

1. Objectifs :

Dans cette manipulation, on va découvrir la manipulation des ports d'entrées-sortie du 68HC11. Pour ce faire, on se servira toujours le simulateur THRSim11, qui met à la disposition de l'utilisateur un certain nombre de périphériques pour le test et l'évaluation de son programme. Ainsi, on passera par :

- L'utilisation des ports parallèles pour la lecture et l'affichage d'une donnée en mode parallèle.
- L'utilisation de l'interface de communication série (SCI) pour la réception et l'émission d'un caractère
- Mise en œuvre du concept de sous-programme pour la génération d'une séquence d'attente. Ainsi, la relation entre le temps de simulation affiché et les cycles d'horloge dans le simulateur THRSim11 sera discutée.

2. Ports parallèles :

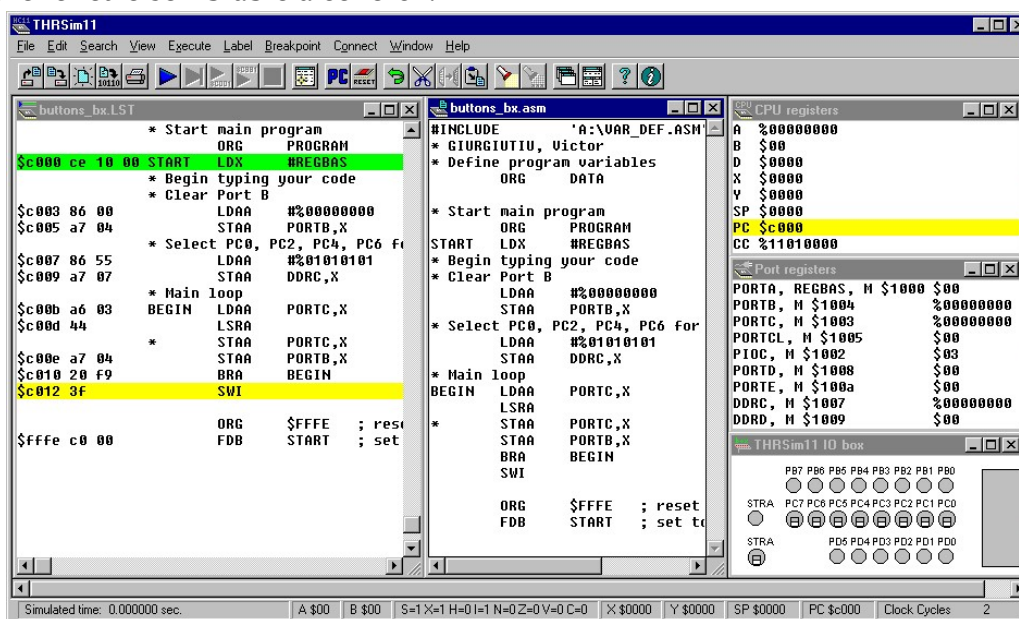
2.1. Exercice 1 :

Ecrire le programme pour lire, continuellement, les données présentes sur les broches du port C et les renvoie sur le port B.

Démarche :

- Ouvrir THRSim11 et Fermez la fenêtre Commandes.
- Ecrire le programme « Boutons.asm »
- Afficher les registres : *View, Registers, CPU Registrers* ;
- Ouvrir la boîte THRSim11 IO : *View, I/O Box* ;
- Assembler Boutons.asm.
- Définir un point d'arrêt à l'instruction « SWI » ;
- Définir des étiquettes standard : *Label, Set Standard Labels* ;
- Définir l'affichage de l'accumulateur A, PORTB et PORTC en binaire.
- Organiser les fenêtres pour un bénéfice maximum

On obtient une fenêtre semblable à celle-ci :



- Exécuter le programme : *Run* ;
- Actionner le bouton PC2 et noter l'effet.
- Refaire pour plusieurs boutons.

2.2. Exercice 2 :

Ecrire un programme pour afficher le chiffre hexadécimal (0-F) placé à la case mémoire d'adresse \$0300 (data) sur un afficheur 7 segments.

- Quelle est la condition pour allumer un segment ?
- Donner la table de correspondance entre les codes binaires et les codes hexadécimaux de tous les chiffres qu'on désire allumer (0 à F).
- Donner la directive pour placer ces codes hexa dans une table commençant à l'adresse \$0000.
- Ecrire le programme complet.

3. Interface de communication série (SCI) :

3.1. Exercice 1 : Transmission d'un caractère

Le principe de notre programme est le suivant :

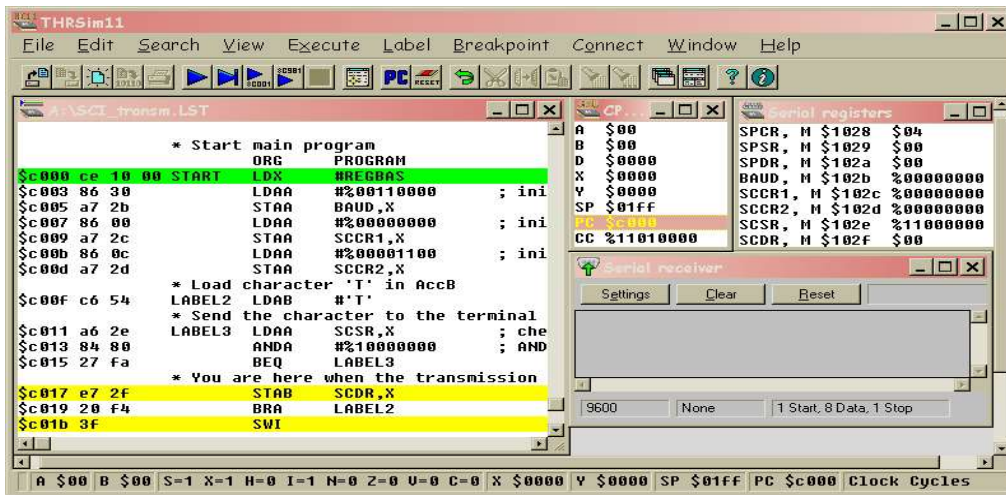
Après l'initialisation du SCI, le caractère à transmettre (T par exemple) est chargé dans accB. Ensuite, l'état du drapeau TDRE (registre de données de transmission vide) est testé dans une boucle. Lorsque TDRE est vrai, le contenu de accB est stocké dans le SCDR (registre de données de communication série). Cette opération réinitialise automatiquement TDRE.

Une première ébauche du programme «Transmit.asm » peut être :

```
START      LDX      #REGBAS
           LDAA     #%00110000
           STAA     BAUD,X
           LDAA     #%00000000
           STAA     SCCR1,X
           LDAA     #%00001100
           STAA     SCCR2,X
LABEL2     LDAB     #'T'
           * Emettre le caractère au terminal
LABEL3     LDAA     SCSR,X
           ANDA     #%10000000
           BEQ     LABEL3
           * ici le reg. de transmission est vide
           STAB     SCDR,X
           BRA     LABEL2
           SWI
```

Démarche :

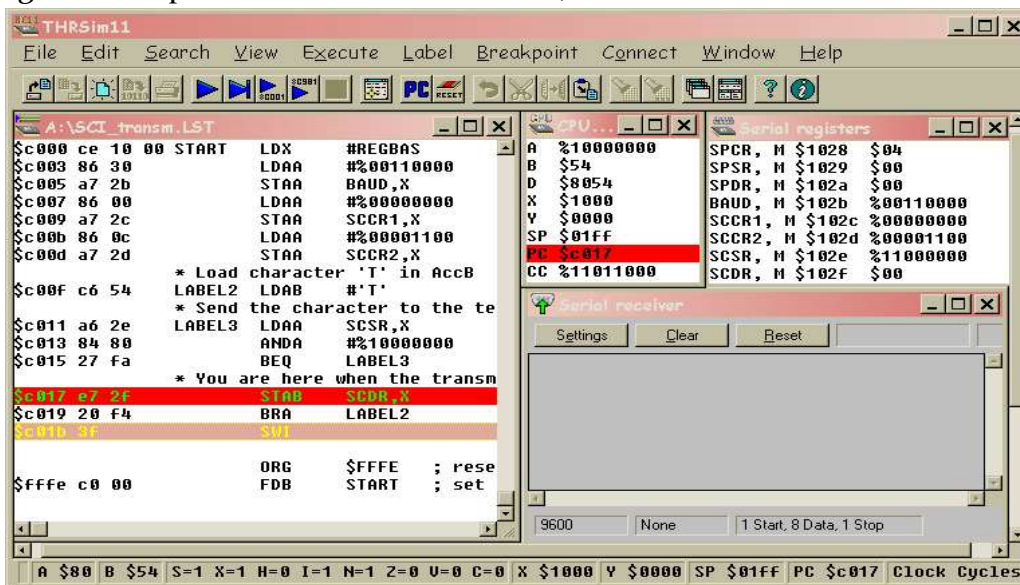
- Ouvrez THRSim1 et Fermez la fenêtre Commandes.
- Afficher les registres série : View, Registers, Serial ;
- Afficher le récepteur série : Viw, Serial Receiver ;
- Assemblez « Ttransmit.asm » ;
- Définir des points d'arrêt sur STAB SCDR, et sur SWI.
- Réinitialiser les registres : bouton RESET
- Définir des étiquettes standard : Label / Set Standard Labels.
- Définir l'affichage de A, BAUD, SCCR1, SCCR2 et SCSR en binaire.
- On obtient un écran ressemblant à ceci:



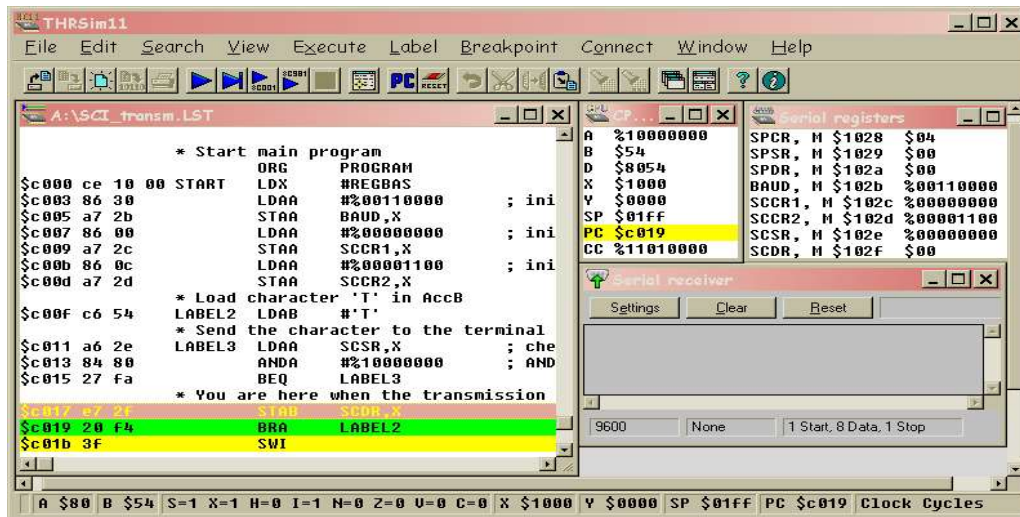
Que signifie la valeur de réinitialisation de SCSR ?

Exécution :

- Appuyer sur le bouton RUN.
- Indiquer la condition pour que le programme sort de la boucle
- Le programme accède à la ligne : STAB SCDR, X
- Cette ligne est un point d'arrêt. En ce moment, l'écran ressemble à ceci:



- Passez manuellement à la ligne suivante pour enregistre B dans SCDR.
- L'écran ressemble à ceci:



- Interpréter et conclure
- Appuyer sur RUN une autre fois
- Que constatez-vous ?
- Appuyer sur RUN 2 fois
- Conclusion

3.2. Exercice 2 : Réception d'un caractère

Voici une variante de programme pour la réception d'un caractère :

```

START      LDX      #REGBAS
           LDAA     #%00110000
           STAA    BAUD,X
           LDAA     #%00000000
           STAA    SCCR1,X
           LDAA     #%00001100
           STAA    SCCR2,X
LABEL0     NOP

LABEL1     LDAA     SCSR,X
           ANDA    #%00100000
           BEQ     LABEL1
           LDAB    SCDR,X
           STAB   PORTB,X
           BRA     LABEL0
           SWI
    
```

Démarche :

- Ouvrir THRSim11 et Fermez la fenêtre Commandes.
- Afficher les registres de ports et les registres série.
- Afficher émetteur série
- Assemblez « Recept.asm »
- Définir les points d'arrêt sur LDAB SCDR, X et SWI.
- Réinitialiser les registres.
- Définir des étiquettes standard.
- Définir l'affichage de BAUD, SCCR1, SCCR2, SCSR et l'accA sur binaire.
- Votre écran devrait ressembler à ceci :

The screenshot shows the THRSim11 software interface. The main window displays assembly code for an SCI reception program. The code includes initialization of BAUD=9600, setting up 8-bit words, and a loop that checks for reception and displays data through port B. The status bar at the bottom shows simulated time as 0.000000 sec and various register values.

```

SCI_recept.LST
$c000 ce 10 00 START LDX #REGBAS
          * GIURGIUTIU, Victor
$c003 86 30 LDAA #%0110000 ; initialize BAUD=9600
$c005 a7 2b STAA BAUD,X
$c007 86 00 LDAA #%00000000 ; initialize 8-bit word
$c009 a7 2c STAA SCCR1,X
$c00b 86 0c LDAA #%00001100 ; initialize TE and RE
$c00d a7 2d STAA SCCR2,X
$c00f 01 LABEL0 NOP
          * Wait for a keypress reception
          LABEL1 LDAA SCSR,X ; check if RDRF is set
$c012 84 20 ANDA #%0100000 ; AND with mask for RDRF
$c014 27 fa BEQ LABEL1
          * You are here when the reception reg. is full
$c016 e6 2f LDAB SCDR,X ; load SCI data in AccB
$c018 e7 04 STAB PORTB,X ; display data through port B
$c01a 20 f3 BRA LABEL0 ; loop back and do it again
$c01c 3f SWI
          ORG $FFFE ; reset vector
$fffe c0 00 FDB START ; set to start of program
  
```

Serial registers window:

A	%0100000	SPCR, M \$1028	\$04
B	\$41	SPSR, M \$1029	\$00
D	\$2041	SPDR, M \$102a	\$00
X	\$1000	BAUD, M \$102b	%00000000
Y	\$0000	SCCR1, M \$102c	%00000000
SP	\$01ff	SCCR2, M \$102d	%00000000
PC	\$c000	SCSR, M \$102e	%11000000
CC	%11010000	SCDR, M \$102f	\$41

Serial transmitter window:

Settings: 9600, None, 1 Start, 8 Data, 1 Stop

Port registers window:

PORTB, M \$1004 \$00

Simulated time: 0.000000 sec. A \$20 B \$41 S=1 X=1 H=0 I=1 N=0 Z=0 V=0 C=0 X \$1000 Y \$0000 SP \$01ff PC \$c000 Clock Cycles 2

- Exécuter le programme
- Justifier le déroulement en boucle du programme
- Tapez le caractère T dans la fenêtre de l'émetteur série et appuyez sur Send
- Interpréter le résultat
- Avancer manuellement et relever le changement des différents contenus
- Exécuter le programme pour plusieurs caractères