BTS INFORMATIQUE ET RESEAUX
POUR L'INDUSTRIE ET LES SERVICES TECHNIQUES

Session 2007

Epreuve E.4
Etude d'un Système Informatisé

Prélèvements sur sites volcaniques

Annexes

ANNEXE A1
Documentation de l'UART 16C550A (*extraits*)

16C550A UART
Récepteur / Transmetteur Asynchrone Universel

Caractéristiques

- Interface de communication asynchrone utilisant les signaux standards de contrôle Modem (CTS, RTS, DSR, DTR, RI et DCD).
- Interface série totalement programmable:
  - 5, 6, 7 ou 8 bits par caractères
  - Gestion de la parité
  - 1, 1.5 ou 2 bits d'arrêt
  - Vitesse jusqu'à 256kbds
- Gestion des interruptions de transmission, de réception, d'état de ligne et de données contrôlées indépendamment.
- FIFO activable et paramétrable de 16 caractères en transmission et réception de données.
- Générateur de Baud programmable pour une fréquence interne de 16*Baud.
- Logiciellement compatible avec le 16C450.

Description Générale

L'UART 16C550A est un circuit d'interface asynchrone série pour microprocesseur. Ces caractéristiques le rendent logiciellement compatible avec le 16C450 et le plus ancien 8250. Le circuit 16C550A totalement paramétrable et programmable est prévu pour l'interfaçage de liens de type série asynchrone. Il intègre deux FIFO pour diminuer le nombre d'interruptions du microprocesseur. Il inclus un générateur de fréquence interne de 16 coups d'horloge par bit de donnée qui autorise le transfert des données jusqu'à la vitesse de 256kbs.

Programmation

La sélection de l'un des 12 registres internes est effectuée à l'aide des 3 signaux d'adresse A0, A1 et A2 et d'un bit interne DLAB appartenant au registre de contrôle de ligne. Le tableau récapitulatif des registres internes par rapport à l'adresse de base et l'état de DLAB est donné au tableau A1.1.

Tableau A1.1. Plan d'adressage de l'UART 16C550A

Adresse relative	Nom	DLAB	Fonction en lecture	Fonction en écriture
0	THR/RBR	0	Tampon de réception	Tampon d'émission
	DLLB	1	Registre de poids faible du diviseur de fréquence (LSB)	
1	IER	0	Masque des interruptions *	
	DLHB	1	Registre de poids fort du diviseur de fréquence (MSB)	
2	IIR/FCR	X	Registre d'état des interruptions *	Contrôle des FIFO *
3	LCR	X	Registre de contrôle de ligne	
4	MCR	X	Registre de contrôle du modem	
5	LSR	X	Registre d'état de ligne	
6	MSR	X	Registre d'état du modem	
7	SR	X	Registre de personnalisation *	

* Non documenté dans l'annexe

Description des registres

- Tampon de réception (THR)
Ce registre contient le dernier caractère reçu.

- Tampon d'émission (RBR)
Ce registre contient le caractère qui sera émis par copie dans le registre de sérialisation.

- Registre du diviseur de fréquence (DLHB / DLLB)

Le registre du diviseur de fréquence sur 16 bits est accessible aux adresses relatives 0 et 1 à condition que le bit DLAB (*Divisor Latch Access Bit*) soit à 1. Ce registre permet de programmer le générateur de fréquence interne pour régler le débit binaire sériel. La fréquence du générateur interne est 16 fois plus grande que celle du débit binaire. Le calcul du diviseur est effectué soit à l'aide des tableaux A1.2 et A1.3 ou avec la relation :

Diviseur = Fréquence_horloge / (Débit_binaire x 16)

Où le débit binaire sériel est exprimé en Bauds et la fréquence d'horloge de l'UART 16C550A en Hz.

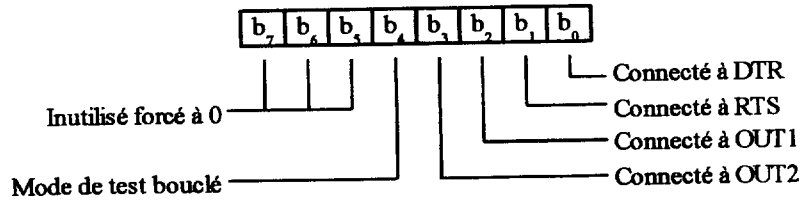
Tableau A1.2. Vitesse pour un quartz **1,8432MHz**

Vitesse en Bauds	Diviseur	Erreur %
75	1536	
110	147	0,026
150	768	
300	384	
600	192	
1200	96	
1800	64	
2000	58	0,69
2400	48	
3600	32	
4800	24	
7200	16	
9600	12	
19200	6	
38400	3	
57600	2	
115200	1	

Tableau A1.3. Vitesse pour un quartz **3,072MHz**

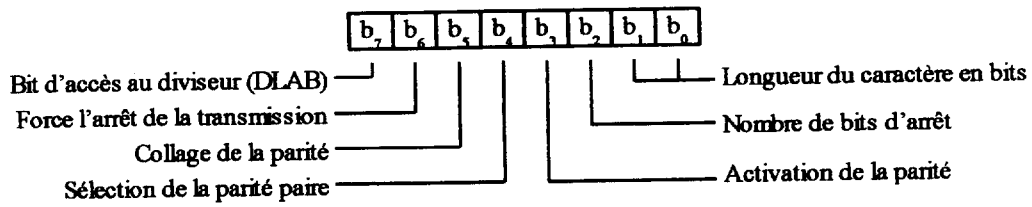
Vitesse en Bauds	Diviseur	Erreur %
75	2560	
110	1745	0,026
150	1280	
300	640	
600	320	
1200	160	
1800	107	0,312
2000	96	
2400	80	
3600	53	0,628
4800	40	
7200	27	1,23
9600	20	
19200	10	
38400	5	

- Registre de contrôle du modem (MCR)



Le bit de poids faible est connecté à la sortie DTR (Data Terminal Ready) et permet d'indiquer au Modem que le terminal est prêt à envoyer des données. Le positionnement à 1 de ce bit force à 0 la sortie DTR et inversement. Le bit 1 est connecté à la sortie RTS (Request To Send) et permet au terminal de demander au Modem une émission de données. Le positionnement à 1 de ce bit force à 0 la sortie RTS et inversement.
Les bits 2 et 3 permettent de contrôler respectivement les sorties utilisateur OUT1 et OUT2. Comme précédemment le contrôle de ces sorties est inversé.
Le bit 4 permet de mettre en œuvre une boucle interne afin de tester le transfert de données. Pendant cette phase de test, la sortie est forcée à 1 et l'entrée déconnectée extérieurement mais connectée en interne à la sortie du registre de sérialisation de l'émission de données. Les données émises seront ainsi transmises vers le registre de réception. Les bits 5 à 7 sont en permanence à 0.

- Registre de contrôle de ligne (LCR)



La longueur du caractère transmis est déterminé par les deux bits de poids faible et est réglable entre 5 et 8 bits. Le tableau A1.4 montre les diverses possibilités de réglage du nombre de bits. Le nombre de bits d'arrêt est aussi paramétrable et dépend de la valeur du bit 2 mais aussi de la taille du caractère transmis comme le montre le tableau A1.5.
La parité peut être activée à l'aide du bit 3. La mise à 1 du bit 4 permet de contrôler si la parité sera paire ou impaire pour la mise à 0. Le collage de parité, bit 5, force la parité à 0 si celle-ci est sélectionnée paire ou à 1 si celle-ci est sélectionnée impaire. Le collage de parité n'est possible que si la parité est activée.
Le bit 6 force un arrêt de l'émission en forçant la sortie à 0. Le bit 7 sélectionne l'accès au registre diviseur de fréquence (DLAB).

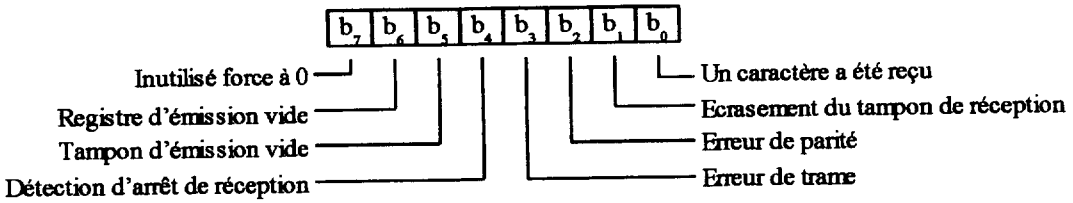
Tableau A1.4. Réglage du nombre de bits caractère

b1	b0	Taille
0	0	5
0	1	6
1	0	7
1	1	8

Tableau A1.5. Réglage du nombre de bits d'arrêt

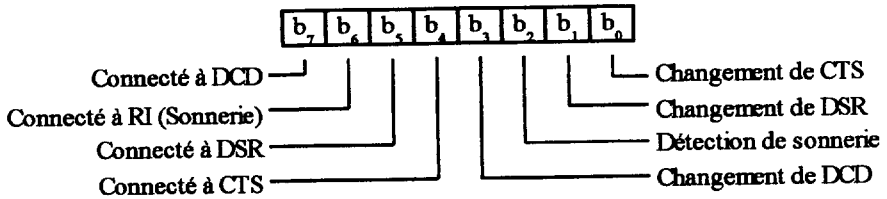
b2	Taille	Nb bits d'arrêt
0	X	1
1	5	1,5
1	6	2
1	7	2
1	8	2

- Registre d'état de ligne (LSR)



Le bit 0 indique qu'un caractère a été reçu et est disponible dans le tampon. Ce bit est remis à 0 lors de la lecture du tampon. Si le tampon n'est pas lu à temps et qu'un écrasement de la donnée est effectué par une nouvelle réception alors le bit 1 sera positionné. Le bit 2 sera positionné lors d'une détection d'erreur de parité sur une réception à condition que la parité soit activée. Si une réception indique une longueur incorrecte du bit d'arrêt correspondant à une erreur de trame alors le bit 3 sera positionné à 1. Le bit 4 indique le forçage d'un arrêt de réception par le modem.
L'émission d'un caractère est réalisée à l'aide du registre d'émission qui sera sérialisé. Dès que ce registre est vide le contenu du tampon d'émission y est transféré pour être à son tour émis. Si le tampon d'émission est vide l'indicateur binaire bit 5 est positionné à 1 pour indiquer qu'un nouveau caractère est demandé par le processus d'émission. D'autre part, si le registre d'émission et le tampon sont vides alors le bit 6 sera positionné à 1. En résumé, ce bit indique une rupture du flux d'émission. Ces bits, 5 et 6, sont mis à 0 dès qu'une nouvelle donnée a été écrite dans le tampon de réception par le microprocesseur. Les bits 1 à 5 sont remis à 0 lors d'une lecture du registre d'état par le microprocesseur.

- Registre d'état du modem (MSR)



Les bit 0,1 et 3 sont respectivement associés aux entrées CTS, DSR et DCD. Lors d'un changement de niveau sur une de ces trois entrées le bit correspondant est positionné à 1. Par contre le bit 2 qui est associé à l'entrée de détection de sonnerie du modem (entrée RI), n'est positionné à 1 que lors de l'apparition d'un front montant sur cette entrée. Ces quatre bits sont mis à 0 après la lecture du registre d'état du modem par le microprocesseur.
Les bits 4 à 7 sont respectivement connectés à travers un inverseur aux entrées CTS, DSR, RI, DCD.
Si le mode de test bouclé est activé par la mise à 1 du bit 4 du registre de contrôle du modem, ces bits 4 à 7 sont alors respectivement connectés en interne aux signaux de sorties RTS, DTR, OU1 et OUT2.

Annexe 2

Le protocole NMEA (*extraits*)

NMEA (*National Marine & Electronics Association*) est une association à but non lucratif fondée par un groupement de professionnels de l'industrie de l'électronique des instruments et périphériques marins, conjointement avec des fabricants, des distributeurs, des revendeurs, des institutions d'enseignements. L'association a pour but d'harmoniser et de standardiser les équipements de la marine.

NMEA est à l'origine de nombreux standards et en particulier du Standard NMEA-0183, utilisé dans les appareils GPS actuels.

Sous ce standard, toutes les données sont transmises sous la forme de trames ou phrases (*sentences*) constituées de caractères ASCII imprimables, elles commencent par le caractère '$' et sont terminées par les caractères [CR] Retour Chariot et [LF] Saut de ligne.

La longueur maximale d'une trame ne peut excéder 82 caractères.

Format simplifié des trames NMEA :

$	GP	<id>	<données>	*<checksum>	CR	LF

- les caractères GP indiquent qu'il s'agit d'une trame GPS (*Global Positioning System*)
- <id> est une suite de 3 caractères (GGA, GLL, GSV, ...) définissant le type de la trame
- les données sont séparées par des virgules, leur nombre dépend du type de trame
- la somme de contrôle (checksum) est précédée d'un caractère '*', elle est calculée par OU exclusif entre tous les caractères situés entre '$' et '*' . Ce champ est présenté sous forme hexadécimale, sa présence est en général facultative.
- Une trame NMEA ne comporte jamais de caractère Espace (0x20).

Détail d'une trame GGA avec checksum (CRLF non représentés) :

```
$GPGGA,124819,4807.048,N,02131.324,E,1,08,0.9,545.4,M,46.9,M,,*4C
```

GGA	= indicatif « Données d'acquisition Fix et Date - GPS »
124819	= acquisition du Fix à 12:48:19 UTC
4807.048,N	= Latitude 48° 7,048' Nord
02131.324,E	= Longitude 21° 31,324' Est
1	= Fix qualification (0 = non valide, 1 = Fix GPS, 2 = Fix DGPS, ...)
08	= nombre de satellites en poursuite
0.9	= dilution horizontale
545.4,M	= altitude en mètres au dessus du niveau moyen des océans (MSL : *Mean See Level*)
46.9,M	= correction de la hauteur de la géoïde en mètres par rapport à l'ellipsoïde WGS84
(champ vide)	= nombre de secondes écoulées depuis la dernière mise à jour DGPS
(champ vide)	= identification de la station DGPS
*4C	= préfixe + somme de contrôle

Détail d'une trame GLL sans checksum (CRLF non représentés) :

```
$GPGLL,4916.75,N,12311.12,W,225444,A
```

GLL	= indicatif « Positionnement Géographique Longitude / Latitude - GPS »
4916.75,N	= Latitude 49° 16,75' Nord soit 49° 16' 45" Nord
12311.12,W	= Longitude 123° 11,12' Ouest soit 123° 11' 7" Ouest
225444	= acquisition du Fix à 22:54:44 UTC
A	= indication de données valides

ANNEXE A3
Documentation sur les Sémaphores RTAI

RTAI (*Real Time Application Interface*) est une extension libre du noyau Linux lui apportant des fonctionnalités temps réel dures.

À proprement parler, RTAI n'est pas un système d'exploitation en temps réel, tel que Vxworks, OS-9 ou QNX. Linux n'est pas et ne sera jamais un système temps réel. RTAI est un noyau temps réel à priorité fixe qui exécute Linux en temps que tâche de faible priorité. Pour celà, le noyau RTAI s'intercale entre le matériel et le noyau Linux d'origine.

Les tâches temps réel développées sont donc concurrentes du noyau Linux.

Comme les autres OS temps-réel, RTAI offre aux applications au moins les services suivants :

- Une couche de gestion de matériel traitant les interruptions ou les polling de processeur/périphérique ;
- Un ordonnanceur programmable traitant l'activation de processus, priorités, par tranche de temps ;
- Des moyens de communications notamment avec le noyau Linux.

Parmi les moyens de communication disponibles sous RTAI, on trouve des mécanismes comme la gestion des files FIFO, les messages, la mémoire partagée et les sémaphores...

Sémaphores

Description

Les fonctions de gestion des sémaphores appartiennent à la librairie FIFO de RTAI. Le fichier d'entête de cette librairie est *rtai_fifos.h*. Les sémaphores binaires peuvent avoir de multiples utilisations. Ils permettent notamment la synchronisation des échanges de données avec la mémoire partagée. Les fonctions de gestion des sémaphores peuvent être appelée par les tâches temps réel (espace noyau) ou les processus linux (espace utilisateur). Cependant, seules les fonctions non bloquantes sont utilisables par les tâches temps réel. Les six fonctions sont listées ci-dessous :

rtf_sem_init(fd, init_val);	*Non bloquant*
rtf_sem_wait(fd);	*bloquant*
rtf_sem_trywait(fd);	*Non bloquant*
rtf_sem_timed_wait(fd, ms_delay);	*Non bloquant*
rtf_sem_post(fd);	*Non bloquant*
rtf_sem_destroy(fd);	*Non bloquant*

int rtf_sem_init (unsigned int minor, int value)

Initialise et crée un sémaphore binaire identifié par un descripteur de fichier ou un numéro de FIFO. Cette fonction peut être utilisée dans les espaces utilisateur ou noyau.

Paramètres :
- *minor* est le numéro du sémaphore. Les sémaphores doivent être associé à une FIFO pour des besoins d'identification.
- *value* valeur initiale du sémaphore. Doit être choisie entre 0 et 1.

Valeur retournée : *0* en cas de succès, ou *EINVAL* si *minor* se réfère à un descripteur invalide.

int rtf_sem_destroy (unsigned int minor)

Efface un sémaphore binaire préalablement crée avec *rtf_sem_init()*. Cette fonction peut être utilisée dans les espaces utilisateur ou noyau. Une tâche bloquée sur ce sémaphore retournera une erreur s'il est détruit.

Paramètres : *minor* est le descripteur de sémaphore.

Valeur retournée : *0* en cas de succès, ou *EINVAL* si *minor* se réfère à un descripteur invalide.

int rtf_sem_post (unsigned int minor)

Rendre le sémaphore. Le sémaphore est activé et le premier processus qui attend la file démarrera. Cette fonction peut être utilisée dans les espaces utilisateur ou noyau.

Paramètres : *minor* est le descripteur de sémaphore.

Valeur retournée : *0* en cas de succès, ou *EINVAL* si *minor* se réfère à un descripteur invalide.

int rtf_sem_wait (unsigned int minor)

Prendre le sémaphore. Cette fonction attend qu'un évènement soit signalé par le sémaphore. La valeur du sémaphore est positionnée à 1 pour être testée et mise à 0. S'il y a 1, *rtf_sem_wait()* se termine immédiatement, sinon le processus appelant est bloqué et ajouté dans la file d'attente dans l'ordre de priorité déterminé par la priorité temps réelle POSIX. Le processus est débloqué s'il est en première place dans la file d'attente et que le sémaphore est positionné par une tâche ou si le sémaphore est détruit. Comme cette fonction peut être bloquante, elle ne doit être utilisée que dans l'espace utilisateur.

Paramètres : *minor* est le descripteur de sémaphore.

Valeur retournée : *0* en cas de succès, ou *EINVAL* si *minor* se réfère à un descripteur invalide.

int rtf_sem_timed_wait (unsigned int minor, int ms_delay)

Récupère le sémaphore avec un temps maximal. Cette fonction est presque identique à *rtf_sem_wait()* mais la libération du processus dans la file d'attente survient s'il est en première place dans la file d'attente et que le sémaphore est positionné par une tâche, si le temps imparti est écoulé ou si le sémaphore est détruit. Comme cette fonction peut être bloquante, elle ne doit être utilisée que dans l'espace utilisateur.

Paramètres : *minor* est le descripteur de sémaphore.
ms_delay est le délai d'attente en milliseconde.

Valeur retournée : *0* en cas de succès, ou *EINVAL* si *minor* se réfère à un descripteur invalide.

int rtf_sem_trywait (unsigned int minor)

Récupère le sémaphore sans blocage. Cette fonction est similaire à *rtf_sem_wait()* mais elle ne bloque jamais l'appelant. Si le sémaphore n'est pas libre *rtf_sem_trywait()* retourne immédiatement et le sémaphore demeure inchangé. Cette fonction n'est pas bloquante elle peut donc être utilisée dans les espaces utilisateur ou noyau.

Paramètres : *minor* est le descripteur de sémaphore.

Valeur retournée : *0* en cas de succès, ou *EINVAL* si *minor* se réfère à un descripteur invalide.