

# Le micro-contrôleur Siemens C167

## 1 La carte micro-contrôleur et son interface

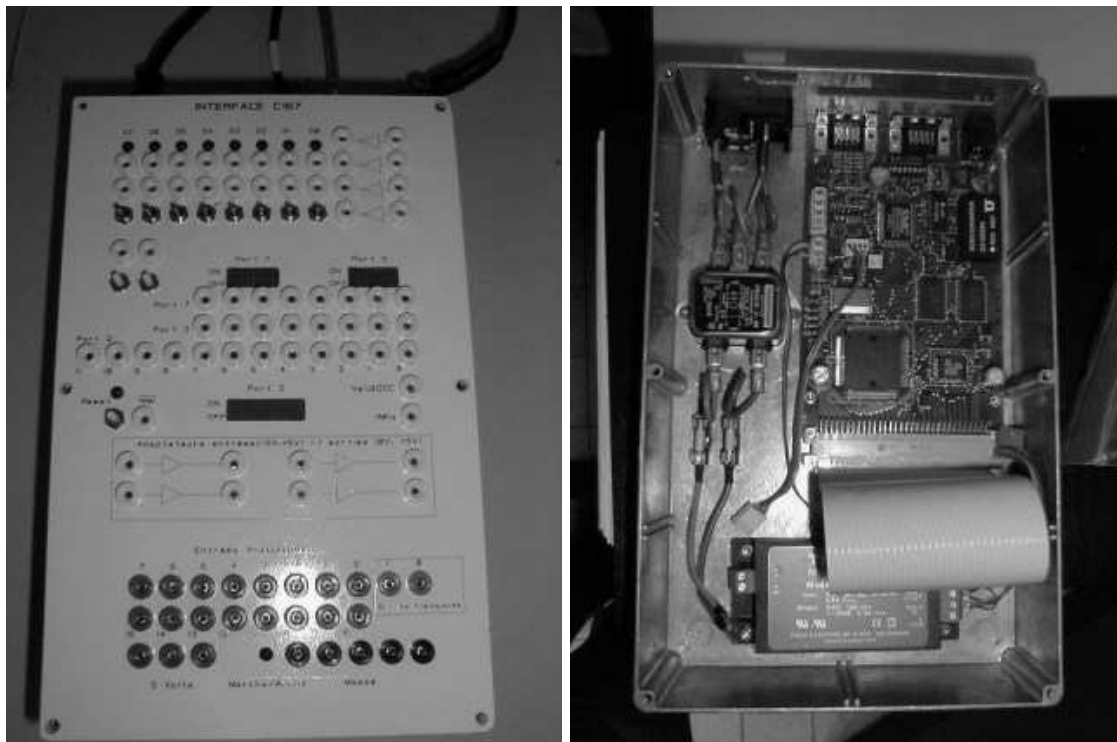
### 1.1 Carte microsys167-Eth

Cette carte a été conçue par la société Hightec (Allemagne) autour d'un micro-contrôleur Siemens C167 CR cadencé à 20MHz. Sur la carte sont disponibles 1Mo de RAM et 512 Ko de Flash-EPROM et les interfaces de communication suivantes :

- une interface série asynchrone et synchrone (RS232),
- une interface série synchrone (RS485),
- une interface bus CAN (Controller Area Network),
- une liaison ethernet qui sera utilisée pour le téléchargement et la mise au point (débugueur),
- un connecteur externe de 96 points. Un certain nombre de ces points sont ramenés sur la face avant d'un boîtier d'interface (Ports P7, P3, P2 et P5, reset,  $\overline{NMI}$ ).

La face avant est représentée en fin de présentation de la partie matériel. Les connecteurs correspondant aux interfaces série, CAN, ethernet se retrouvent en face arrière du boîtier d'interface.

### 1.2 L'interface carte microsys167-Eth ↔ extérieur



Sur la face avant sont disponibles :

- huit interrupteurs, deux boutons poussoirs,
- huit LEDs,
- quatre inverseurs TTL,
- un bouton **reset**, une entrée  $\overline{NMI}$
- une horloge de 1 MHz et son entrée de validation (**ValidOSC**)
- quatre adaptateurs analogiques permettant de ramener une tension  $\in [-5V, +5V]$  en une tension  $\in [0,5V]$ ,
- les ports d'entrée-sortie logiques Port P7 (huit bits), Port P3 (huit bits), Port P2 (douze bits). Ces ports sont programmables en entrée/sortie mais il faut aussi sur l'interface positionner les lignes en entrée/sortie par des interrupteurs deux positions ON/OFF (de type "DIL"). Cette contrainte est liée à l'interface, les interrupteurs pilotent des buffers de bus placés sur les lignes des ports. Les interrupteurs ON(entrée)/OFF(sortie) sont positionnés pour les applications qui seront traitées en TP, il ne sera pas nécessaire de les manipuler.
- seize entrées analogiques correspondant au Port 5,
- deux sorties analogiques 0 et 1, sélectionnées par Chip Select et correspondant respectivement aux adresses 0x2xxxxx et 0x3xxxxx (voir polycopié de cours, chapitre 3) .

Les fonctions alternatives des ports, disponibles sur la face avant, sont rappelées ici.

### Port P7

Les huit bits du port sont programmables en entrées/sorties logiques, les fonctions alternatives sont les sorties de l'unité PWM et les entrées capture/sorties comparaison de l'unité CAPCOM2.

Fonction	Sens	Fonction	Sens
P7.0	$\leftrightarrow$	POUT0	$\rightarrow$
P7.1	$\leftrightarrow$	POUT1	$\rightarrow$
P7.2	$\leftrightarrow$	POUT2	$\rightarrow$
P7.3	$\leftrightarrow$	POUT3	$\rightarrow$
P7.4	$\leftrightarrow$	CC28IO	$\leftrightarrow$
P7.5	$\leftrightarrow$	CC29IO	$\leftrightarrow$
P7.6	$\leftrightarrow$	CC30IO	$\leftrightarrow$
P7.7	$\leftrightarrow$	CC31IO	$\leftrightarrow$

### Port P3

Les broches du Port P3 sont des entrées/sorties logiques ou bien des fonctions d'entrée ou de sortie de différents timers, de l'unité ASC0 (série asynchrone) et de l'unité SSC (série synchrone).

Fonction	Sens	Fonction	Sens
P3.0	$\leftrightarrow$	T0IN	$\leftarrow$
P3.1	$\leftrightarrow$	T6OUT	$\rightarrow$
P3.2	$\leftrightarrow$	CAPIN	$\leftarrow$
P3.3	$\leftrightarrow$	T3OUT	$\rightarrow$
P3.4	$\leftrightarrow$	T3EUD	$\leftarrow$
P3.5	$\leftrightarrow$	T4IN	$\leftarrow$
P3.6	$\leftrightarrow$	T3IN	$\leftarrow$
P3.7	$\leftrightarrow$	T2IN	$\leftarrow$

Pour la signification des fonctions on pourra se rapporter au polycopié de cours. De plus le port P3 est un port 16 bits et les broches P3.10 et P3.11 correspondent aux lignes TxD0 et RxD0 de l'interface série ASC0.

### Port P2

Les fonctions alternatives du port P2 sont les entrées capture/sorties comparaison de l'unité CAPCOM1 et les interruptions externes rapides.

Fonction	Sens	Fonction	Sens	Fonction	Sens
P2.0	↔	CC0IO	↔		
P2.1	↔	CC1IO	↔		
P2.2	↔	CC2IO	↔		
P2.3	↔	CC3IO	↔		
P2.4	↔	CC4IO	↔		
P2.5	↔	CC5IO	↔		
P2.6	↔	CC6IO	↔		
P2.7	↔	CC7IO	↔		
P2.8	↔	CC8IO	↔	EX0IN	←
P2.9	↔	CC9IO	↔	EX1IN	←
P2.10	↔	CC10IO	↔	EX2IN	←
P2.11	↔	CC11IO	↔	EX3IN	←

### Port P5

Le port P5 fonctionne uniquement en entrée, il est destiné aux entrées analogiques mais les lignes de ce port peuvent être aussi utilisées en entrées logiques ou encore en entrées externes de certains timers.

Fonction	Sens	Fonction	Sens	Fonction	Sens
P5.0	←	AN0	←		
P5.1	←	AN1	←		
P5.2	←	AN2	←		
P5.3	←	AN3	←		
P5.4	←	AN4	←		
P5.5	←	AN5	←		
P5.6	←	AN6	←		
P5.7	←	AN7	←		
P5.8	←	AN8	←		
P5.9	←	AN9	←		
P5.10	←	AN10	←	T6EUD	←
P5.11	←	AN11	←	T5EUD	←
P5.12	←	AN12	←	T6IN	←
P5.13	←	AN13	←	T5IN	←
P5.14	←	AN14	←	T4EUD	←
P5.15	←	AN15	←	T2EUD	←

### 1.3 Les interruptions

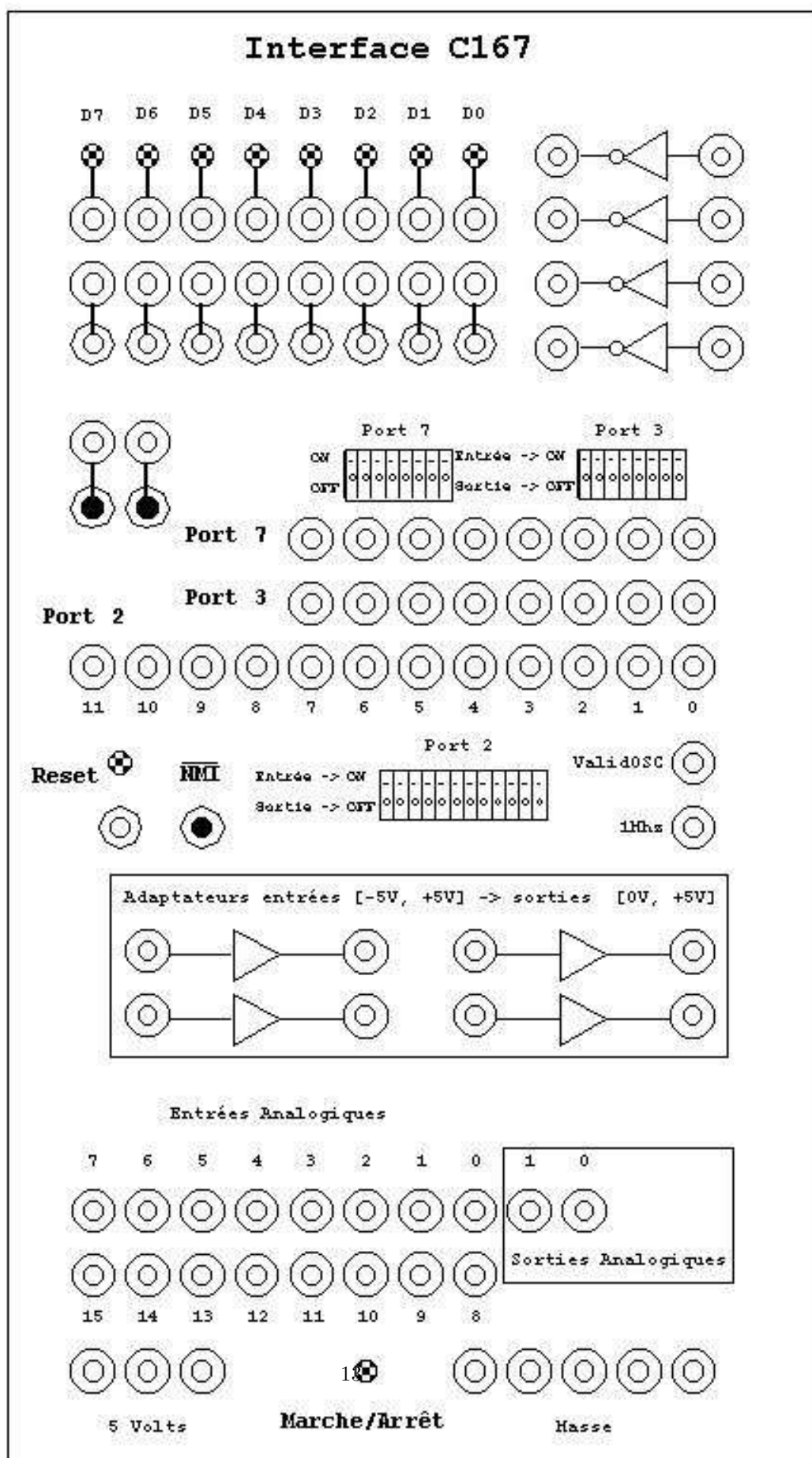
Le C167 offre 56 sources d'interruptions associées à des périphériques. Pour une source d'interruption donnée, notée par exemple source, le registre de contrôle de l'interruption est sourceIC, le bit d'autorisation est sourceIE, et le drapeau d'interruption est sourceIR, à cette interruption est associée un numéro de trappe auquel correspond une adresse dans la table des vecteurs d'interruption (adresse =  $4 \times$  numéro de trappe); pour T0 nous aurons respectivement T0IC, T0IE, T0IR, le numéro de trappe est 0x20, l'adresse du vecteur d'interruption est 0x000080.

Nous rappelons ici pour certaines interruptions le numéro de trappe qu'il est nécessaire de connaître pour mettre en œuvre des fonctions d'interruptions.

Source de l'interruption (ou du service PEC)	nom	numéro de trappe
CAPCOM registre 0	CC0	0x10
...	...	...
CAPCOM reg. 8 ou interr. externe EX0	CC8	0x18
CAPCOM reg 9 ou interr. externe EX1	CC9	0x19
CAPCOM reg 10 ou interr. externe EX2	CC10	0x1A
CAPCOM reg 11 ou interr. externe EX3	CC11	0x1B
...	...	...
CAPCOM registre 31	CC31	0x46
CAPCOM Timer 0	T0	0x20
CAPCOM Timer 1	T0	0x21
CAPCOM Timer 7	T0	0x3D
CAPCOM Timer 8	T0	0x3E
GPT1 Timer 2	T2	0x22
GPT1 Timer 3	T3	0x23
GPT1 Timer 4	T4	0x24
GPT2 Timer 5	T5	0x25
GPT2 Timer 6	T6	0x26
GPT2 CAPREL	CR	0x27
AD Conversion terminée	ADC	0x28
AD Erreur d'écrasement	ADE	0x29
ASC0 Transmission	S0T	0x2A
ASC0 Buffer de transm. transféré	S0TB	0x47
ASC0 Réception	S0R	0x2B
ASC0 Erreur Réception	S0E	0x2C
PWM canal 0..3	PWM	0x3F

Les interruptions externes rapides EX0 à EX7 ont pour vecteur d'interruption le vecteur de l'interruption normale, c'est à dire celui de la voie CAPCOM correspondante (de CC8 pour EX0 à CC15 pour EX7).

## Interface C167 – Sérigraphie Face avant -



## 2 Environnement logiciel

Un chapitre du polycopié de cours a été consacré au système de développement qui sera utilisé en Travaux Pratiques. Ces outils sont des outils classiques GNU adaptés pour le C167 par la société Hightec.

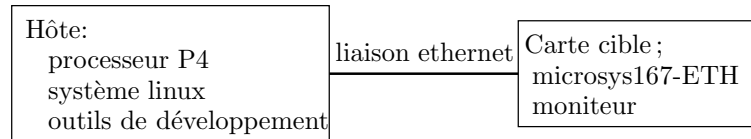


FIG. .1 – Ensemble station de développement et carte cible

Dans le cas de l'installation en salle de TP, une machine linux de nom XXX communique par liaison ethernet avec une carte nommée **mXXX**, par exemple à la machine *tamia* est associé la carte micro-contrôleur *mtamia* (m pour micro-contrôleur).

### 2.1 Compilation et édition de liens

Le compilateur gcc166 est invoqué par la commande :

```
gcc166 -Wall -m7 -g -o prog prog.c
```

(m7 précise que la compilation doit se faire pour une cible C167).

### 2.2 Téléchargement

Le chargement de l'exécutable sur la cible et la mise au point à distance se font par l'intermédiaire du débogueur (tgdb166, adaptation du débogueur gdb).

Lancement: **tgdb166**

Une fenêtre débogueur s'ouvre (cf. figure page suivante).

Elle présente une barre de menu (**File, Options, Running, ...**) et trois fenêtres:

- une fenêtre source,
- un bandeau de commande: il présente un certain nombre de commandes sous forme d'icônes,
- une fenêtre de travail: entrée des commandes et affichage des résultats des commandes.

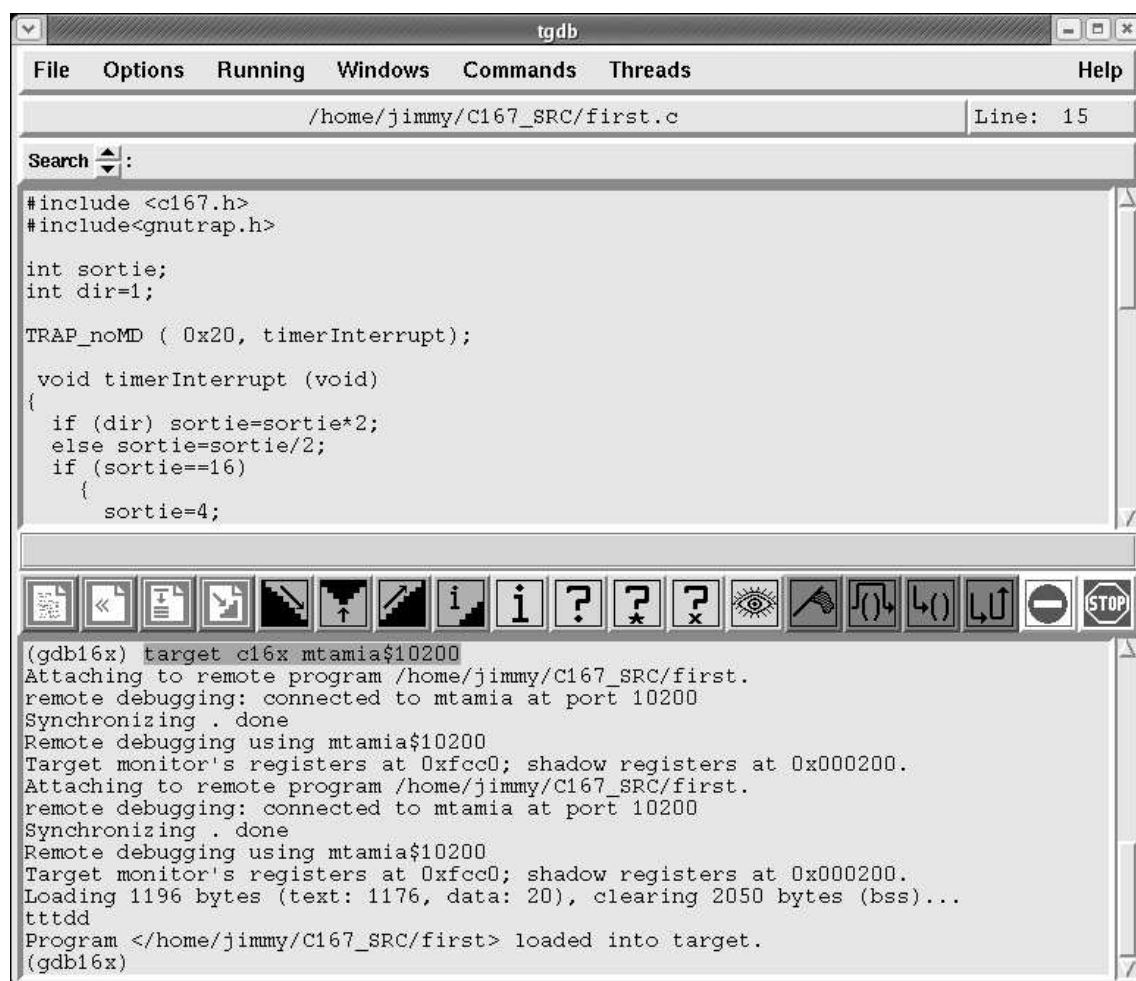
La connexion du débogueur sur la machine XXX au moniteur de la carte cible nommée mXXX est obtenue par la commande target:

```
target c16x mXXX$10200
```

soit sur la machine *tamia*:

```
target c16x mtamia$10200
```

c16x désigne le type de carte(C167), mXXX le nom de la carte, \$10200 est un service réseau pour le téléchargement.









## 2.3 Mise au point, le débogueur

Une session de travail type (par exemple sur *tamia* et la carte *mtamia*) se présente ainsi:

- édition du fichier source (`emacs prog.c &` par exemple).
- compilation: `gcc166 -m7 -g -o prog prog.c`
- lancement du débogueur: `tgdb166`

A partir de l'interface graphique de *tgdb*:

- choix du fichier à télécharger  
Menu **File**, puis **Load Program**, choix de `prog` dans la liste des programmes présentés,
- connexion: `target c16x mtamia$10200` dans la fenêtre inférieure.  
Après demande de confirmation du chargement, et reset éventuel de la carte cible, le programme sera chargé.
- ce programme doit apparaître dans la fenêtre source, sinon repasser dans **File**, puis **Visit** et choisir le programme `prog.c` ou utiliser l'icône: .
- le programme peut être lancé (icône  du bandeau de commande)
- ou exécuté pas à pas (sans entrer dans les fonctions  ou bien en pas à pas dans les fonctions .

- des points d'arrêt sont placés et supprimés sur un clic du bouton droit de la souris,
- des variables peuvent être visualisées en les sélectionnant et cliquant sur l'icône 
- des variables peuvent être suivies lors de la mise au point dans une fenêtre d'affichage (menu **Windows** puis **Watches**). La sélection des variables se fait par sélection avec la souris, puis clic sur l'icône 

## 2.4 Accès aux registres et aux bits de l'espace adressable par bits

Un fichier à inclure `c167.h` contient pour tous les registres du micro-contrôleur une déclaration du type suivant :

```
#define registre (* (unsigned int *) adresse)
```

soit un forçage de type de la constante adresse vers un pointeur et une indirection de ce pointeur, le résultat est associé au registre. Ceci permet un accès au registre en écriture et en lecture par une affectation :

- lecture `valeur=registre;`
- écriture `registre=donnee;`

Des macros incluses automatiquement par référence à `c167.h` permettent l'accès aux bits des registres adressables par bits, les principales macros (voir polycopié de cours et travaux dirigés) sont:

- `SET_SFRBIT(BIT)`: met à 1 le bit BIT
- `CLR_SFRBIT(BIT)`: met à 0 le bit BIT
- `SFR_BIT(BIT)` : renvoie la valeur du bit BIT
- `WAIT_UNTIL_BIT_SET(BIT)` attend le passage à 1 du bit BIT
- `WAIT_UNTIL_BIT_CLR(BIT)` attend le passage à 0 du bit BIT

La mise en place des interruptions s'effectue grâce à des macros contenues dans le fichier `gnutrap.h`:

```
TRAP(num_trap,nom_fonction)
```

```
TRAP_noMD(num_trap,nom_fonction)
```



# Annexe: Fonctions d’affichage sur une console

La mise au point de vos programmes s’exécutant sur la cible est faite avec le débogueur (tgdb166) par insertion de points d’arrêt, visualisation de variables en ces points, exécution de fonctions pas à pas ...

L’absence d’une console où afficher le contenu des variables en cours d’exécution est pénalisant dans les cas où le suivi des variables est nécessaire pour une mise au point dynamique. Aussi, afin de faciliter la mise au point de votre programme s’exécutant sur la cible un certain nombre de fonctions sont proposées. Elles permettent l’affichage sur une console dédiée, de caractères, de tableaux de caractères et d’entiers.

Les prototypes des fonctions permettant la trace des variables sont :

```
void init_ASC0_384(void);
void putchar_console(char c);
void puts_console(char *chaine);
void printf_entier_console(char* mess, int var);
```

Ces prototypes sont déclarés dans le fichier **commc167.h**, les fonctions correspondantes sont dans une bibliothèque *libtpc167* (référéncée par **-ltpc167** à l’édition de liens).

## 1 Les fonctions

Toutes les variables sont communiquées de la cible au PC Linux au travers de la ligne série à la vitesse de 38400 bauds.

**init\_ASC0\_384()** initialise la communication côté C167.

**putchar\_console(c)** affiche le caractère c sur la console.

**puts\_console(str)** affiche la chaîne de caractères str sur la console.

Ces deux fonctions sont équivalentes à putchar() et puts().

**printf\_entier\_console(“format”,var)** affiche un entier de taille 8 ou 16 bits, non signé ou signé sur la console.

Comme pour une entrée-sortie formatée, format est une chaîne de caractères incluant du texte à éditer et un spécificateur de format %c, %d, %i, %x, %o ... Comme vous pouvez le remarquer printf\_entier\_console() est limité aux entiers et ne permet pas d’afficher avec les spécificateurs %s (faire avec puts\_console pour les chaînes) ou %f (pas d’affichage de réels). D’autre part printf\_entier\_console() prend un seul paramètre (et non une liste de paramètres), et n’affichera donc qu’une donnée.

L’objectif est de suivre des variables “critiques” en cours d’exécution, la taille des données que vous pouvez transmettre est limitée à 30 caractères (par émission).

## 2 Exemples d'utilisation

```
#include <commc167.h>
/* déclarations */
unsigned char paire;
char string[20];
int mot;
/* initialisation de la ligne série */
init_ASCO_384();

/* affichage caractère */
putchar_console(paire);
putchar_console('\n');
// ou
printf_entier_console('paire= %c \n', paire); /* caractère */
printf_entier_console('paire= %x \n', paire); /* hexa */
printf_entier_console('paire= %d \n', paire); /* décimal */

/* affichage tableaux de caractères */
puts_console("tableau de caracteres\n");
puts_console(string);
puts_console("\n");
// ou
putchar_console('\n');

/* affichage entier */
printf_entier_console("mot = %d \n", mot);
printf_console("mot= %x \n", mot);
```

Compilation et édition de liens :

```
gcc166 -m7 -Wall -g -o prog prog.c -ltpc167
```

Lancement de la console d'affichage sur le PC Linux: **consolec167**

le processus **consolec167** initialise la communication série côté PC et attend les données pour affichage.