NF 02
2009 / 2010

Catherine Marque

SUJETS DE TD

Université de Technologie de Compiègne

TD NF02

TD N° 1 – Circuits combinatoires

I- Etude d'un additionneur

I-a) Le demi-additionneur (pas de Carry entrante)
- Définition des variables
- Table de vérité
- Simplification
- Schéma logique

I -b) Additionneur complet
- Défmition des variables
- Table de vérité
- Simplification
- Schéma logique
- Schéma pratique

II- Conception d'un codeur

On voudrait construire un circuit de commande d'une machine à café. La machine comporte quatre électrovalves contrôlant le dosage des matières premières : l'eau, le café, le sucre et le lait. Un signal de niveau 1 (d'état 1) sur l'électrovalve permet son ouverture. Les options offertes sont au nombre de cinq :
- café avec sucre
- café sans sucre
- capuccino
- lait avec sucre
- lait sans sucre

La pression sur un des boutons de sélection permet de sélectionner la boisson choisie. Donner le circuit logique permettant la commande des électrovalves.

III- Conception d'un décodeur

Une machine à sous comporte trois roues. Chaque roue contient quatre symboles : ♣ , ♦ , ♥ , ♠.

Il existe quatre combinaisons gagnantes. Chaque combinaison gagnante correspond à l'allumage d'une lampe.

1) Définissez les variables et les fonctions du circuit logique permettant le fonctionnement de la machine.

2) Pour simplifier le problème, on se limite à deux symboles seulement pour chaque roue : • et ■. Les combinaisons gagnantes seront au nombre de 2 : gain maximal si on a 3 • , gain minimal si on a 2 • adjacents et les autres possibilités sont perdantes.

Donnez le circuit logique commandant l'allumage des trois lampes :

L1 : gain maximal

L2 : gain minimal

L3 : pas de gain

TD N° 2 – Circuits logiques

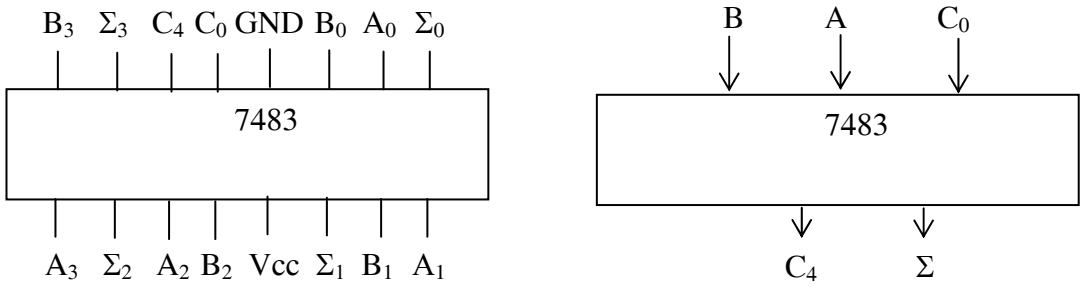
I- Additionneur à 4 bits

Le circuit 7483 est un additionneur complet à 4 bits. Il effectue l'opération d'addition entre les deux mots binaires :

A : $A_3\ A_2\ A_1\ A_0$ et B : $B_3\ B_2\ B_1\ B_0$

Il reçoit une retenue entrante C0 et donne une retenue C4.

Le résultat est le mot 4 bits $\Sigma = \Sigma_3\ \Sigma_2\ \Sigma_1\ \Sigma_0$ tel que $\Sigma = A + B$.



1) À l'aide de deux circuits 7483, réalisez un additionneur à 8 bits.

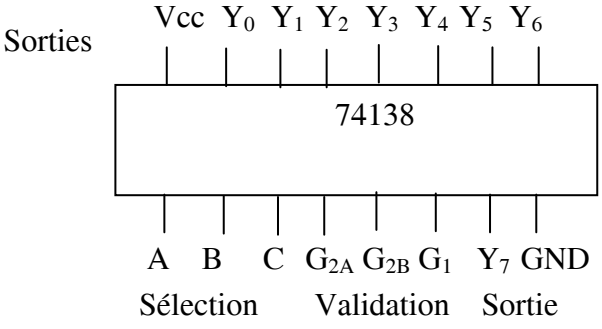
2) En regardant la table de vérité de la porte XOR, on peut dire qu'elle remplit la fonction d'une porte inverseur programmable.

3) Pour obtenir A-B on fait A+(-B) où -B est le complément à 2 de B. Le complément à 2 est obtenu en inversant les bits et en ajoutant 1.

Réalisez à l'aide d'un 7483 et d'un 7486 (4 portes XOR) un circuit Additionneur-Soustracteur à 4 bits.

II- Décodeur

Le circuit 74138 est un décodeur binaire-décimal (3 entrées - 8 sorties).



Données techniques

Type	Temps de propagation	Consommation
74LS138	22 ns	32 mW
74S138	8 ns	245mW

Table de fonctionnement (de vérité)

G_1	G_2	C	B	A	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
x	1	x	x	x	1	1	1	1	1	1	1	1
0	x	x	x	x	1	1	1	1	1	1	1	1
1	0	0	0	0	0	1		.	.	.		1
1	0	0	0	1	1	0	1					
1	0	0	1	0		1	0	.				.
1	0	0	1	1	.		1	0	.			.
1	0	1	0	0	.			.	0	.		.
1	0	1	0	1	.				.	0		
1	0	1	1	0						.	0	1
1	0	1	1	1	1		.	.	.		1	0

Les deux premières lignes correspondent au circuit non sélectionné. Les lignes d'après correspondent à l'état de fonctionnement en décodage ($G_2 = G_{2A} + G_{2B}$).

1) À partir de ce circuit, réalisez un circuit décodeur 4- 16, avec une entrée « validation ».

III- Multiplexeur (commutateur électronique)

Le circuit 74151 est un multiplexeur 8 voies en 1. L'entrée S est une entrée de validation.

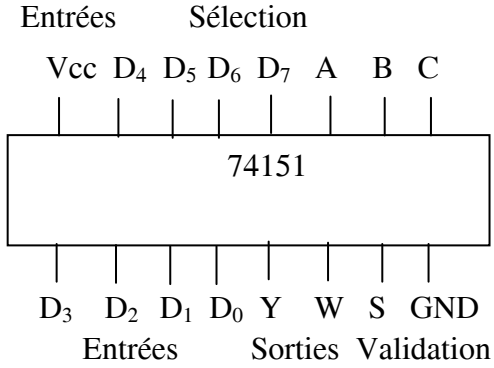


Table de fonctionnement

Entrées				Sorties	
C	B	A	S	Y	W
x	x	x	1	0	1
0	0	0	0	D_0	$\overline{D_0}$
0	0	1	0	D_1	$\overline{D_1}$
0	1	0	0	D_2	$\overline{D_2}$
0	1	1	0	D_3	$\overline{D_3}$
1	0	0	0	D_4	$\overline{D_4}$
1	0	1	0	D_5	$\overline{D_5}$
1	1	0	0	D_6	$\overline{D_6}$
1	1	1	0	D_7	$\overline{D_7}$

1) Réalisez un multiplexeur type 74151 à l'aide d'un décodeur 74138.
2) Réalisez un multiple xe ur 16 voies en 1 à base de 74151.

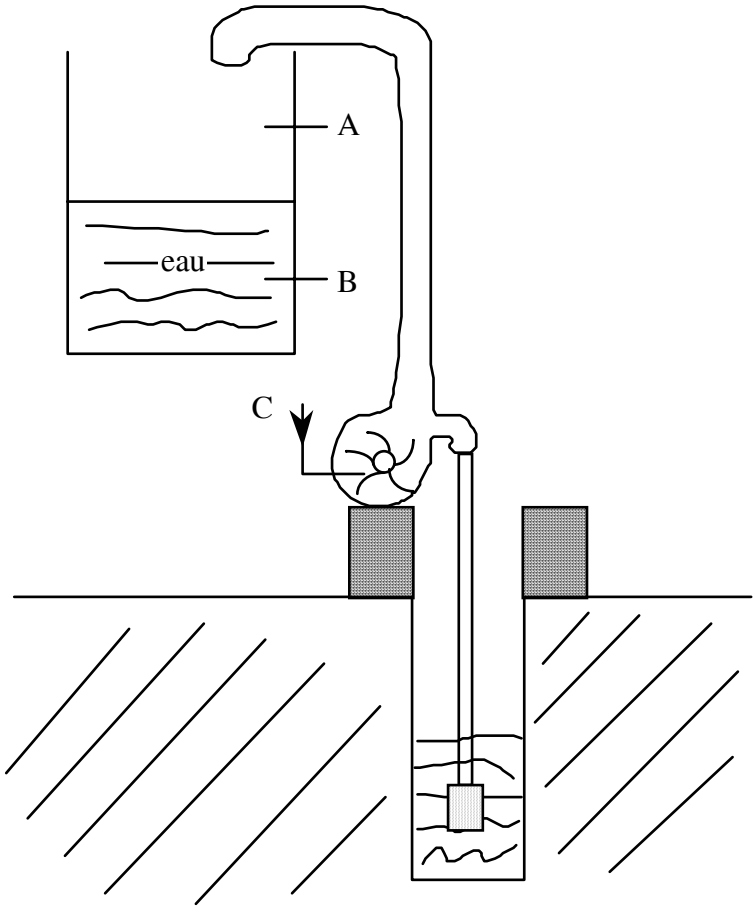
TD N° 3 - Bascules

I- Bascule R-S

1) En utilisant des portes NAND, réaliser une bascule R-S (avec table de fonctionnement)

2) En utilisant des portes NOR, réaliser une bascule R-S (avec table de fonctionnement)

3) Dans une maison de campagne le réservoir d'eau est situé sur le toit de la maison ; il est alimenté à partir d'un puits à l'aide d'une pompe.



puits

Pour automatiser la pompe d'eau, on a mis deux détecteurs de niveau d'eau dans le réservoir : l'un au bas du réservoir et l'autre au haut du réservoir. Les détecteurs donnent une valeur 1 si l'eau atteint leur niveau.

La pompe peut être commandée par une entrée de commande C:

- si C = 1 => pompe active
- si C = 0 => pompe éteinte

3-a) Réalisez le circuit de commande capable d'activer la pompe si l'eau atteint le niveau bas et de l'arrêter si l'eau atteint le niveau haut, en utilisant une bascule R-S de type NAND.

3-b) Même question avec une bascule de type NOR.

Note : réalisez chaque solution avec une seul genre de porte logique.

II. Bascules JK

Le magasin TROC souhaite installer sur sa vitrine, une enseigne lumineuse. L'allumage de chaque lettre sera suivant le cycle :

TROC les 4 lettres sont éteintes
𝕋ROC
T𝕉OC
TR𝕆C
TRO𝕔
𝕋R𝕆𝕔

Comment commander l'allumage de chaque lettre à l'aide de bascules JK (boîtier 74LS73A) ?
On souhaite qu'à la mise sous tension toutes les lettres soient éteintes.
Que faut-il modifier pour que l'allumage de toutes les lettres dure 2 fois plus longtemps ?

Table de vérité de la bascule JK donnée par le constructeur

INPUTS				OUTPUTS	
CLEAR	CLOCK	J	K	Q	$\overline{Q}$
L	X	X	X	L	H
H	0	L	L	Q_0	$\overline{Q}_0$
H	↓	H	L	H	L
H	↓	L	H	L	H
H	↓	H	H	TOGGLE	TOGGLE
H	H	X	X	Q_0	$\overline{Q}_0$

TD N° 4 - Compteurs

I- Les compteurs binaires

Réalisez à l'aide de chacun des compteurs 74163, 74169, 74191, 74193 (dont la documentation est fournie) un compteur à dix, c'est à dire un circuit qui compte de 0, l, ... jusqu'à 9 et qui reprenne à 0 ... etc.

Remarques :

- Le nombre 10 ne doit pas figurer sur les sorties du compteur.
- Chaque chiffre (0 ... 9) doit rester pendant une période d'horloge sur les sorties du compteur.
- Donner plusieurs méthodes s'il en existe.
- Dans la suite, A et QA sont les bits de poids faibles.

II- Réalisation d'un compteur-décompteur 8 bits (74193)

Mise en cascade de 2 compteur-décompteurs 4 bits. (un compteur pour les bits de poids faibles LSB et un autre compteur pour les bits de poids forts MSB).

III- Registre à décalage bidirectionnel à 4 bits (74LS194 boîtier 16 pattes)

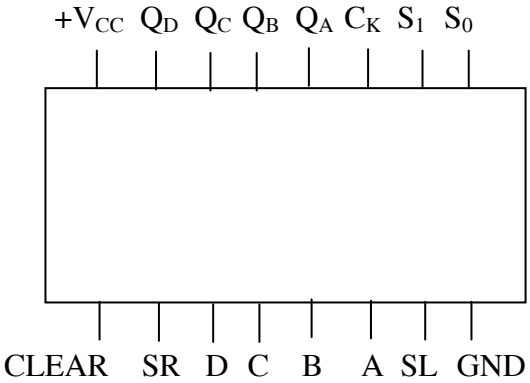
III-l) Ce circuit dispose de :

- 1 entrée d'horloge active sur un front montant
- 1 entrée CLEAR asynchrone
- 4 lignes d'entrée D C B A pour un chargement parallèle du registre à décalage (A est le LSB)
- 2 bits de contrôle S_0 et S_1 qui permettent la sélection de 4 modes de fonctionnement:

1) chargement parallèle synchrone
2) Décalage à gauche (dans le sens Q_A -> Q_D)
3) Décalage à droite (dans le sens Q_D -> Q_A)
4) Arrêt du fonctionnement ou mémorisation

- 4 lignes de sortie Q_D Q_C Q_B Q_A (Q_A = LSB).

III-2) Brochage du circuit



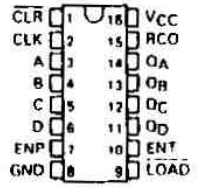
SN54160 THRU SN54163, SN54LS160A THRU SN54LS163A, SN54S162, SN54S163, SN74160 THRU SN74163, SN74LS160A THRU SN74LS163A, SN74S162, SN74S163
SYNCHRONOUS 4-BIT COUNTERS

OCTOBER 1976 – REVISED MARCH 1988

'160, '161, 'LS160A, 'LS161A . . . SYNCHRONOUS COUNTERS WITH DIRECT CLEAR
'162, '163, 'LS162A, 'LS163A, 'S162, 'S163 . . . FULLY SYNCHRONOUS COUNTERS

- Internal Look-Ahead for Fast Counting
- Carry Output for n-Bit Cascading
- Synchronous Counting
- Synchronously Programmable
- Load Control Line
- Diode-Clamped Inputs

SERIES 54', 54LS', 54S' . . . J OR W PACKAGE
SERIES 74' . . . N PACKAGE
SERIES 74LS', 74S' . . . D OR N PACKAGE
(TOP VIEW)



NC – No internal connection

TTL
MSI

TYPES SN54LS168, SN54LS169, SN54S168, SN54S169, SN74LS168, SN74LS169, SN74S168, SN74S169
SYNCHRONOUS 4-BIT UP/DOWN COUNTERS

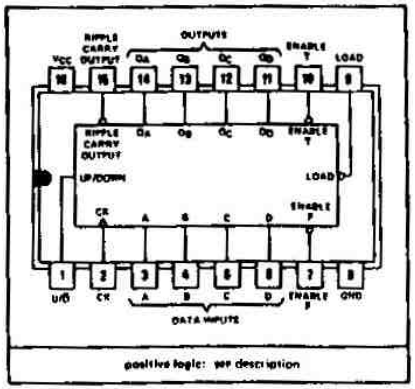
BULLETIN NO. DL-S 7412086, MARCH 1974

'LS168, 'S168 . . . SYNCHRONOUS UP/DOWN DECADE COUNTERS
'LS169, 'S169 . . . SYNCHRONOUS UP/DOWN BINARY COUNTERS

Programmable Look-Ahead Up/Down Binary/Decade Counters

- Fully Synchronous Operation for Counting and Programming
- Internal Look-Ahead for Fast Counting
- Carry Output for n-Bit Cascading
- Fully Independent Clock Circuit

SERIES SN54LS', SN54S' . . . J OR W PACKAGE
SERIES SN74LS', SN74S' . . . J OR N PACKAGE
(TOP VIEW)



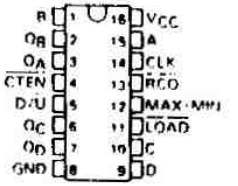
positive logic: see description

SN54190, SN54191, SN54LS190, SN54LS191, SN74190, SN74191, SN74LS190, SN74LS191
SYNCHRONOUS UP/DOWN COUNTERS WITH DOWN/UP MODE CONTROL

DECEMBER 1972 – REVISED MARCH 1988

- Counts 8-4-2-1 BCD or Binary
- Single Down/Up Count Control Line
- Count Enable Control Input
- Ripple Clock Output for Cascading
- Asynchronously Presettable with Load Control
- Parallel Outputs
- Cascadable for n-Bit Applications

SN54190, SN54191, SN54LS190, SN54LS191 . . . J PACKAGE
SN74190, SN74191 . . . N PACKAGE
SN74LS190, SN74LS191 . . . D OR N PACKAGE
(TOP VIEW)



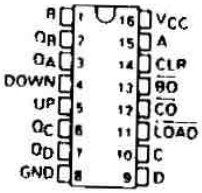
SN54192, SN54193, SN54LS192 SN54LS193, SN74192, SN74193, SN74LS192, SN74LS193
SYNCHRONOUS 4-BIT UP/DOWN COUNTERS (DUAL CLOCK WITH CLEAR)

DECEMBER 1972 – REVISED MARCH 1988

- Cascading Circuitry Provided Internally
- Synchronous Operation
- Individual Preset to Each Flip-Flop
- Fully Independent Clear Input

TYPES	TYPICAL MAXIMUM COUNT FREQUENCY	TYPICAL POWER DISSIPATION
'192, '193	32 MHz	325 mW
'LS192, 'LS193	32 MHz	95 mW

SN54192, SN54193, SN54LS192
SN54LS193 . . . J OR W PACKAGE
SN74192, SN74193 . . . N PACKAGE
SN74LS192, SN74LS193 . . . D OR N PACKAGE
(TOP VIEW)



III-3) Table de fonctionnement

CLEAR	S_1	S_0	C_k	SL	SR	D	C	B	A	Q_D	Q_C	Q_B	Q_A	Mode
0	X	X	X	X	X	X	X	X	X	0	0	0	0	Mise à zero
1	X	X	0	X	X	X	X	X	X	Q_D^0	Q_C^0	Q_B^0	Q_A^0	Mémorisation
1	1	1	↓	X	X	d	c	b	a	d	c	b	a	Chargement parallèle
1	0	1	↓	X	1	X	X	X	X	1	Q_D^{-1}	Q_C^{-1}	Q_B^{-1}	Décalage à droite avec des '1'
1	0	1	↓	X	0	X	X	X	X	0	Q_D^{-1}	Q_C^{-1}	Q_B^{-1}	Décalage à droite avec des '0'
1	1	0	↓	1	X	X	X	X	X	Q_C^{-1}	Q_B^{-1}	Q_A^{-1}	1	Décalage à gauche avec des '1'
1	1	0	↓	0	X	X	X	X	X	Q_C^{-1}	Q_B^{-1}	Q_A^{-1}	0	Décalage à gauche avec des '0'
1	0	0	X	X	X	X	X	X	X	Q_D^0	Q_C^0	Q_B^0	Q_A^0	Mémorisation

III-4) Exercice

Réalisez à l'aide du circuit 74LSl94 (registre à décalage) un circuit qui commande l'allumage de 4 lampes selon la séquence suivante:

	Lampe 1	Lampe 2	Lampe 3	Lampe 4
1^{er} cycle	1	0	0	0
	0	1	0	0
	0	0	1	0
	0	0	0	1
	0	0	1	0
	0	1	0	0
	1	0	0	0
	0	1	0	0
$2^{ème}$ cycle	etc ...			

La valeur 'l' signifie que la lampe est allumée.

TD N° 5 - Fonctionnement du HCS12

I. Codage de l'instruction LDD

Soit la ligne de programme suivante écrite à l'adresse $1000 :

LDD #$2000

Donnez le codage de cette ligne en utilisant les documents du constructeur (page suivante).
Expliquez l'exécution de l'instruction.

II. Transfert de données depuis la mémoire

Soit la ligne de programme suivante écrite à l'adresse $1000 :

LDD 4,X

où le registre 16 bits X contient la valeur $1100.

Codez cette ligne en utilisant les documents du constructeur (page suivante).
Expliquez l'exécution de l'instruction.

III. Branchement inconditionnel BRA

En utilisant la notice, coder la ligne de programme écrite en $1000 et décrire son exécution :

BRA $1000

IV. Temps d'excécution d'une boucle

On suppose que le registre D contient initialement la valeur $2000.
Calculez le temps d'exécution du programme suivant en prenant une horloge de 4 MHz :

```
BCL :   SUBD #1
        BNE BCL
```

Indication de temps de cycle : SUBD : 2 cycles ; BNE : 3 cycles si branchement, 1 cycle sinon

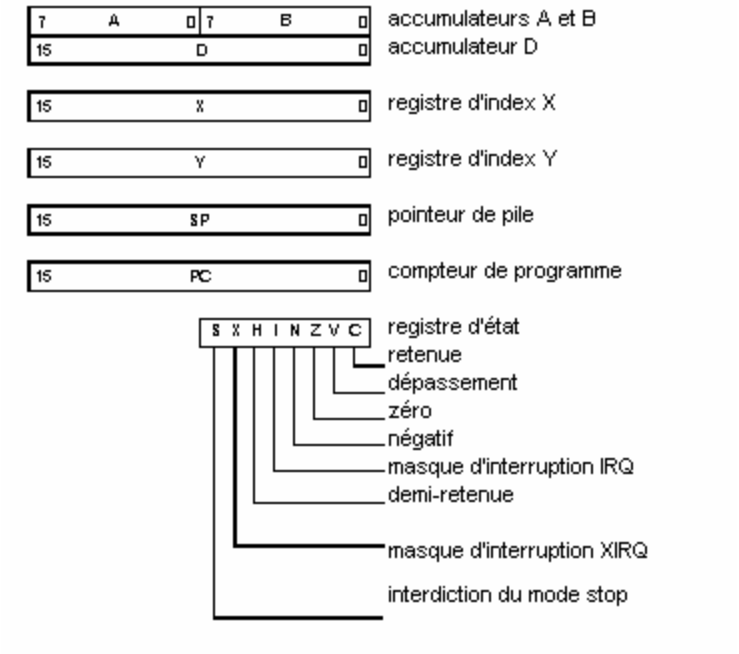


Fig.1 - Registres du microcontrôleur HC12

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail	S X H I	N Z V C
BNE rel8 1	Branch if Not Equal (if Z = 0)	REL	26 rr	PPP/P		
BRA rel8	Branch Always (if 1 = 1)	REL	20 rr	PPP		

LDD # opr16i
LDD opr8a
LDD opr16a
LDD oprx0_xysp
LDD oprx9,xysp
LDD oprx16,xysp
LDD [D, xysp]
LDD [oprx16,xysp]
	(M:M+1)⇒ A:B
	Load Double Accumulator D (A:B)
		IMM
		DIR
		EXT
		IDX
		IDX1
		IDX2
		[D,IDX]
		[IDX2]
			CC jj kk
			DC dd
			FC hh ll
			EC xb
			EC xb ff
			EC xb ee ff
			EC xb
			EC xb ee ff
				PO
				RPf
				RPO
				RPf
				RPO
				fRPP
				fIfRPf
				fIPRPf
						Δ Δ 0

SUBD # opr16i
SUBD opr8a
SUBD opr16a
SUBD oprx0_xysp
SUBD oprx9,xysp
SUBD oprx16,xysp
SUBD [D, xysp]
SUBD [oprx16,xysp]
	(D) – (M:M+1)⇒ D
	Subtract Memory from D (A:B)
		IMM
		DIR
		EXT
		IDX
		IDX1
		IDX2
		[D,IDX]
		[IDX2]
			83 jj kk
			93 dd
			B3 hh ll
			A3 xb
			A3 xb ff
			A3 xb ee ff
			A3 xb
			A3 xb ee ff
				PO
				RPf
				RPO
				RPf
				RPO
				fRPP
				fIfRPf
				fIPRPf
						Δ Δ Δ Δ

INH — Inherent; no operands in object code
IMM — Immediate; operand in object code
DIR — Direct; operand is the lower byte of an address from $0000 to $00FF
EXT — Operand is a 16-bit address
REL — Two's complement relative offset; for branch instructions
IDX — Indexed (no extension bytes); includes:
5-bit constant offset from X, Y, SP, or PC
Pre/post increment/decrement by 1 . . . 8
Accumulator A, B, or D offset
IDX1 — 9-bit signed offset from X, Y, SP, or PC; 1 extension byte
IDX2 — 16-bit signed offset from X, Y, SP, or PC; 2 extension bytes
[IDX2] — Indexed-indirect; 16-bit offset from X, Y, SP, or PC
[D, IDX] — Indexed-indirect; accumulator D offset from X, Y, SP, or PC

jj — High-order byte of a 16-bit immediate data value.
kk — Low-order byte of a 16-bit immediate data value.
rr — Signed relative offset $80 (–128) to $7F (+127). Offset relative to the byte following the relative offset byte, or low-order byte of a 16-bit relative offset for long branches.
xb — Indexed addressing post-byte

Access Detail
Each code letter except (,), and comma equals one CPU cycle. Uppercase = 16-bit operation and lowercase = 8-bit operation.
f —Free cycle, CPU doesn't use bus
R —16-bit data read
O —Optional program word fetch (P) if instruction is misaligned and has an odd number of bytes of object code — otherwise, appears as a free cycle (f); Page 2 prebyte treated as a separate 1-byte instruction
P —Program word fetch (always an aligned-word read)
PPP/P —Short branch, PPP if branch taken, P if not

Δ — Status bit affected by operation.

Fig.2 – Résumé du jeu d'instructions (Motorola reference manual S12CPU)

Table A-3. Indexed Addressing Mode Postbyte Encoding (xb)

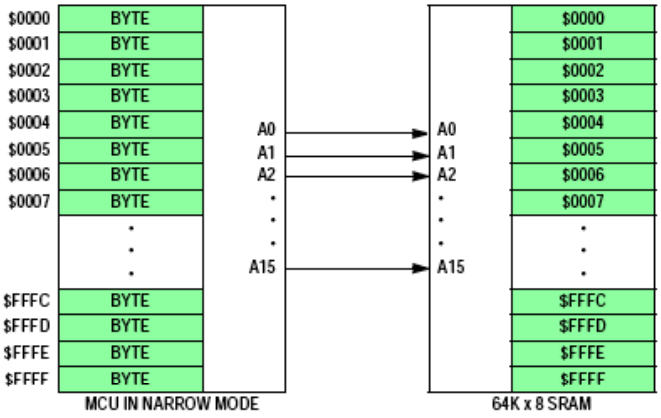
00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
0,X	−16,X	1,+X	1,X+	0,Y	−16,Y	1,+Y	1,Y+	0,SP	−16,SP	1,+SP	1,SP+	0,PC	−16,PC	n,X	n,SP
5b const	5b const	pre-inc	post-inc	5b const	5b const	pre-inc	post-inc	5b const	5b const	pre-inc	post-inc	5b const	5b const	9b const	9b const
01	11	21	31	41	51	61	71	81	91	A1	B1	C1	D1	E1	F1
1,X	−15,X	2,+X	2,X+	1,Y	−15,Y	2,+Y	2,Y+	1,SP	−15,SP	2,+SP	2,SP+	1,PC	−15,PC	−n,X	−n,SP
5b const	5b const	pre-inc	post-inc	5b const	5b const	pre-inc	post-inc	5b const	5b const	pre-inc	post-inc	5b const	5b const	9b const	9b const
02	12	22	32	42	52	62	72	82	92	A2	B2	C2	D2	E2	F2
2,X	−14,X	3,+X	3,X+	2,Y	−14,Y	3,+Y	3,Y+	2,SP	−14,SP	3,+SP	3,SP+	2,PC	−14,PC	n,X	n,SP
5b const	5b const	pre-inc	post-inc	5b const	5b const	pre-inc	post-inc	5b const	5b const	pre-inc	post-inc	5b const	5b const	16b const	16b const
03	13	23	33	43	53	63	73	83	93	A3	B3	C3	D3	E3	F3
3,X	−13,X	4,+X	4,X+	3,Y	−13,Y	4,+Y	4,Y+	3,SP	−13,SP	4,+SP	4,SP+	3,PC	−13,PC	[n,X]	[n,SP]
5b const	5b const	pre-inc	post-inc	5b const	5b const	pre-inc	post-inc	5b const	5b const	pre-inc	post-inc	5b const	5b const	16b indr	16b indr
04	14	24	34	44	54	64	74	84	94	A4	B4	C4	D4	E4	F4
4,X	−12,X	5,+X	5,X+	4,Y	−12,Y	5,+Y	5,Y+	4,SP	−12,SP	5,+SP	5,SP+	4,PC	−12,PC	A,X	A,SP
5b const	5b const	pre-inc	post-inc	5b const	5b const	pre-inc	post-inc	5b const	5b const	pre-inc	post-inc	5b const	5b const	A offset	A offset
05	15	25	35	45	55	65	75	85	95	A5	B5	C5	D5	E5	F5
5,X	−11,X	6,+X	6,X+	5,Y	−11,Y	6,+Y	6,Y+	5,SP	−11,SP	6,+SP	6,SP+	5,PC	−11,PC	B,X	B,SP
5b const	5b const	pre-inc	post-inc	5b const	5b const	pre-inc	post-inc	5b const	5b const	pre-inc	post-inc	5b const	5b const	B offset	B offset
06	16	26	36	46	56	66	76	86	96	A6	B6	C6	D6	E6	F6
6,X	−10,X	7,+X	7,X+	6,Y	−10,Y	7,+Y	7,Y+	6,SP	−10,SP	7,+SP	7,SP+	6,PC	−10,PC	D,X	D,SP
5b const	5b const	pre-inc	post-inc	5b const	5b const	pre-inc	post-inc	5b const	5b const	pre-inc	post-inc	5b const	5b const	D offset	D offset
07	17	27	37	47	57	67	77	87	97	A7	B7	C7	D7	E7	F7
7,X	−9,X	8,+X	8,X+	7,Y	−9,Y	8,+Y	8,Y+	7,SP	−9,SP	8,+SP	8,SP+	7,PC	−9,PC	[D,X]	[D,SP]
5b const	5b const	pre-inc	post-inc	5b const	5b const	pre-inc	post-inc	5b const	5b const	pre-inc	post-inc	5b const	5b const	D indirect	D indirect
08	18	28	38	48	58	68	78	88	98	A8	B8	C8	D8	E8	F8
8,X	−8,X	8,−X	8,X−	8,Y	−8,Y	8,−Y	8,Y−	8,SP	−8,SP	8,−SP	8,SP−	8,PC	−8,PC	n,Y	n,PC
5b const	5b const	pre-dec	post-dec	5b const	5b const	pre-dec	post-dec	5b const	5b const	pre-dec	post-dec	5b const	5b const	9b const	9b const
09	19	29	39	49	59	69	79	89	99	A9	B9	C9	D9	E9	F9
9,X	−7,X	7,−X	7,X−	9,Y	−7,Y	7,−Y	7,Y−	9,SP	−7,SP	7,−SP	7,SP−	9,PC	−7,PC	−n,Y	−n,PC
5b const	5b const	pre-dec	post-dec	5b const	5b const	pre-dec	post-dec	5b const	5b const	pre-dec	post-dec	5b const	5b const	9b const	9b const
0A	1A	2A	3A	4A	5A	6A	7A	8A	9A	AA	BA	CA	DA	EA	FA
10,X	−6,X	6,−X	6,X−	10,Y	−6,Y	6,−Y	6,Y−	10,SP	−6,SP	6,−SP	6,SP−	10,PC	−6,PC	n,Y	n,PC
5b const	5b const	pre-dec	post-dec	5b const	5b const	pre-dec	post-dec	5b const	5b const	pre-dec	post-dec	5b const	5b const	16b const	16b const
0B	1B	2B	3B	4B	5B	6B	7B	8B	9B	AB	BB	CB	DB	EB	FB
11,X	−5,X	5,−X	5,X−	11,Y	−5,Y	5,−Y	5,Y−	11,SP	−5,SP	5,−SP	5,SP−	11,PC	−5,PC	[n,Y]	[n,PC]
5b const	5b const	pre-dec	post-dec	5b const	5b const	pre-dec	post-dec	5b const	5b const	pre-dec	post-dec	5b const	5b const	16b indr	16b indr
0C	1C	2C	3C	4C	5C	6C	7C	8C	9C	AC	BC	CC	DC	EC	FC
12,X	−4,X	4,−X	4,X−	12,Y	−4,Y	4,−Y	4,Y−	12,SP	−4,SP	4,−SP	4,SP−	12,PC	−4,PC	A,Y	A,PC
5b const	5b const	pre-dec	post-dec	5b const	5b const	pre-dec	post-dec	5b const	5b const	pre-dec	post-dec	5b const	5b const	A offset	A offset
0D	1D	2D	3D	4D	5D	6D	7D	8D	9D	AD	BD	CD	DD	ED	FD
13,X	−3,X	3,−X	3,X−	13,Y	−3,Y	3,−Y	3,Y−	13,SP	−3,SP	3,−SP	3,SP−	13,PC	−3,PC	B,Y	B,PC
5b const	5b const	pre-dec	post-dec	5b const	5b const	pre-dec	post-dec	5b const	5b const	pre-dec	post-dec	5b const	5b const	B offset	B offset
0E	1E	2E	3E	4E	5E	6E	7E	8E	9E	AE	BE	CE	DE	EE	FE
14,X	−2,X	2,−X	2,X−	14,Y	−2,Y	2,−Y	2,Y−	14,SP	−2,SP	2,−SP	2,SP−	14,PC	−2,PC	D,Y	D,PC
5b const	5b const	pre-dec	post-dec	5b const	5b const	pre-dec	post-dec	5b const	5b const	pre-dec	post-dec	5b const	5b const	D offset	D offset
0F	1F	2F	3F	4F	5F	6F	7F	8F	9F	AF	BF	CF	DF	EF	FF
15,X	−1,X	1,−X	1,X−	15,Y	−1,Y	1,−Y	1,Y−	15,SP	−1,SP	1,−SP	1,SP−	15,PC	−1,PC	[D,Y]	[D,PC]
5b const	5b const	pre-dec	post-dec	5b const	5b const	pre-dec	post-dec	5b const	5b const	pre-dec	post-dec	5b const	5b const	D indirect	D indirect

Key to Table A-3

TD N° 6 – Association de boîtiers et décodage d'adresse avec HCS12

Un système de gestion de données est basé sur un microcontrôleur HCS12 configuré en mode étendu court (narrow mode with one byte-wide memory) : le bus d'adresse est de 16 bits (A0 ... A15), et le bus de données de 8 bits (D0 ... D7). On dispose d'une ligne R/$\overline{W}$ pour la lecture et l'écriture.

Les connexions entre le HCS12 et la mémoire se font comme indiqué dans la figure ci-dessous :



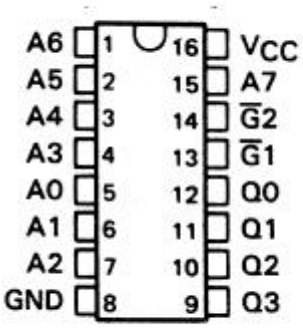
Address Connection for MCU in Narrow Mode to One Byte-Wide Memory

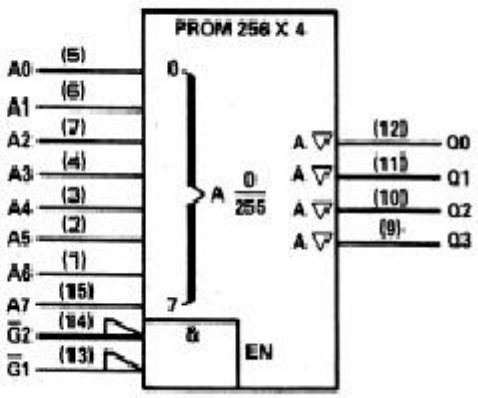
I. Association de boîtiers

On veut utiliser les boîtiers mémoire TBP24S10 et CY7C185 ci-dessous, pour raccorder à ce système les zones de mémoire suivantes :

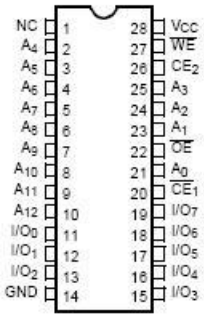
- ✔ 256 mots de PROM
- ✔ 16 k mots de SRAM

TBP24S10





CY7C185



Pin Name	Pin Function
A_0-A_{12}	Address Inputs
WE	Write Enable
CS_1, CS_2	Chip Select
OE	Output Enable
I/O_1-I/O_8	Data Inputs/Outputs
V_{CC}	Power (+ 5V)
V_{SS}	Ground
N.C.	No Connection

Truth Table

CE_1	CE_2	WE	OE	Input/Output	Mode
H	X	X	X	High Z	Deselect/Power-Down
X	L	X	X	High Z	Deselect/Power-Down
L	H	H	L	Data Out	Read
L	H	L	X	Data In	Write
L	H	H	H	High Z	Deselect

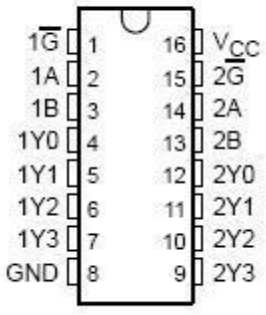
Donner les caractéristiques (type de mémoire, taille) de chacun des boîtiers.
Combien et comment doit-on raccorder les boîtiers au système ?

II. Map mémoire

On désire que ces 2 zones PROM et SRAM soient situées respectivement à l'adresse la plus basse et à l'adresse $8000.

a) Donnez les adresses logiques de début et de fin de chacune de ces zones mémoire.
b) Donnez le schéma de sélection de chaque zone en utilisant le bus d'adresse complet.
c) Donnez un décodage minimal pour faire la sélection de ces zones.
d) Utilisez des décodeurs 2x4 (74LS139).

74ALS139



FUNCTION TABLE

INPUTS			OUTPUTS			
ENABLE $\overline{G}$	SELECT B	SELECT A	Y0	Y1	Y2	Y3
H	X	X	H	H	H	H
L	L	L	L	H	H	H
L	L	H	H	L	H	H
L	H	L	H	H	L	H
L	H	H	H	H	H	L

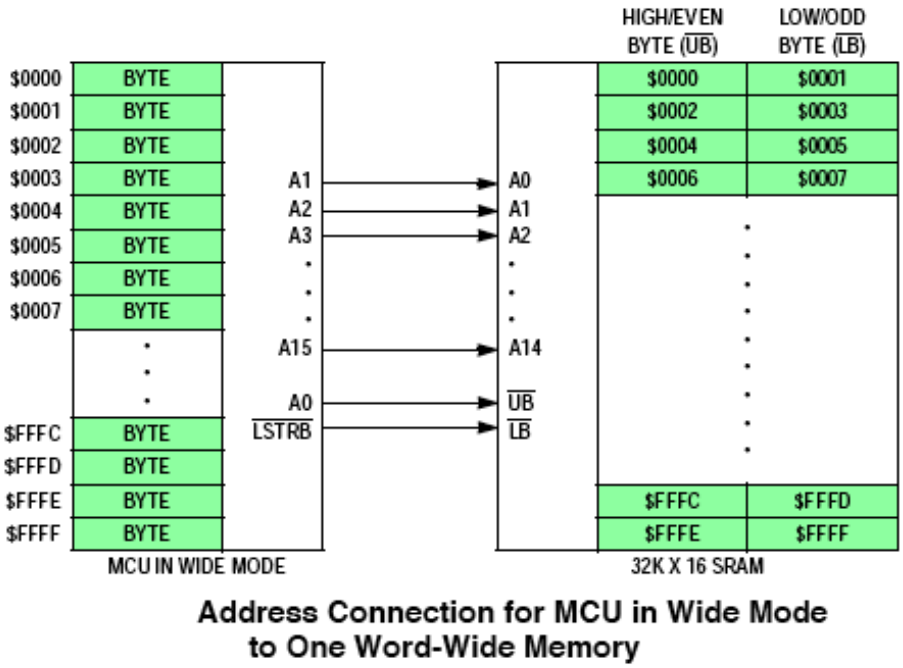
Précisez dans ce cas combien d'adresses logiques correspondent à une adresse physique (1 case de chaque zone).

TD N° 6 bis – Association de boîtiers et décodage d'adresse avec HCS12

On réutilise le système du TD précédent avec un HCS12 configuré en mode étendu large (*wide mode with word-wide memory*) : le bus de données de 16 bits (ports A et B). On dispose de :

- d'une ligne R/$\overline{W}$ pour choisir la lecture ou l'écriture,
- d'une ligne $\overline{LSTRB}$ (active à 0) pour choisir les adresses impaires,
- et de l'adresse A0 (active à 0) pour choisir les adresses paires.

Les connexions entre le HCS12 et la mémoire se font suivant le schéma ci-dessous :



Address Connection for MCU in Wide Mode to One Word-Wide Memory

I. Association de boîtiers

On veut raccorder à ce système des boîtiers mémoire PROM 256 x 4 bits (TBP24S10) et SRAM 8k x 8 bits (CY7C185) pour constituer les zones de mémoire suivantes :

- ✔ 256 mots de PROM
- ✔ 16 k mots de SRAM

Combien et comment doit-on raccorder de boîtiers au système ?

II. Map mémoire

On désire que ces 2 zones PROM et SRAM soient situées respectivement à l'adresse la plus basse et à l'adresse $8000.

a) Donnez les adresses logiques de début et de fin de chacune de ces zones mémoire.

b) Donnez le schéma de sélection de chaque zone en utilisant le bus d'adresse complet.

c) Donnez un décodage minimal pour faire la sélection de ces zones.

d) Utilisez des décodeurs 2x4 (74LS139).

TD N° 7 - Programmation simple

I. Addition 16 bits

Additionnez le contenu d'une variable exprimée sur 16 bits (stockée en mémoire à l'adresse symbolique VALEUR1 et à l'adresse physique $2000) au contenu d'une variable exprimée sur 16 bits (d'adresse symbolique VALEUR2 et d'adresse physique $2002). Placez le résultat dans la variable 16 bits RESULTAT située à l'emplacement $2004.

Exemple d'exécution :

	adresse symbolique	contenu mémoire mots de 8 bits	adresse physique hexadécimale
Entrée	VALEUR1	10	2000
		F5	2001
	VALEUR2	26	2002
		21	2003
Sortie	RESULTAT	37	2004
		16	2005

II. Addition 32 bits

Additionnez les contenus de deux variables de 32 bits d'adresses symboliques VALEUR1 (adresse physique 2000) et VALEUR2 (adresse physique $2004). Rangez le résultat dans RESULTAT (adresse physique $2008).

Exemple d'exécution :

	adresse symbolique	contenu mémoire mots de 16 bits	adresse physique hexadécimale
Entrée	VALEUR1	F210	2000
		0123	2002
	VALEUR2	4002	2004
		3F51	2006
Sortie	RESULTAT	3212	2008
		4074	200A

L'opération réalisée sera :

```
   F2100123
+  40023F51
------------
(1) 32124074
```

III. Factorielle

Recherchez la factorielle d'une variable 8 bits d'adresse symbolique VALEUR d'adresse physique $2010 en utilisant une table de factorielles déjà calculées. Cette table (d'adresse symbolique FTABLE et d'adresse physique $2000) comporte les factorielles de 0, 1 …… 7. Rangez le résultat dans une variable RESULTAT d'adresse $2012. On suppose que VALEUR est comprise entre 0 et 7.

Exemple d'exécution :

	adresse symbolique	contenu mémoire mots 16 bits	adresse physique hexadécimale	donnée
Entrée	FTABLE	0001	2000	$0! = (1)_{10}$
		0001	2002	$1! = (1)_{10}$
		0002	2004	$2! = (2)_{10}$
		0006	2006	$3! = (6)_{10}$
		0018	2008	$4! = (24)_{10}$
		0078	200A	$5! = (120)_{10} = (78)_{16}$
		02D0	200C	$6! = (720)_{10}$
		13B0	200E	$7! = (5040)_{10}$
	VALEUR	0500	2010	
Sortie	RESULTAT	0078	2012	

IV. Maximum de deux valeurs 16 bits

Trouvez la plus grande des deux valeurs exprimées sur 16 bits : VALEUR1 d'adresse $2000 et VALEUR2 d'adresse $2002. Placez le résultat dans la variable RESULTAT d'adresse $2004. On supposera que les valeurs sont non signées.

Exemple d'exécution :

	adresse symbolique	contenu mémoire sur 2 octets ou 16 bits	adresse physique hexadécimale
Entrée	VALEUR1	1234	2000
	VALEUR2	4321	2002
Sortie	RESULTAT	4321	2004

TD N° 8 - Programmes

I. Somme de nombres 8 bits

Calculez la somme d'une suite de nombres non signés de taille 8 bits. La longueur (en octet) de la suite est définie par la variable d'adresse symbolique LONGUEUR et d'adresse physique $2000. L'adresse de début de la suite est contenue dans la variable mot d'adresse 16 bits symbolique DEBUT et d'adresse physique $2001. Prenez en compte les retenues et exprimez le résultat sur 16 bits. Le résultat sera stocké dans une variable d'adresse symbolique TOTAL et d'adresse physique $2002.

Exemple d'exécution :

	adresse symbolique	contenu mémoire	adresse physique
Entrée	LONGUEUR	03	2000
	DEBUT	30	2001
		00	2002
Sortie	TOTAL	00	2003
		A4	2004

On suppose qu'aux adresses $3000, $3001 et $3002 on a les valeurs hexadécimales 40, 22 et 42.

II. Conversion de code ASCII vers décimal

Convertissez le caractère ASCII contenu dans la variable CAR, d'adresse physique $2000, en un chiffre décimal. Rangez le résultat dans la variable CHIFFRE d'adresse physique $2001. Si le contenu de CAR n'est pas la représentation ASCII d'un chiffre décimal, mettez le contenu de CHIFFRE à $FF.

2 exemples d'exécution :

a)	adresse symbolique	contenu mémoire	adresse physique
Entrée	CAR	37	2000
Sortie	CHIFFRE	07	2001

b)	adresse symbolique	contenu mémoire	adresse physique
Entrée	CAR	SS	2000
Sortie	CHIFFRE	FF	2001

Code ASCII hexadécimal	Chiffre décimal
30	0
31	1
.	.
.	.
.	.
38	8
39	9

III. Tri par bulle décroissant

Triez dans l'ordre décroissant une liste de nombres binaires 8 bits non signés. L'adresse 16 bits de début de la liste se trouve dans la variable LIST d'adresse physique $2000. Le premier élément de la liste est le nombre d'éléments contenus dans celle-ci. Ainsi la liste aura au maximum 255 éléments.

Exemple :

1) Avant exécution :

adresse physique hexadécimale	contenu de la mémoire		adresse symbolique	contenu de la mémoire	adresse physique hexadécimale
3000	06	Nbre d'éléments de la liste	LIST	30	2000
3001	2A	1er élément de la liste		00	2001
3002	B5				
3003	60				
3004	3F				
3005	D1				
3006	19				

2) Après exécution du programme de tri :

adresse physique hexadécimale	contenu de la mémoire		adresse symbolique	contenu de la mémoire	adresse physique hexadécimale
3000	06		LIST	30	2000
3001	D1	+ grand élément de la liste		00	2001
3002	B5				
3003	60				
3004	3F				
3005	2A				
3006	19				

TD N° 9 – Sous-programmes

I. Conversion d'un chiffre hexadécimal en ASCII

Table de conversion d'une valeur hexadécimale sur 1 position (0, 1, 2, …, A, B, C, D, E, F) en son équivalent en code ASCII.

hexadécimal	0	1	2	…	A	…	F
Code ASCII	00110000	00110001	00110010	…	01000001	…	01000110

La valeur a pour adresse symbolique CHIFFRE et pour adresse physique $2000. Le code ASCII généré aura pour adresse symbolique CARA et pour adresse physique $2001. On réalisera un sous-programme dont le seul paramètre, qui servira d'entrée et de sortie, sera le registre B.

Exemples d'exécution :

	adresse symbolique	contenu mémoire sur un octet	adresse physique hexadécimale
Entrée	CHIFFRE	0C	2000
Sortie	CARA	43	2001
Entrée	CHIFFRE	06	2000
Sortie	CARA	36	2001

II. Conversion d'un nombre hexadécimal 4 chiffres en ASCII

Convertissez une valeur hexadécimale de 4 positions (contenue dans la variable NOMBRE, d'adresse physique 2000), en quatre codes ASCII. Ces codes ASCII seront rangés dans le tableau CHAINE d'adresse physique 2002. Réaliser cette tâche en utilisant un sous-programme ayant deux paramètres d'entrée : la variable NOMBRE et l'adresse CHAINE du tableau de résultats. On utilisera la pile pour passer ces deux arguments.

Exemple d'exécution :

adresse symbolique	contenu mémoire sur un octet	adresse physique hexadécimale	
NOMBRE	4C	2000	
	D0	2001	
CHAINE	34	2002	code de 4
	43	2003	code du C
	44	2004	code du D
	30	2005	code du 0

TD N° 10 – Interruption

Une entrée d'immeuble dispose d'un système de sécurité qui n'ouvre la porte du bâtiment qu'à la reconnaissance d'un code connu, exprimé sur 4 positions hexadécimales. Sachant qu'à la demande d'ouverture de la porte, c'est-à-dire lorsque l'on introduit un code, le périphérique dit « PORTE » envoie une interruption sur la ligne $\overline{\text{IRQ}}$ (entrée PTE1 du HC12 mise à zéro) ; il écrit aussi le code d'entrée dans la partie basse d'un mot 32 bits d'adresse symbolique PAAR.

I. Programmes

Écrivez :

1) la séquence de traitement d'interruption d'adresse symbolique PTISECU qui devra :

- lire le code dans les 16 bits de poids faible de PAAR

- le comparer avec une table de codes connus, table stockée à l'adresse symbolique CONNUS, avec comme première information le nombre connus de codes (16 bits). Ceci sera fait à l'aide d'un sous-programme qui restituera son résultat dans le bit de poids faible du registre B (1 code autorisé, 0 code non autorisé)

- provoquer la commande d'ouverture de la porte, en mettant à 1 le bit de poids fort de PAAR

- retourner vers la tâche de fond (programme principal, séquence d'appel).

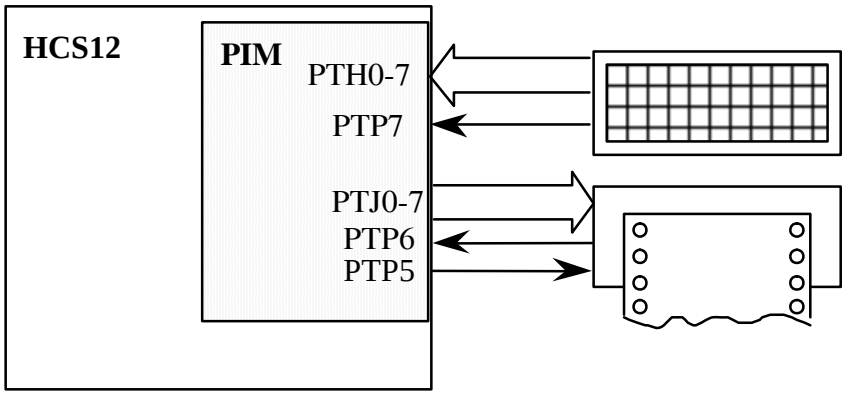
2) le programme principal avec :
- initialisation du registre de contrôle de $\overline{\text{IRQ}}$ (adresse $1E)
- autorisation des interruptions et chargement du vecteur d'interruption (adresse $FFF2)
- tâche de fond qui sera constituée d'une boucle d'attente de type BOUCLE: JMP BOUCLE

II. Autres solutions

Que doit-on modifier si on prend la ligne $\overline{\text{XIRQ}}$ au lieu de la ligne $\overline{\text{IRQ}}$?
Pourrait-on résoudre le problème sans utiliser d'interruption ?

TD N° 11 et 12 – Entrées-sorties

I. Système

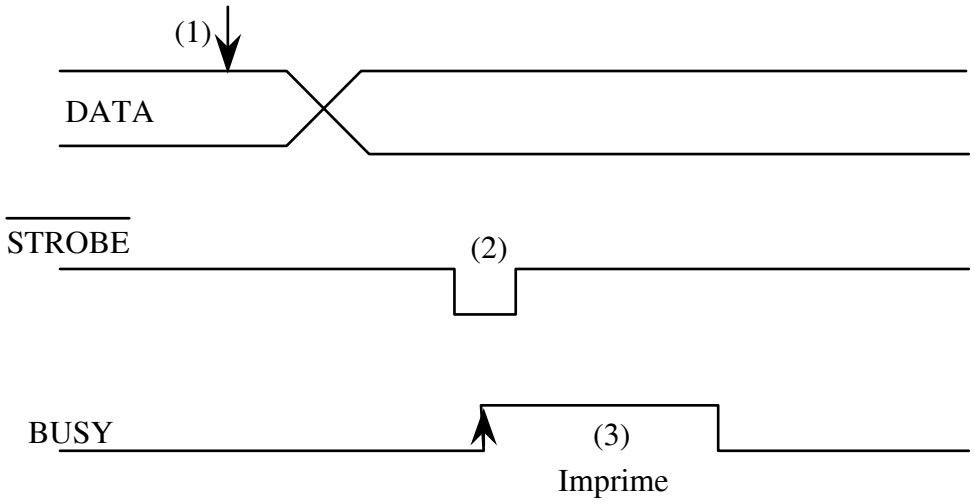


Les lignes non utilisées du port P sont à relier avec des résistances internes d'abaissement du HCS12.

Fonctionnement du clavier

L'appui sur une touche génère un front descendant sur l'entrée 7 du port P.
Le positionnement sur le port H du code 8 bits ASCII correspond à la touche appuyée.
Dans ce cas là, on veut que le HCS12 soit alors interrompu et lise la valeur présente sur le port H. Le vecteur d'interruption du port P (adresse symbolique Vportp) est situé à l'adresse $FF8E.

Fonctionnement de l'imprimante



Processus d'impression d'un caractère

(1) Le microcontrôleur positionne le code ASCII du caractère sur le Port J.

(2) Il signale que la donnée est prête en générant une impulsion (active à 0) sur la ligne $\overline{\text{STROBE}}$ (sortie PTP5 du port P).

(3) L'imprimante détecte cette impulsion et positionne sa ligne BUSY (entrée PTP6) à 1 pendant la durée de l'impression, puis se remet en état de disponibilité BUSY = 0.

QUESTION 1 :

Écrivez le programme qui va permettre la réception d'une chaîne de 16 caractères entrés à partir du clavier et leur stockage dans un tableau d'adresse $1200.

Adresses paires	Contenu 8 bits	Contenu 8 bits	Adresses impaires
	20	21	$120F
	39	32	
	45	20	
	4E	45	
	41	4E	
	45	20	
	4E	4E	
$1200	42	4F	

Pour cela, on définira les contenus des différents registres de l'interface, puis le programme assembleur correspondant.

QUESTION 2 :

Modifiez le programme précédent pour qu'après la réception des 16 caractères clavier terminée, le microcontrôleur les imprime l'un après l'autre sur l'imprimante.