

**J.L.S. INFORMATIQUE**

2, rue Clément ADER

B.P. 50065

57972 YUTZ CEDEX

Tel : 33 (0)3 82 86 00 16

Fax : 33 (0)3 82 86 00 12

URL : [www.jls-info.com](http://www.jls-info.com)

**MCUHC11**

**Carte universelle à MC68HC11**



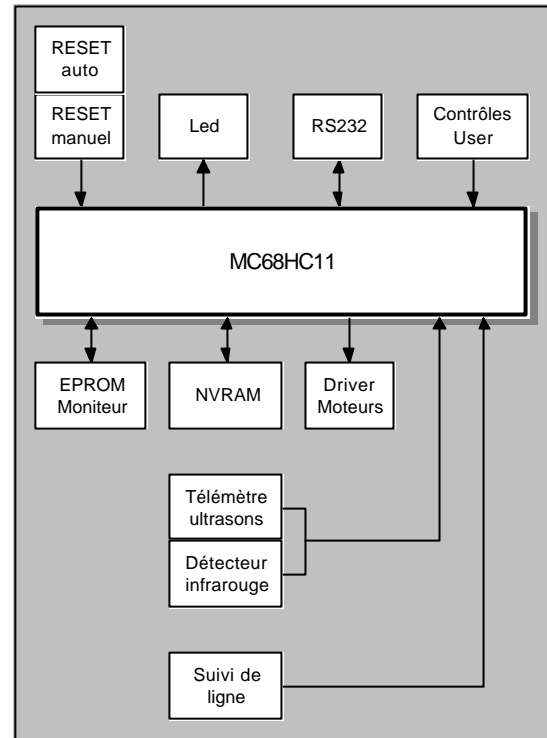
## Caractéristiques générales

La carte MCUHC11 est dédiée principalement à des applications de robotique mobile. Basée sur un microcontrôleur 8 bits MC68HC11, elle se caractérise par de nombreux avantages :

- Microcontrôleur MC68HC11 à 7.3728MHz
- EPROM 8Ko, moniteur embarqué
- NVRAM 32Ko
- 256o EEPROM interne (en fonction du 68HC11)
- UART RS232
- Double driver moteurs
- Servomoteur pour tête de détection
- Leds de contrôle
- *User switch*
- *User Push Button*
- Dimensions 100mm x 160mm
- Fonctionnement de 0°C à 60°C
- Alimentation +12V

De plus, elle dispose de deux cartes additionnelles dédiées :

- Télémètre à ultrasons de 15cm à 85cm
- Détecteur infrarouge de 2 cm à 1m
- 
- Capteurs infrarouges pour suivi de ligne au sol



# Memory Map MCUHC11

Pages 4096 o	Memory Map MC68HC11A1 interne	Memory Map Composants Externes
0000	0000 - 00FF : 256 o RAM interne <i>Partiellement réservé moniteur</i>	
0FFF		
1000	1000 - 103F : I / O	
1FFF		
2000		2000 - 9FFF : 32 Ko NVRAM
2FFF		
3000		
3FFF		
4000		
4FFF		
5000		
5FFF		
6000		
6FFF		
7000		
7FFF		
8000		
8FFF		
9000		
9FFF		
A000		\$A000 : Leds
AFFF		
B000	B600 - B7FF : 512 o EEPROM	
BFFF		
C000		
CFFF		
D000		
DFFF		
E000		E000 - FFFF : 8 Ko EPROM moniteur
EFFF		
F000		
FFFF	FFD6 - FFFF : vecteurs	

## Connexion des entrées – sorties du MC68CH11

Nom	Description	Type naturel	Type MCUHC11	Périphérique
/XIRQ	(pull-up 4.7K)	I	I	extension
/IRQ	(pull-up 4.7K)	I	I	extension
MODA	A = B = 1 : mode <i>extended</i>	I/O	I/O	NA
MODB	(pull-up 4.7K)	I	I	NA
/RESET	(pull-up 4.7K)	I/O	I/O	extension
VRL	Réf. Basse CAN	I	I	0V
VRH	Réf. Haute CAN	I	I	+5V
STRA	AS	I/O	I/O	NA
STRB	R/W	0	0	NA
XTAL	Sortie Osc. Int.	Q	Q	NA
EXTAL	Entrée Osc. Int.	Q	Q	NA
E	Sortie Horloge Bus	0	0	NA
PA0	IC3	I	I	Détection US
PA1	IC2	I	I	extension
PA2	IC1	I	I	extension
PA3	OC5	0	0	IN1 Moteur Gauche
PA4	OC4	0	0	IN2 Moteur Gauche
PA5	OC3	0	0	IN3 Moteur Droit
PA6	OC2	0	0	IN4 Moteur Droit
PA7	PAI	I/O	I/O	extension
PB0 – PB 7	A8 – A15	0	0	NA
PC0 – PC7	AD0 – AD7	I/O	I/O	NA
PD0	RxD	I/O	-	RS232
PD1	TxD	I/O	-	RS232
PD2	MISO	I/O	I	USR SW (User switch)
PD3	MOSI	I/O	I	USR PB (User Push Button)
PD4	SCK	I/O	0	Servomoteur
PD5	/SS	I/O	0	Détection IR
PE0	AIN0	I	I	Détection IR Sol 1
PE1	AIN1	I	I	Détection IR Sol 2
PE2	AIN2	I	I	Détection IR Sol 3
PE3	AIN3	I	I	Détection IR Sol 4
PE4	AIN4	I	I	extension
PE5	AIN5	I	I	extension
PE6	AIN6	I	I	extension
PE7	AIN7	I	I	extension

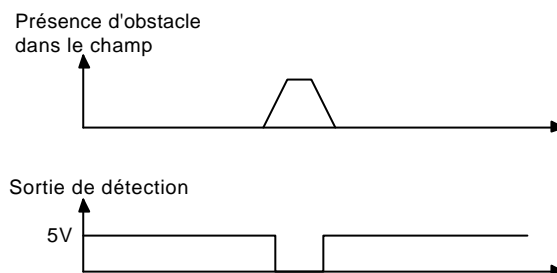
# Carte de détection US + IR, connecteur JP1

La carte additionnelle ultrason et infrarouge est reliée à la carte principale par une nappe 4 fils. L'alimentation +5V est fournie par celle – ci.

VCC	1
US	2
IR	3
GND	4

## ▪ Détection infrarouge.

Tant que l'obstacle est présent dans le champ de détection, la sortie est active (état bas).



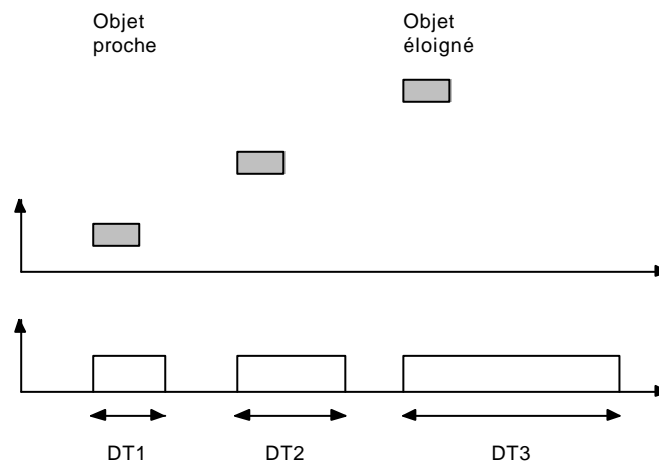
## ▪ Détection ultrason.

La durée à l'état haut de la sortie est proportionnelle à l'éloignement de l'objet dans le champ de détection.

La distance de détection est comprise entre 15cm et 85cm.

La distance se calcule suivant la formule suivante :

$$D(m) = \frac{V(340ms^{-1}) * DT(s)}{2}$$



## Carte de suivi de ligne, connecteur JP2

La carte additionnelle de suivi de ligne est reliée à la carte principale par une nappe 10 fils avec connecteurs à sertir. CIR0 à CIR3 fournissent le niveau de réception infrarouge des opto - coupleurs de réflexion.

VCC	1	2	VCC
CIR0	3	4	CIR1
CIR2	5	6	CIR3
GND	7	8	GND
GND	9	10	GND

## Connecteur d'alimentation JP4

+12V	1
+12V	2
0V	3
0V	4

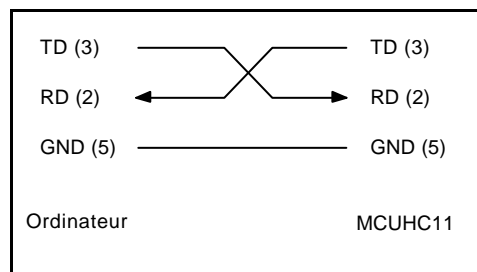
## Connecteur RS232 JP5

La liaison RS232 autorise deux utilisations :

- une liaison vers PC hôte pour téléchargement de code dans la NVRAM par le moniteur intégré
- une liaison vers un périphérique quelconque

Le protocole de transmission mis en place par défaut dans le moniteur est 38400 bauds, 1 stop, pas de parité. Bien entendu, la configuration de la SCI peut être modifiée par le code utilisateur.

GND	5
TD	3
RD	2



## Connecteur de détection US + IR JP6

La carte additionnelle est reliée à ce connecteur par une nappe 4 fils. PA0 et PD5 seront configurés en entrée.

VCC	1
PA0	2
PD5	3
GND	4

## Connecteur de suivi de ligne JP7

La carte additionnelle de suivi de ligne sera reliée à ce connecteur par une nappe 10 fils avec connecteurs à sertir. PE0 à PE3 seront activées comme entrées analogiques, pour le convertisseur interne.

VCC	1	2	VCC
PE0	3	4	PE1
PE2	5	6	PE3
GND	7	8	GND
GND	9	10	GND

## Connecteur d'extension libre JP8

VCC	1	2	VCC
/XIRQ	3	4	PA1
/RESET	5	6	PA2
/IRQ	7	8	PA7
GND	9	10	GND

## Connecteur du servomoteur JP9

Le servomoteur sera relié directement à ce connecteur par son câble. PD4 sera configuré en sortie.

PD4	1
VCC	2
GND	3

## Connecteur d'extension libre JP10

VCC	1	2	VCC
VCC	3	4	PE4
PE5	5	6	PE6
PE7	7	8	GND
GND	9	10	GND

## Connecteur moteur n°1 JP11

VMot	1
M1-	2
GND	3
M1+	4

Arbitrairement ce connecteur est dédié au moteur gauche. Les ports PA3 et PA4 (configurés en sortie) pilotent le pont en H du driver.

PA4	PA3	M1-	M1+
0	0	0V	0V
0	1	0V	VMot
1	0	VMot	0V
1	1	VMot	VMot

## Connecteur moteur n°2 JP12

VMot	1
M2-	2
GND	3
M2+	4

Arbitrairement ce connecteur est dédié au moteur gauche. Les ports PA5 et PA6 (configurés en sortie) pilotent le pont en H du driver.

PA6	PA5	M2-	M2+
0	0	0V	0V
0	1	0V	VMot
1	0	VMot	0V
1	1	VMot	VMot

# Moniteur de téléchargement embarqué

Le moniteur Y11 a été développé pour télécharger rapidement et simplement un programme dans la NVRAM de la carte. Ses fonctions sont donc restreintes :

- effacement NVRAM
- dump mémoire de \$0000 à \$FFFF
- lancement manuel du code utilisateur \$2040
- aide sur les commandes
- chargement code utilisateur
- lancement automatique du code utilisateur \$2040 en fonction de USR SW.

La liaison entre la carte MCUHC11 et un ordinateur de développement s'effectue par un port série de type RS232. Les paramètres de communication sont : 9600 bauds, pas de parité, 8 bits de données, 1 bit de stop. Le fichier utilisateur à transférer est le fichier compilé, avec l'extension S19 ou S1.

Le code utilisateur doit commencer à l'adresse \$2040, à laquelle branche le moniteur de manière automatique ou en mode manuel (en fonction de l'état du switch USR SW).

Remarque : les commandes sont traitées en minuscule.

## Invite du moniteur :

```
Y11 : Bootloader V1.1
(c) JLS Informatique Y.L. 2002

Y11 commands
c : clear NVRAM
d : dump
g : go $2040
h : help
l : load .S19 file

>
```

Au démarrage, après affichage des informations commerciales, le moniteur passe en mode MONITEUR si le switch est placé dans cette position. Dans le cas contraire, le moniteur passe en mode USER et branche directement en \$2040 pour exécuter le code utilisateur.

## Commande "Clear NVRAM" : c

Effacement de la mémoire NVRAM, remplissage de \$2000 à \$9FFF par \$FF.

```
>c
Clear NVRAM

>
```



# Memory Map et pseudos-vecteurs

Memory-map général:

Adresse(s)	Type Mémoire	Description
\$0000 - \$00FF	256o Ram interne	Moniteur Y11 : Variables + Pile + pseudo-vecteur SCI USER : Pile + pseudo-vecteur SCI
\$1000 - \$103F	Registres internes	Registres I/O
\$2000 - \$203F	NVRAM externe	Pseudos-vecteurs
\$2040 - \$9FFF	NVRAM externe	Code utilisateur
\$A000	Latch	Leds
\$B600 - \$B7FF	EEPROM interne	Libre USER
\$E000 - \$FFFF	EPROM externe	Moniteur Y11

Memory-map détaillé, RAM interne :

Adresse(s)	Description
\$0000 - \$0007	Réservé moniteur Y11
\$0008	JMP opcode
\$0009 - \$000A	Adresse SCI_ISR (moniteur Y11 / USER)
< \$00FF	Pile moniteur Y11 / USER

Memory-map détaillé, NVRAM externe :

Adresse(s)	Description
\$2000	JMP opcode + SPI_ISR
\$2003	JMP opcode + ACC_IN_ISR
\$2006	JMP opcode + ACC_OV_ISR
\$2009	JMP opcode + TIM_OV_ISR
\$200C	JMP opcode + TOC5_ISR
\$200F	JMP opcode + TOC4_ISR
\$2012	JMP opcode + TOC3_ISR
\$2015	JMP opcode + TOC2_ISR
\$2018	JMP opcode + TOC1_ISR
\$201B	JMP opcode + TIC3_ISR
\$201 <sup>E</sup>	JMP opcode + TIC2_ISR
\$2021	JMP opcode + TIC1_ISR
\$2024	JMP opcode + RTI_ISR
\$2027	JMP opcode + IRQ_ISR
\$202A	JMP opcode + XIRQ_ISR
\$202D	JMP opcode + SWI_ISR
\$2030	JMP opcode + ILL_ISR
\$2033	JMP opcode + COP_ISR
\$2036	JMP opcode + CLK_ISR
\$FFD6	Pseudo-vecteur SCI (\$0008)
\$FFFE - \$FFFF	Reset (\$E000 vers moniteur Y11)

# Programme d'exemple

```
*-----  
* title      : test mcuhcl1 : leds  
* revision   : 1.0  
* last update : 10/05/2002  
* target     : mc68hc11a1  
* crystal    : 7.3728mhz  
* author     : yann leidwanger  
* email      : tech@jls-info.com  
*-----  
  
*-----  
* equivalences  
*-----  
siram equ    $0000          ; start of internal ram  
eiram equ    $00ff          ; end of internal ram  
seram equ    $2000          ; start of external nvram  
sueram equ   $2040          ; start of user external nvram  
eeram equ    $9fff          ; end of external nvram  
led equ     $a000          ; leds  
pv_sci equ   $0008          ; pv_sci y11 / user  
  
cr equ     $0d             ; ascii codes  
lf equ     $0a  
eot equ    $04  
esc equ    $1b  
  
*-----  
* user pseudos-vectors  
*-----  
pv_spi org    seram          ; start of pv addresses  
      fcb     $7e             ; jump opcode  
      fdb     spi_isr        ; pseudo-vecteur spi  
pv_acci      fcb     $7e             ; jump opcode  
      fdb     acci_isr       ; pseudo-vecteur accumulator in  
pv_acc0      fcb     $7e             ; jump opcode  
      fdb     acc0_isr       ; pseudo-vecteur accumulator overflow  
pv_timo      fcb     $7e             ; jump opcode  
      fdb     timo_isr       ; pseudo-vecteur timer overflow  
pv_toc5      fcb     $7e             ; jump opcode  
      fdb     toc5_isr       ; pseudo-vecteur toc5  
pv_toc4      fcb     $7e             ; jump opcode  
      fdb     toc4_isr       ; pseudo-vecteur toc4  
pv_toc3      fcb     $7e             ; jump opcode  
      fdb     toc3_isr       ; pseudo-vecteur toc3  
pv_toc2      fcb     $7e             ; jump opcode  
      fdb     toc2_isr       ; pseudo-vecteur toc2  
pv_toc1      fcb     $7e             ; jump opcode  
      fdb     toc1_isr       ; pseudo-vecteur toc1  
pv_tic3      fcb     $7e             ; jump opcode  
      fdb     tic3_isr       ; pseudo-vecteur tic3
```

```

pv_tic2
    fcb    $7e            ; jump opcode
    fdb    tic2_isr      ; pseudo-vecteur tic2
pv_tic1
    fcb    $7e            ; jump opcode
    fdb    tic1_isr      ; pseudo-vecteur tic1
pv_rti
    fcb    $7e            ; jump opcode
    fdb    rti_isr       ; pseudo-vecteur real time interrupt
pv_irq
    fcb    $7e            ; jump opcode
    fdb    irq_isr       ; pseudo-vecteur irq

pv_xirq
    fcb    $7e            ; jump opcode
    fdb    xirq_isr      ; pseudo-vecteur xirq
pv_swi
    fcb    $7e            ; jump opcode
    fdb    swi_isr       ; pseudo-vecteur software interrupt
pv_ill
    fcb    $7e            ; jump opcode
    fdb    ill_isr       ; pseudo-vecteur illegal opcode
pv_cop
    fcb    $7e            ; jump opcode
    fdb    cop_isr       ; pseudo-vecteur computer operating properly
pv_clk
    fcb    $7e            ; jump opcode
    fdb    clk_isr       ; pseudo-vecteur clock

*-----
* user code
*-----
    org    sueram
reset
    lds    #eiram         ; sp init
    jsr    init_syst      ; system init
    jsr    init_sci       ; sci init
    cli
    ldx    #msg0
    jsr    out_str_sci    ; menu msg
main
    jsr    sub_led
    bra   main

*-----
* subroutines
*-----
sub_led
    ldaa   #$ff
    staa   count
sub_led1
    ldaa   count
    staa   led           ; led test
    ldaa   #$14         ; 10ms
    jsr    tempo
    dec    count
    bne   sub_led1
    rts

*-----
* tempo = acca x ((accb * 500ns * 5cy) + (500ns *5cy))
* tempo = acca * 500æs
* 1ms   -> $02
* 1.5ms -> $03
* 2ms   -> $04

```

```

* 5ms   -> $0a
* 10ms  -> $14
* 20ms  -> $28
*-----
tempo
    ldab    #$c7
tempol
    decb
    bne     tempol
    deca
    bne     tempo
    rts

*-----
* system init
*-----
init_syst
    ldaa    #$7e           ; jump opcode
    staa    >pv_sci       ; sci pseudo-vector
    ldd     #sci_isr      ; sci_isr address
    std     >pv_sci+1
    clr     led
    rts

*-----
* sci subroutines
*-----
init_sci
    ldaa    #$30           ; 9600,n,8,1
    staa    baud
    clr     sccr1
    ldaa    #$2c
    staa    sccr2         ; enable receive interrupt
    rts

out_sci
    ldab    scsr
    andb    #$80
    beq     out_sci       ; uart empty ?
    ldab    scsr         ; reset flag
    staa    scdr         ; transmit data
    rts

out_str_sci
    ldaa    ,x            ; read location
    cmpa    #eot          ; end of string ?
    beq     out_str_sci2
    jsr     out_sci       ; transmit data
    inx
    ; inc ptr
    bra     out_str_sci

out_str_sci2
    rts

*-----
* constants
*-----
msg0
    fcb     cr
    fcb     lf
    fcc     'Test Leds'
    fcb     cr
    fcb     lf

```

```
        fcb      eot

*-----
* interrupts subroutines
*-----
sci_isr
spi_isr
acci_isr
acco_isr
timo_isr
toc5_isr
toc4_isr
toc3_isr
toc2_isr
toc1_isr
tic3_isr
tic2_isr
tic1_isr
rti_isr
irq_isr
xirq_isr
swi_isr
ill_isr
cop_isr
clk_isr
        rti

*-----
* variables
*-----
count  rmb      1      ; sub_led

        end
```

***J.L.S. INFORMATIQUE***

*2, rue Clément Ader*

*B.P. 50065*

*57972 Yutz Cedex*

*France*

Tél.: 03 82 86 00 16

Fax : 03 82 86 00 12

**[www.jls-info.com](http://www.jls-info.com)**