00000001); Delay(iINTERVAL);
} }

```

## PP-IN

// Ce programme va simplement récupérer l'état du port PA (configuré pour l'occasion en entrée) et en faire une recopie sur le port PD (configuré pour l'occasion en sortie).

```

void PPI_Init(void)
{ // Initialisation des ports PA, PB et PC en entrées.
  // 0 : Broche en sortie.
  // 1 : Broche en entrée.

  PPI_SetMode(PA,0xff);
  PPI_Out  (PA,0xff);
  PPI_SetMode(PB,0xff);
  PPI_Out  (PB,0xff);
  PPI_SetMode(PC,0xff);
  PPI_Out  (PC,0xff);

  // Initialisation du port PD en sortie et extinction des Leds de la platine

  PPI_SetMode(PD,0x00);
  PPI_Out  (PD,0xff);
}

void EXPORT_Init(void)
{ // Initialisation des ports EXPA, EXPB et EXPD en entrées (haute impédance).

  PortSetMode(EXPA,0xff);
  PortSetMode(EXPB,0xff);
  PortSetMode(EXPC,0xff);
  PortSetMode(EXPD,0xff);
}

void Delay(int pTIME)
{
  while(pTIME--);
}

void main(void)

```

```

{
  // Initialisation des ports du ROVIN

  PPI_Init();
  EXPORT_Init();

  while(1)
  { // Boucle principale (recopie de l'état du port PA sur PD)

    PPI_Out(PD,PPI_In(PA));
  } }

```

// Ce programme configure les ports en entrée ou en sortie et fait clignoter toutes les Leds du port PD (configuré pour //occasion en sortie) en plaçant indéfiniment ce dernier à 0xff, puis à 0x00 et ainsi de suite avec une temporisation entre //chaque changement d'état.

```

void PPI_Init(void)
{ // Initialisation des ports PA, PB et PC en entrées.
  // 0 : Broche en sortie.
  // 1 : Broche en entrée.

  PPI_SetMode(PA,0xff);
  PPI_Out  (PA,0xff);
  PPI_SetMode(PB,0xff);
  PPI_Out  (PB,0xff);
  PPI_SetMode(PC,0xff);
  PPI_Out  (PC,0xff);

  // Initialisation du port PD en sortie et extinction des Leds de la platine

  PPI_SetMode(PD,0x00);
  PPI_Out  (PD,0xff);
}

void EXPORT_Init(void)
{ //Initialisation des ports EXPA, EXPB et EXPD en entrées (haute impédance).

  PortSetMode(EXPA,0xff);
  PortSetMode(EXPB,0xff);
  PortSetMode(EXPC,0xff);
  PortSetMode(EXPD,0xff);
}

void Delay(int pTIME)

```



```

{
  while(pTIME--);
}

void main(void)
{ //Initialisation des ports du ROVIN

  PPI_Init();
  EXPORT_Init();

  while(1)
  {
    PPI_Out(PD,0x00); // Allume toutes les Leds
    Delay(0x5ff);     // Tempo
    PPI_Out(PD,0xff); // Eteind toutes les Leds
    Delay(0x5ff);     // Tempo
  } }

```

**Travail à faire :**

- Tester ces 3 programmes puis modifier un des trois programmes pour afficher sur un afficher 7 segment les nombres en hexadécimale de 1 a F.

**Programme : Pilotage d'une EEPROM I2C**

// Ce programme va écrire et lire dans une mémoire EEPROM série à l'aide d'une communication I2C. Pour ce faire, un //port I2C sera déclaré en affectant la broche PA.5 à SDA et PA.4 à SCL. Le programme va simplement écrire les octets //00 à FF depuis l'adresse 00 à FF. Ce dernier va ensuite relire les adresses 00 à FF et afficher le contenu de chaque octet  
 // Lu sur les Leds du port PD.

```

void PPI_Init(void);
void EXPORT_Init(void);
void Delay(int pTIME);

```

```
int i;
```

```

void main(void)
{

```

```

//[Initialisation des ports]
PPI_Init();
EXPORT_Init();

//Création du port I2C]
I2cCreate(0, PA, 5, 4, I2C_MSBFIRST, 0);

//Test d'écriture de la mémoire EEprom 24LC32

for(i=0; i<=0xff; i++)
{
  I2cStart(0); // Condition START I2C !

  I2cBitOut(0, 8, 0b10100000); // Envoi 8 bits formant l'octet de contrôle de la
24LC32 !
  if(!I2cAck(0,0xff)) break; // Test ACK de la part de la 24LC32 (TIME OUT
= 32 ms)

  I2cBitOut(0, 8, i>>8); // Envoi adresse HAUTE !
  if(!I2cAck(0,0xff)) break;

  I2cBitOut(0, 8, i); // Envoi adresse BASSE !
  if(!I2cAck(0,0xff)) break;

  I2cBitOut(0, 8, i); // Envoi la DONNEE !
  if(!I2cAck(0,0xff)) break;

  I2cStop(0); // Condition STOP I2C!

  //Tempo - L'écriture d'un octet en mémoire EEprom dure environ 5 ms. On attend 6 ms pour
avoir une
  // marge de sécurité. Delay(0x1a);
}

//[Test lecture de la mémoire EEprom 24LC32]

for(i=0; i<=0xff; i++)
{
  I2cStart(0); // Condition START I2C !

  I2cBitOut(0, 8, 0b10100000); // Envoi_ bits formant l'octet de contrôle de la
24LC32 !
  if(!I2cAck(0,0xff)) break; // Test ACK de la part de la 24LC32 (TIME OUT
= 32 ms)

  I2cBitOut(0, 8, i>>8); // Envoi adresse HAUTE!

```

```

if(!I2cAck(0,0xff)) break;

I2cBitOut(0, 8, i);           // Envoi adresse BASSE !
if(!I2cAck(0,0xff)) break;

// Cycle de lecture de la mémoire EEprom

I2cStart(0);                  // Condition START I2C !

I2cBitOut(0, 8, 0b10100001);  // Envoi octet de contrôle (de lecture) de la 24LC32 !
if(!I2cAck(0,0xff)) break;    // Test ACK de la part de la 24LC32 (TIME OUT =
32 ms)

PPI_Out(PD, ~I2cBitIn(0, 8));  // Lecture de la donnée (8 bits, soit 1 octet) et sortie
sur le port PD!

I2cAck(0,0);                  // Génération cycle d'horloge (pour ACK vers la
24LC32)
I2cStop(0);                   // Condition STOP I2C !

Delay(0x5f);                  //Temporisation pour visibilité de l'affichage des Leds
du port PD
}
}

void PPI_Init(void)
{ // Initialisation des ports PA, PB et PC en entrées.
  // 0 : Broche en sortie.
  // 1 : Broche en entrée.

PPI_SetMode(PA,0xff);
PPI_Out (PA,0xff);
PPI_SetMode(PB,0xff);
PPI_Out (PB,0xff);
PPI_SetMode(PC,0xff);
PPI_Out (PC,0xff);

// Initialisation du port PD en sortie et extinction des Leds de la platine

PPI_SetMode(PD,0x00);
PPI_Out (PD,0xff);
}

void EXPORT_Init(void)
{ // Initialisation des ports EXPA, EXPB et EXPD en entrées (haute impédance)

```

```

PortSetMode(EXPA,0xff);
PortSetMode(EXPB,0xff);
PortSetMode(EXPC,0xff);
PortSetMode(EXPD,0xff);

```

```

PortOut(EXPA,0x00);
PortOut(EXPB,0x00);
PortOut(EXPC,0x00);
PortOut(EXPD,0x00);
}

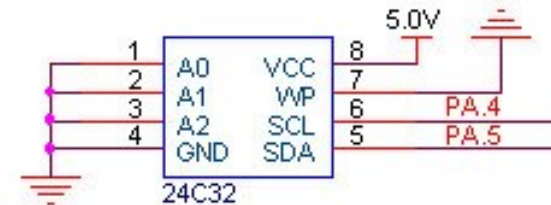
```

```

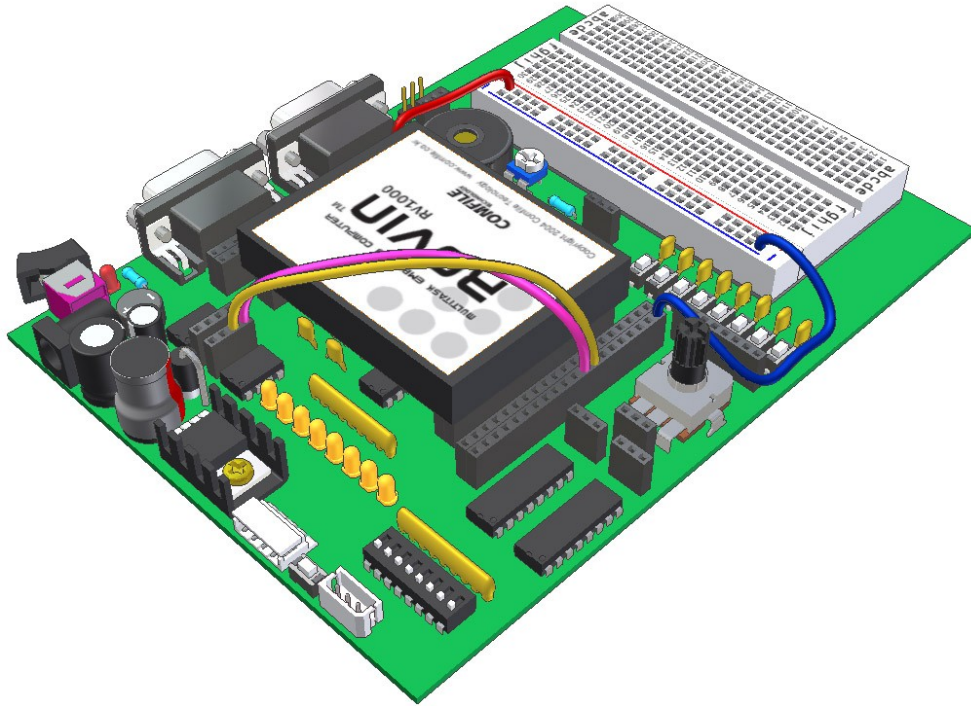
void Delay(int pTIME)
{
  while(pTIME--);
}

```

### Schéma du 24C32



### Câblage sur la Platine



### Travail à faire

### Programme : Pilotage d'un sonar

// Ce programme va vous permettre d'exploiter un module sonar MSU04. La broche PD.0 sera reliée à l'entrée Trigger //Pulse du module afin que le ROVIN lui génère des 'pulses'. En retour, le module MSU04 retournera des impulsions dont //la durée sera proportionnelle à la distance séparant le MSU04 de l'obstacle. La durée des impulsions sera mesurée grâce // à l'entrée Capture 0 qui sera configurée pour détecter une impulsion haute. Puis la valeur sera affichée dans la fenêtre //"On-The-Fly" du PC (vous devrez donc passer en mode DEBUG pour obtenir un affichage dans l'environnement //"ROVIN-IDE").

```
// Initialisation des ports du ROVIN
void PPI_Init(void);
void EXPORT_Init(void);
```

```
// Déclaration des variables
```

```
unsigned short CP_LOW_TIME;
unsigned short CP_HIGH_TIME;
```

```
#define _capTIME_ ((double)1/18432000)*1024
```

```
// Fonction de génération des tempos
```

```
void Delay(int pTIME)
{
    while(pTIME--);
}
```

```
void IRQ_EVENT_(void)
{
```

```
// Gestion des EVENEMENTS Test la nature des événements et affichage de la valeur des
mesures dans la
// fenêtre "On-The-Fly" du PC en mode DEBUG.
```

```
switch(GetMsg())
{
```

```
    case MSG_CAPTURE0 : // Récupère la nature de l'événement et regarde si il s'agit de
capture 0 ?
```

```
        // Oui -> Lecture de la valeur mesurée par Capture 0
        // et sauvegarde cette dernière dans CP_HIGH_TIME.
```

```
        CaptureRead(0, &CP_LOW_TIME, &CP_HIGH_TIME);
```

```
        // Affiche la durée dans la fenêtre "On-The-Fly" (Efface la fenêtre au préalable).
```

```
        DebugClear();
```

```
        DebugPrint("[ Durée du pulse du MSU04 ] = ");
```

```

    DebugDOUBLE(CP_HIGH_TIME, DEC);
    Delay (0xff);           // Tempo nécessaire pour permettre une lecture sur
    l'écran du PC

    // Réactive la capture.
    CaptureOn(0);
}
}

// Fonction principale

void main(void)
{ // Initialisation des ports du ROVIN
  PPI_Init();
  EXPORT_Init();
  PPI_Out(PD,0xfe);        // Eteint les Leds du port et sortie PD.0 à 0

  // [Initialisation de la capture]
  // Utilise l'entrée de capture 0
  // Configure cette entrée en mode "HIGHCYCLE" afin de capturer
  // la durée d'une impulsion haute, puis basse / Introduit un facteur de division de 1024

  CaptureSet(0, CAPTURE_HIGH|CAPTURE_DIV1024);
  CaptureOn(0); // L'entrée Capture 0 sera configurée avec une résistance de PULL-UP !

  // Autorise la génération d'EVENEMENTS
  EventOn();

  // Bouclesans fin... Génère impulsion et ... Attend les événements !
  for(;;)
  {
    PPI_BitOut(PD,0,1);      // Niveau Haut.
    Delay (0x10);
    PPI_BitOut(PD,0,0);      // Niveau Bas.
  }
}

// Intialisation des ports

void PPI_Init(void)
{ // Initialisation des ports PA, PB et PC en entrées.
  // 0 : Broche en sortie.
  // 1 : Broche en entrée.

  PPI_SetMode(PA,0xff); PPI_Out (PA,0xff);

```

```

PPI_SetMode(PB,0xff); PPI_Out (PB,0xff);
PPI_SetMode(PC,0xff); PPI_Out (PC,0xff);

// Initialisation du port PD en sortie et extinction des Leds de la platine

PPI_SetMode(PD,0x00);
PPI_Out (PD,0xff);
}

// Intialisation des ports

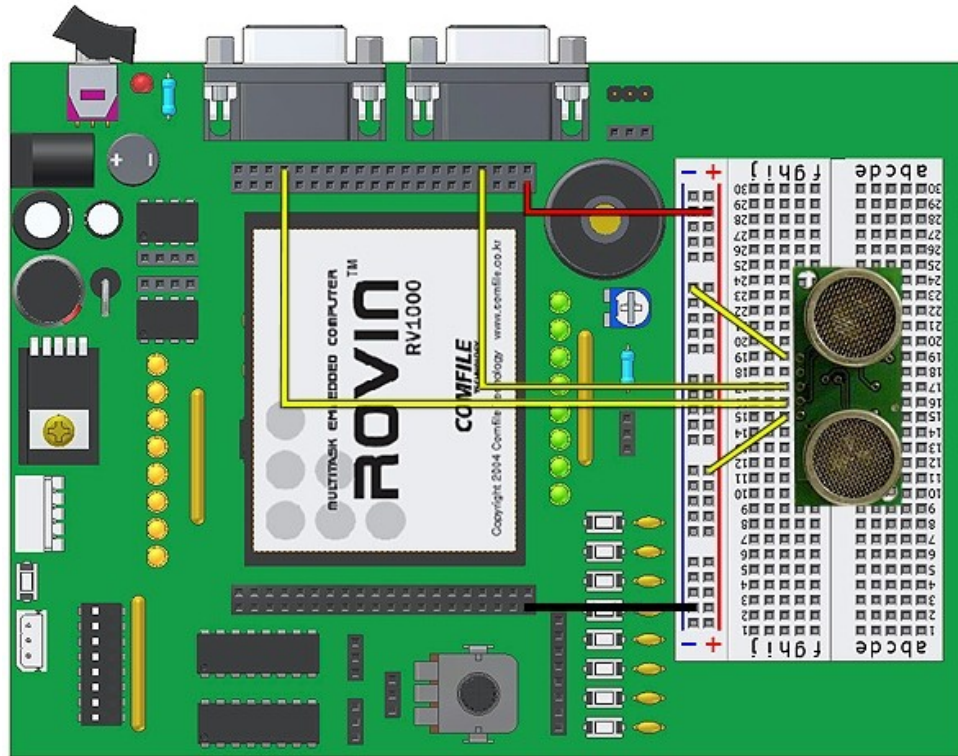
void EXPORT_Init(void)

{ // Initialisation des ports EXPA, EXPB et EXPD en entrées (haute impédance).

  PortSetMode(EXPA,0xff);
  PortSetMode(EXPB,0xff);
  PortSetMode(EXPC,0xff);
  PortSetMode(EXPD,0xff);
}

```

### Câblage de la platine et du sonar

**Travail a faire :**

- Tester le programme.
- Installer l'ensemble sur un robot puis piloter le robot dans un labyrinthe (modifier le code actuel).