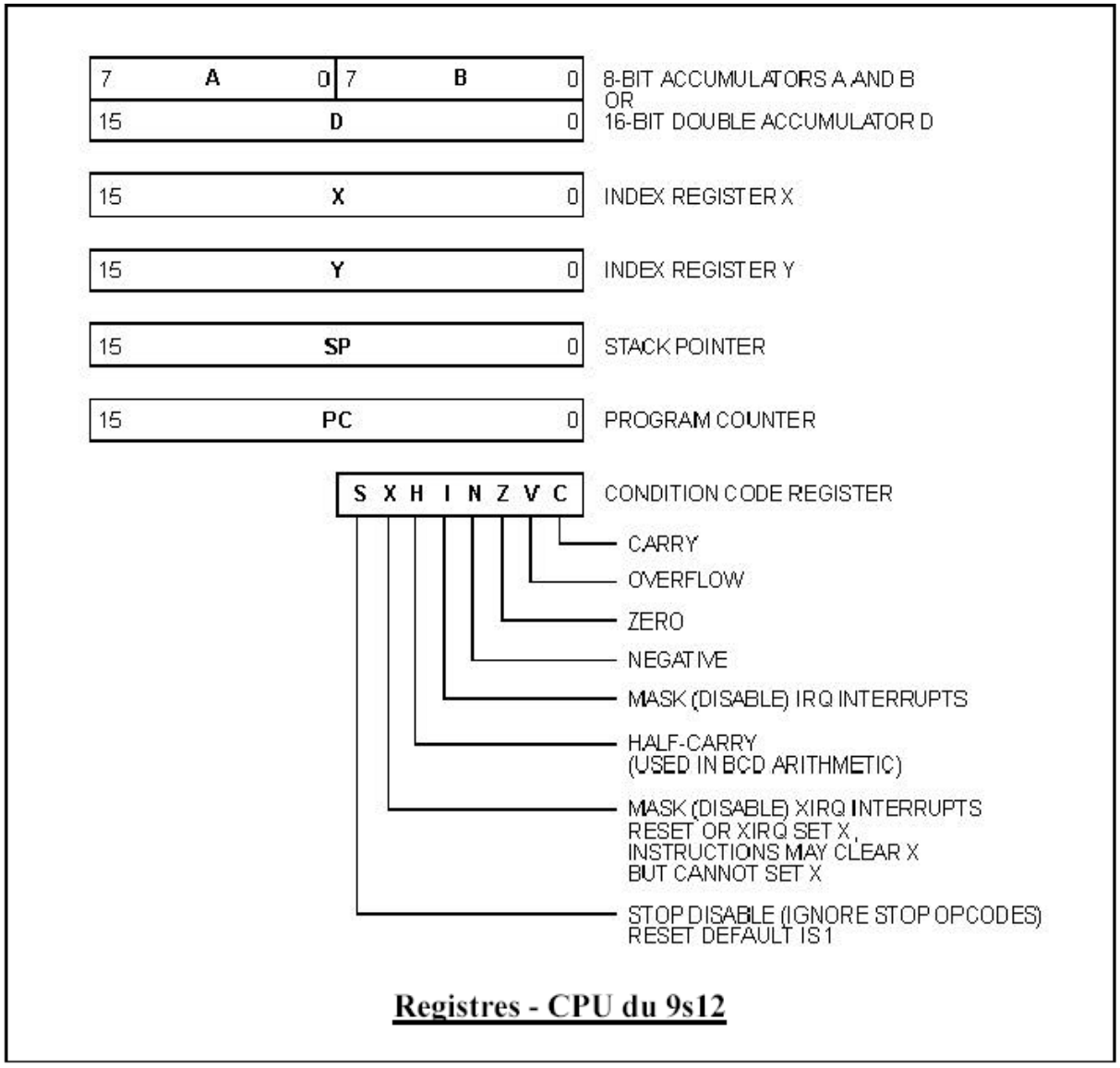
Etudiant: NOM : _____ **Prénom :** _____

IUT GEII Rouen **Sem. 2 – Module II*2***

DOCUMENTS DE REFERENCE
C - TD - TP

Ces documents sont à avoir pour chaque séance de TD et de TP du module ! -

Registres CPU du processeur Freescale 9s12



Registres - CPU du 9s12

Evolution des bits d'état NZVC selon les instructions

	NZVC
ldaa, ldab, ldd, ldx, ldy	⇕⇕0-
movb, movw, abx, aby	----
inx, iny, dex, dey	-⇕--
inca, incb, deca, decb	⇕⇕--
cmpa, cmpb, cpd, cpx, cpy	⇕⇕⇕⇕

Légende
⇕ : bit positionné selon la valeur chargée ou le résultat
0 : bit mis à zéro
1 : bit mis à un
- : bit inchangé

DUT GEII - Informatique des Systèmes Industriels		
UE **UE23**	Matière **Informatique Industrielle**	Volume horaire **12C 20TD 28TP**
Référence **II2**	Module **Architecture des systèmes à processeurs**	Positionnement **S2**

Objectifs :

- Maîtriser l'implémentation des concepts de la programmation structurée et démystifier le langage de haut niveau (exemple : traduction C / Assembleur),
- Comprendre l'architecture d'un système à processeur,
- Comprendre les mécanismes d'interruption.

Compétences minimales :

- Être capable d'écrire un programme langage de haut niveau pour une cible à microprocesseur ou microcontrôleur,
- Savoir interfacer un périphérique, savoir gérer des entrées – sorties,
- Être capable d'évaluer les contraintes de temps dans le cas d'une application simple.

Pré-Requis :

- Module II1 Algorithmique, Programmation
- Module ENSL1 électronique numérique, synthèse logique

Contenu :

- Terminologie : micro-ordinateur, microprocesseur, micro-contrôleur,
- Organisation matérielle d'un micro-contrôleur. Étude de l'espace d'adressage sur un exemple de composant, types de mémoires et leur rôle dans l'architecture,
- Modèle de programmation d'un processeur, jeu d'instructions, exemples de sources en langage assembleur,
- La pile et ses utilisations,
- Analyse du code assembleur généré par un compilateur,
- Interfaces d'entrées-sorties parallèle et série,
- Utilisations des 'timers',
- Fonctionnement en régime d'interruption, procédures de traitement d'interruption.

Modalités de mise en œuvre :

- Utiliser un environnement de développement en langage évolué,
- Écrire des applications sur cible à processeur, mettant en œuvre des périphériques d'entrée/sortie, programmée en langage évolué pouvant inclure des fonctions simples en assembleur (utilisation des instructions de traitement des bits, si elles existent),
- Faire comprendre la part matérielle et la part logicielle dans le traitement des interruptions,
- S'appuyer sur des exemples de programmes de traitement du signal (mise en œuvre de convertisseurs analogique-numérique et numérique-analogique), de dialogue via des interfaces série.

Prolongements :

- Par modules complémentaires MC-II3, MC-II2.

Mots-clés :

- Microcontrôleur, périphériques, architecture, variables, mémoires, registres, ports, interruptions.

Module II2 - Contrat d'études

Organisation générale : le module est étalé sur 8 semaines, avec par semaine :

- 2 x 1h30 C/TD => 24 h TD
- 1 x 3h TP => 24 h TP

Travaux pratiques : ils sont **essentiels** pour la bonne compréhension des notions abordées, et ont pour objectif de faciliter l'acquisition des connaissances. Aucun compte rendu n'est demandé. Les sujets de TP sont distribués en avance, et doivent être lus avant la séance. Un étudiant arrivant en TP n'ayant manifestement pas lu le sujet pourra se voir pénalisé.

Présence : la présence au Cours/TD/TP est obligatoire. En cas d'absence, une justification est indispensable, faute de quoi, si une évaluation a lieu pendant la séance, l'étudiant se verra attribué la note « ABI », empêchant la validation du semestre.

Supports papiers :

- Tous les supports papiers (cours/exercices/TP) ne sont distribués en **une seule fois**, lors de la séance dédiée. En cas d'absence, c'est à l'étudiant de récupérer après coup les documents distribués auprès des autres étudiants.
- La masse de documents distribués est importante (>100 pages), il est primordial d'être organisé, un point clé de ce module est votre capacité à **retrouver rapidement une information** dans les nombreux documents distribués.

Une calculatrice basique sera **nécessaire** en TD et en TP, de préférence avec une fonction de conversion décimal – hexadécimal.

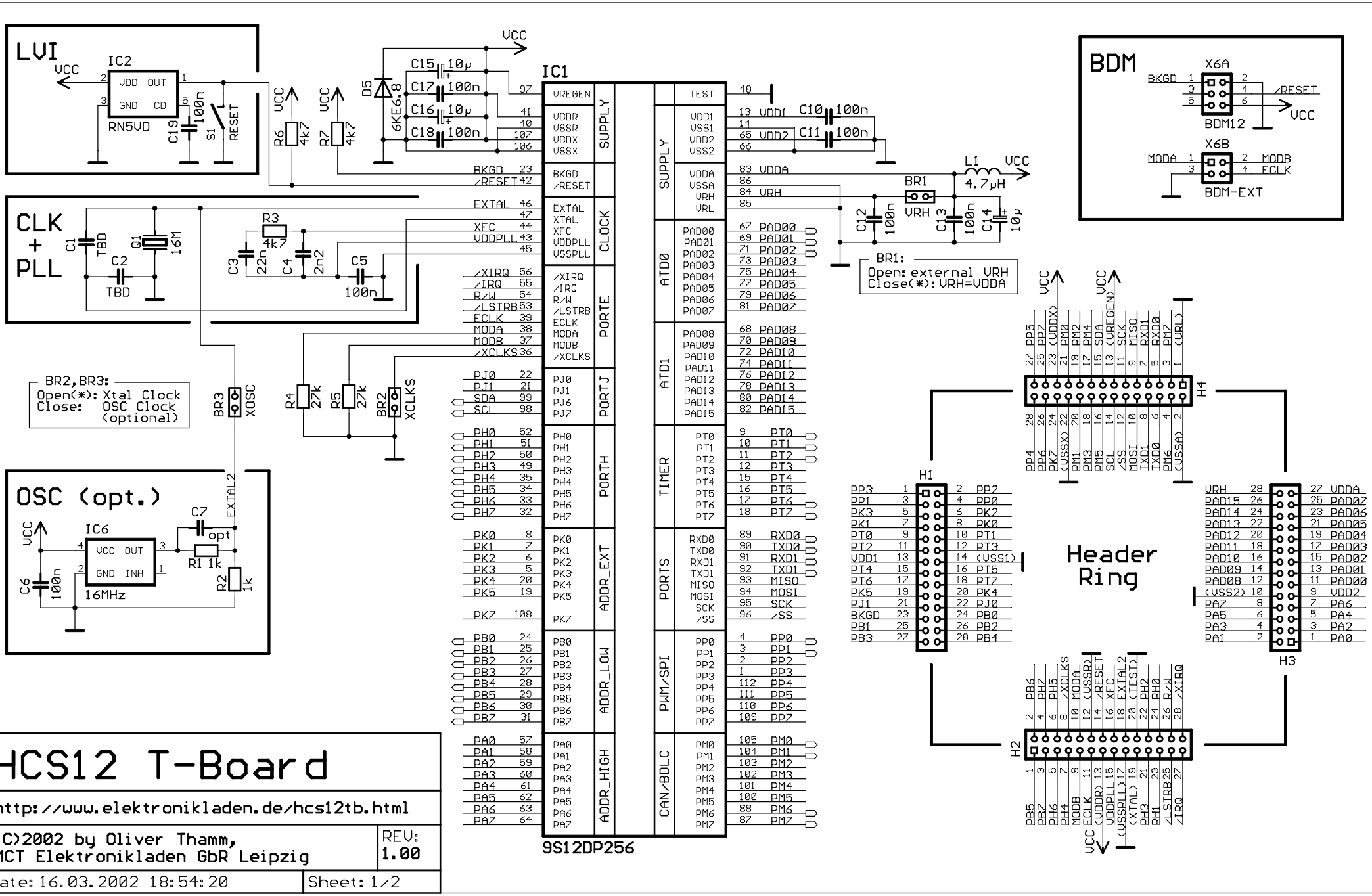
Evaluation : la note du module se composera :

- d'un examen terminal (coeff: 50%), consistant en un travail de synthèse, tous documents autorisés.
- de plusieurs petits devoirs pendant les séances de TD (type QCM ou autre), qui auront lieu de façon de façon aléatoire. (coeff : 30%). Ils pourront être avec ou sans documents, ceci sera laissé à la discrétion de l'enseignant.
- d'une note de TP, déterminée en fonction de l'implication personnelle durant les séances (coeff: 20%).

Afin d'avoir des bons résultats, un travail **constant et régulier** est à fournir : relire les cours, maîtriser les exercices proposés, préparer les TP, etc...

En cas de problème particulier, **ne pas hésiter** à contacter l'enseignant responsable.

Mail : _____ @ univ-rouen.fr



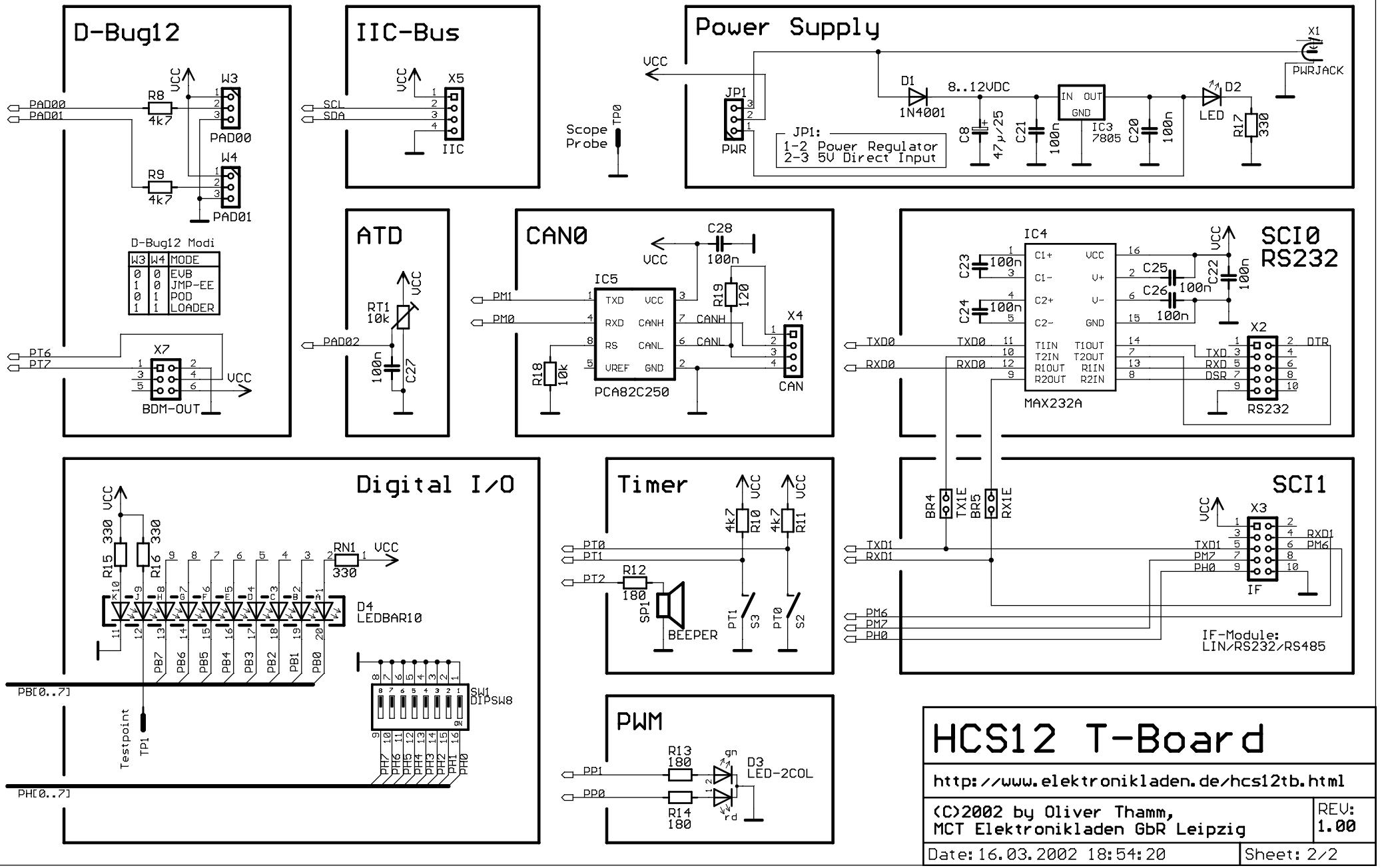
HCS12 T-Board

http://www.elektronikladen.de/hcs12tb.html

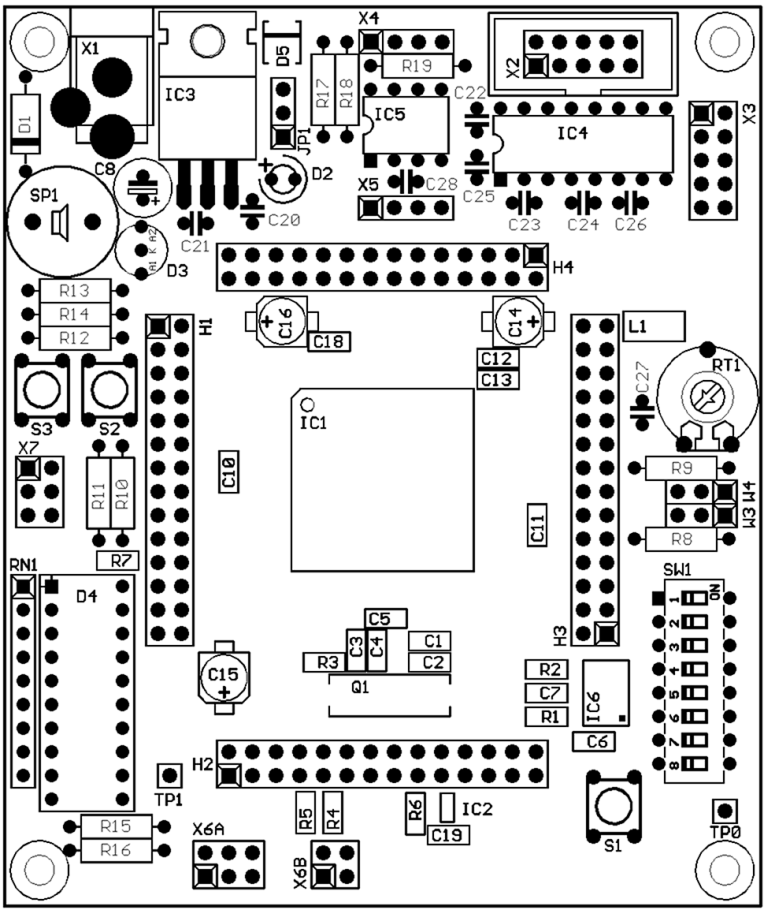
REV: 1.00

Date: 16.03.2002 18:54:20

Sheet: 1/2


D-Bug12
PAD00
PAD01
R8
4k7
R9
4k7
VCC
W3
W4
PAD00
PAD01
D-Bug12 Modi
W3 W4 MODE
0 0 EVB
1 0 JMP-EE
0 1 POD
1 1 LOADER
PT6
PT7
X7
BDM-OUT
IIC-Bus
SCL
SDA
X5
IIC
ATD
RT1
10k
100n
C27
PAD02
Scope Probe
TP0
Power Supply
JP1
PWR
JP1:
1-2 Power Regulator
2-3 5V Direct Input
D1
1N4001
8..12VDC
C8
47µ/25
C21
100n
IC3
7805
C20
100n
D2
LED
R17
330
X1
PWRJACK
CAN0
C28
100n
IC5
TXD
RXD
RS
VREF
VCC
CANH
CANL
GND
PCA82C250
PM1
PM0
R19
120
R18
10k
X4
CAN
SCI0
RS232
IC4
C1+
C1-
C2+
C2-
VCC
V+
V-
GND
T1IN
T2IN
R1OUT
R2OUT
T1OUT
T2OUT
R1IN
R2IN
MAX232A
C23
C24
C25
C26
C22
100n
TXD0
RXD0
X2
DTR
TXD
RXD
DSR
RS232
Digital I/O
R15 330
R16 330
RN1
330
D4
LEDBAR10
PB[0..7]
PB7
PB6
PB5
PB4
PB3
PB2
PB1
PB0
Testpoint
TP1
SW1
DIPSW8
PH[0..7]
PH7
PH6
PH5
PH4
PH3
PH2
PH1
PH0
Timer
R10
R11
4k7
PT0
PT1
PT2
R12
180
SP1
BEEPER
S3
S2
PWM
R13
180
R14
180
D3
LED-2COL
gn
rd
SCI1
BR4
TX1E
BR5
RX1E
TXD1
RXD1
X3
IF
PM6
PM7
PH0
IF-Module:
LIN/RS232/RS485
HCS12 T-Board
http://www.elektronikladen.de/hcs12tb.html
(C)2002 by Oliver Thamm,
MCT Elektronikladen GbR Leipzig
REV:
1.00
Date: 16.03.2002 18:54:20
Sheet: 2/2

3. Parts Location Diagram



Place Plan - Component Side

Figure 1-1 MC9S12DP512 Block Diagram

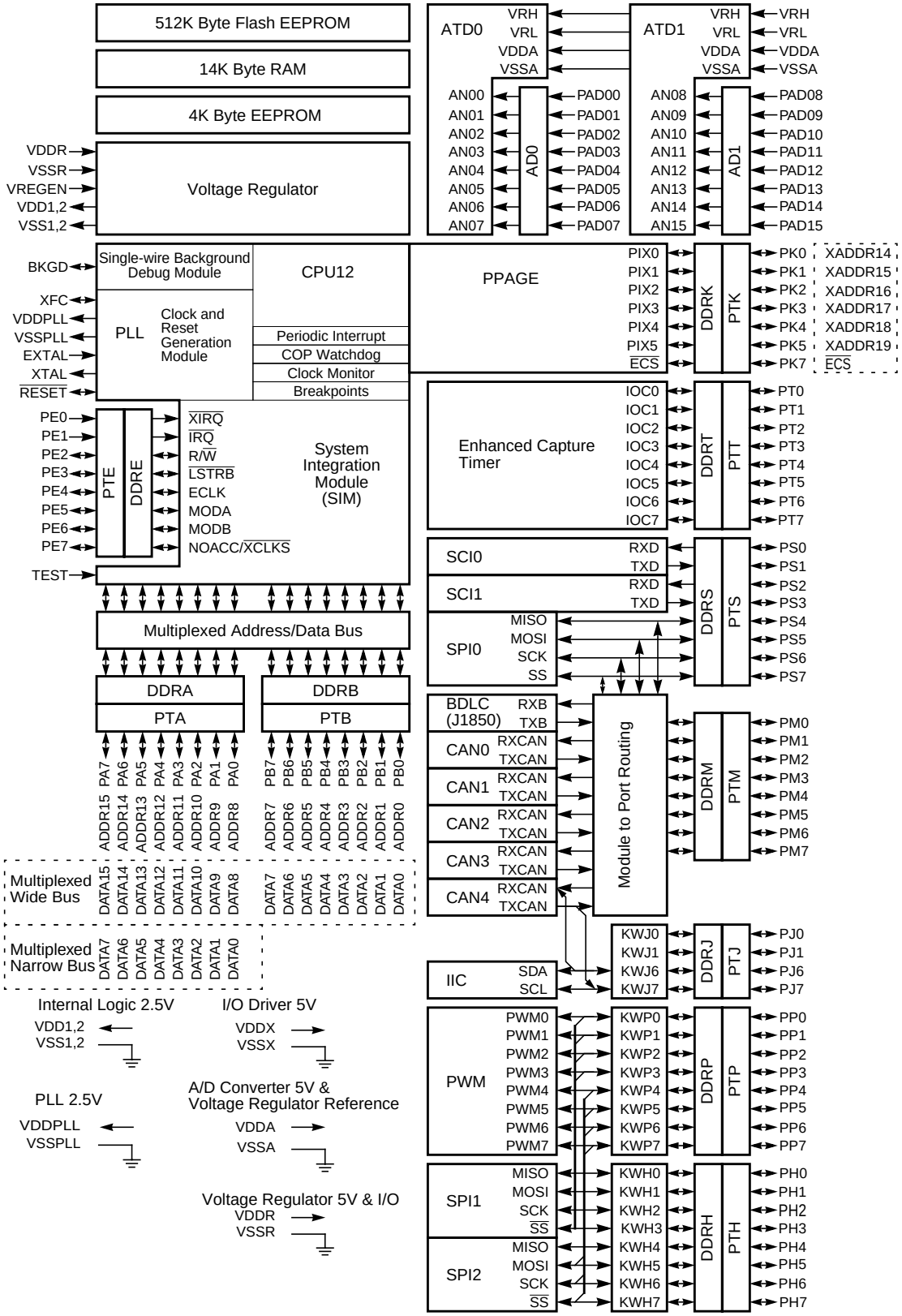
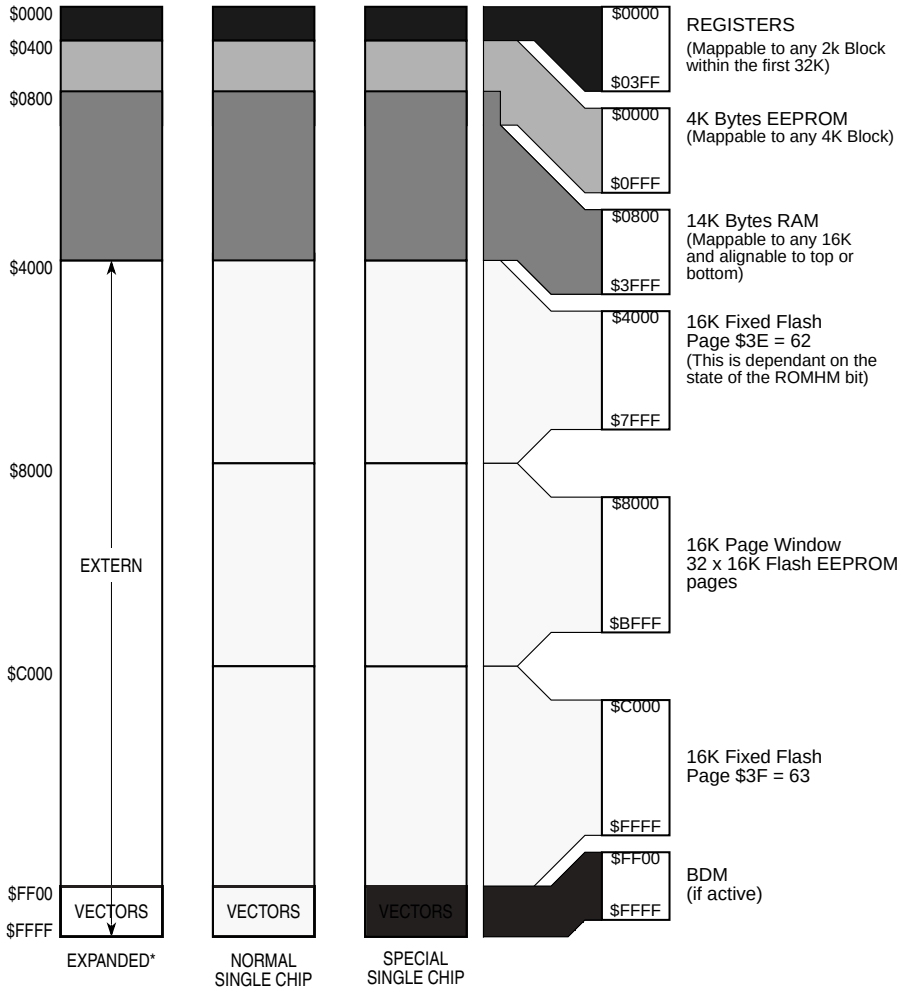


Figure 1-2 MC9S12DP512 Memory Map



* Assuming that a '0' was driven onto port K bit 7 during MCU is reset into normal expanded wide or narrow mode.

Addressing Modes

Table 3-1. M68HC12 Addressing Mode Summary

Addressing Mode	Source Format	Abbreviation	Description
Inherent	**INST** (no externally supplied operands)	INH	Operands (if any) are in CPU registers
Immediate	**INST** #*opr8i* or **INST** #*opr16i*	IMM	Operand is included in instruction stream 8- or 16-bit size implied by context
Direct	**INST** *opr8a*	DIR	Operand is the lower 8 bits of an address in the range $0000–$00FF
Extended	**INST** *opr16a*	EXT	Operand is a 16-bit address
Relative	**INST** *rel8* or **INST** *rel16*	REL	An 8-bit or 16-bit relative offset from the current pc is supplied in the instruction
Indexed (5-bit offset)	**INST** *oprx5*,*xysp*	IDX	5-bit signed constant offset from X, Y, SP, or PC
Indexed (pre-decrement)	**INST** *oprx3*,–*xys*	IDX	Auto pre-decrement x, y, or sp by 1 ~ 8
Indexed (pre-increment)	**INST** *oprx3*,+*xys*	IDX	Auto pre-increment x, y, or sp by 1 ~ 8
Indexed (post-decrement)	**INST** *oprx3*,*xys*–	IDX	Auto post-decrement x, y, or sp by 1 ~ 8
Indexed (post-increment)	**INST** *oprx3*,*xys*+	IDX	Auto post-increment x, y, or sp by 1 ~ 8
Indexed (accumulator offset)	**INST** *abd*,*xysp*	IDX	Indexed with 8-bit (A or B) or 16-bit (D) accumulator offset from X, Y, SP, or PC
Indexed (9-bit offset)	**INST** *oprx9*,*xysp*	IDX1	9-bit signed constant offset from X, Y, SP, or PC (lower 8 bits of offset in one extension byte)
Indexed (16-bit offset)	**INST** *oprx16*,*xysp*	IDX2	16-bit constant offset from X, Y, SP, or PC (16-bit offset in two extension bytes)
Indexed-Indirect (16-bit offset)	**INST [***oprx16*,*xysp***]**	[IDX2]	Pointer to operand is found at... 16-bit constant offset from X, Y, SP, or PC (16-bit offset in two extension bytes)
Indexed-Indirect (D accumulator offset)	**INST [D,***xysp***]**	[D,IDX]	Pointer to operand is found at... X, Y, SP, or PC plus the value in D

Section 5 Resets and Interrupts

5.1 Overview

Consult the Exception Processing section of the CPU12 Reference Manual for information on resets and interrupts.

5.2 Vectors

5.2.1 Vector Table

Table 5-1 lists interrupt sources and vectors in default order of priority.

Table 5-1 Interrupt Vector Locations

Vector Address	Interrupt Source	CCR Mask	Local Enable	HPRIO Value to Elevate
$FFFE, $FFFF	Reset	None	None	–
$FFFC, $FFFD	Clock Monitor fail reset	None	PLLCTL (CME, SCME)	–
$FFFA, $FFFB	COP failure reset	None	COP rate select	–
$FFF8, $FFF9	Unimplemented instruction trap	None	None	–
$FFF6, $FFF7	SWI	None	None	–
$FFF4, $FFF5	XIRQ	X-Bit	None	–
$FFF2, $FFF3	IRQ	I-Bit	IRQCR (IRQEN)	$F2
$FFF0, $FFF1	Real Time Interrupt	I-Bit	CRGINT (RTIE)	$F0
$FFEE, $FFEF	Enhanced Capture Timer channel 0	I-Bit	TIE (C0I)	$EE
$FFEC, $FFED	Enhanced Capture Timer channel 1	I-Bit	TIE (C1I)	$EC
$FFEA, $FFEB	Enhanced Capture Timer channel 2	I-Bit	TIE (C2I)	$EA
$FFE8, $FFE9	Enhanced Capture Timer channel 3	I-Bit	TIE (C3I)	$E8
$FFE6, $FFE7	Enhanced Capture Timer channel 4	I-Bit	TIE (C4I)	$E6
$FFE4, $FFE5	Enhanced Capture Timer channel 5	I-Bit	TIE (C5I)	$E4
$FFE2, $FFE3	Enhanced Capture Timer channel 6	I-Bit	TIE (C6I)	$E2
$FFE0, $FFE1	Enhanced Capture Timer channel 7	I-Bit	TIE (C7I)	$E0
$FFDE, $FFDF	Enhanced Capture Timer overflow	I-Bit	TSRC2 (TOI)	$DE
$FFDC, $FFDD	Pulse accumulator A overflow	I-Bit	PACTL (PAOVI)	$DC
$FFDA, $FFDB	Pulse accumulator input edge	I-Bit	PACTL (PAI)	$DA
$FFD8, $FFD9	SPI0	I-Bit	SPICR1 (SPIE, SPTIE)	$D8
$FFD6, $FFD7	SCI0	I-Bit	SCICR2 (TIE, TCIE, RIE, ILIE)	$D6
$FFD4, $FFD5	SCI1	I-Bit	SCICR2 (TIE, TCIE, RIE, ILIE)	$D4
$FFD2, $FFD3	ATD0	I-Bit	ATDCTL2 (ASCIE)	$D2
$FFD0, $FFD1	ATD1	I-Bit	ATDCTL2 (ASCIE)	$D0
$FFCE, $FFCF	Port J	I-Bit	PIEJ (PIEJ7, PIEJ6, PIEJ1, PIEJ0)	$CE
$FFCC, $FFCD	Port H	I-Bit	PIEH (PIEH7-0)	$CC

$FFCA, $FFCB	Modulus Down Counter underflow	I-Bit	MCCTL (MCZI)	$CA
$FFC8, $FFC9	Pulse Accumulator B Overflow	I-Bit	PBCTL (PBOVI)	$C8
$FFC6, $FFC7	CRG PLL lock	I-Bit	CRGINT (LOCKIE)	$C6
$FFC4, $FFC5	CRG Self Clock Mode	I-Bit	CRGINT (SCMIE)	$C4
$FFC2, $FFC3	BDLC	I-Bit	DLCBCR1 (IE)	$C2
$FFC0, $FFC1	IIC Bus	I-Bit	IBCR (IBIE)	$C0
$FFBE, $FFBF	SPI1	I-Bit	SPICR1 (SPIE, SPTIE)	$BE
$FFBC, $FFBD	SPI2	I-Bit	SPICR1 (SPIE, SPTIE)	$BC
$FFBA, $FFBB	EEPROM	I-Bit	ECNFG (CCIE, CBEIE)	$BA
$FFB8, $FFB9	FLASH	I-Bit	FCNFG (CCIE, CBEIE)	$B8
$FFB6, $FFB7	CAN0 wake-up	I-Bit	CANRIER (WUPIE)	$B6
$FFB4, $FFB5	CAN0 errors	I-Bit	CANRIER (CSCIE, OVRIE)	$B4
$FFB2, $FFB3	CAN0 receive	I-Bit	CANRIER (RXFIE)	$B2
$FFB0, $FFB1	CAN0 transmit	I-Bit	CANTIER (TXEIE2-TXEIE0)	$B0
$FFAE, $FFAF	CAN1 wake-up	I-Bit	CANRIER (WUPIE)	$AE
$FFAC, $FFAD	CAN1 errors	I-Bit	CANRIER (CSCIE, OVRIE)	$AC
$FFAA, $FFAB	CAN1 receive	I-Bit	CANRIER (RXFIE)	$AA
$FFA8, $FFA9	CAN1 transmit	I-Bit	CANTIER (TXEIE2-TXEIE0)	$A8
$FFA6, $FFA7	CAN2 wake-up	I-Bit	CANRIER (WUPIE)	$A6
$FFA4, $FFA5	CAN2 errors	I-Bit	CANRIER (CSCIE, OVRIE)	$A4
$FFA2, $FFA3	CAN2 receive	I-Bit	CANRIER (RXFIE)	$A2
$FFA0, $FFA1	CAN2 transmit	I-Bit	CANTIER (TXEIE2-TXEIE0)	$A0
$FF9E, $FF9F	CAN3 wake-up	I-Bit	CANRIER (WUPIE)	$9E
$FF9C, $FF9D	CAN3 errors	I-Bit	CANRIER (CSCIE, OVRIE)	$9C
$FF9A, $FF9B	CAN3 receive	I-Bit	CANRIER (RXFIE)	$9A
$FF98, $FF99	CAN3 transmit	I-Bit	CANTIER (TXEIE2-TXEIE0)	$98
$FF96, $FF97	CAN4 wake-up	I-Bit	CANRIER (WUPIE)	$96
$FF94, $FF95	CAN4 errors	I-Bit	CANRIER (CSCIE, OVRIE)	$94
$FF92, $FF93	CAN4 receive	I-Bit	CANRIER (RXFIE)	$92
$FF90, $FF91	CAN4 transmit	I-Bit	CANTIER (TXEIE2-TXEIE0)	$90
$FF8E, $FF8F	Port P Interrupt	I-Bit	PIEP (PIEP7-0)	$8E
$FF8C, $FF8D	PWM Emergency Shutdown	I-Bit	PWMSDN (PWMIE)	$8C
$FF80 to $FF8B	Reserved			

5.3 Effects of Reset

When a reset occurs, MCU registers and control bits are changed to known start-up states. Refer to the respective module Block Guides for register reset states.

5.3.1 I/O pins

Refer to the HCS12 Multiplexed External Bus Interface (MEBI) Block Guide for mode dependent pin configuration of port A, B, E and K out of reset.

Refer to the PIM Block Guide for reset configurations of all peripheral module ports.

ABA	Add accumulator B to accumulator A
ABX	Add accumulator B to index register X
ABY	Add accumulator B to index register Y
ADCA	Add with carry to accumulator A
ADCB	Add with carry to accumulator B
ADDA	Add without carry to accumulator A
ADDB	Add without carry to accumulator B
ADDD	Add without carry to accumulator D
ANDA	Logical and with accumulator A
ANDB	Logical and with accumulator B
ANDCC	Logical and CCR with mask
ASL	Arithmetic shift left memory
ASLA	Arithmetic shift left accumulator A
ASLB	Arithmetic shift left accumulator B
ASLD	Arithmetic shift left accumulator D
ASR	Arithmetic shift right memory
ASRA	Arithmetic shift right accumulator A
ASRB	Arithmetic shift right accumulator B
BCC	Branch if carry clear
BCLR	Clear bit(s) in memory
BCS	Branch if carry set
BEQ	Branch if equal
BGE	Branch if greater than or equal
BGND	Enter background debug mode
BGT	Branch if greater than
BHI	Branch if higher
BHS	Branch if higher or same
BITA	Bit test accumulator A
BITB	Bit test accumulator B
BLE	Branch if less than or equal
BLO	Branch if lower
BLS	Branch if lower or same
BLT	Branch if less than
BMI	Branch if minus
BNE	Branch if not equal
BPL	Branch if plus
BRA	Branch always
BRCLR	Branch if bit clear
BRN	Branch never
BRSET	Branch if bits set
BSET	Set bit(s) in memory
BSR	Branch to subroutine
BVC	Branch if overflow cleared
BVS	Branch if overflow set
CALL	call subroutine in extended memory
CBA	Compare accumulator A and B
CLC	Clear carry bit
CLI	Clear interrupt bit
CLR	Clear memory
CLRA	Clear accumulator A
CLRB	Clear accumulator B
CLV	Clear two’s complement overflow bit
CMPA	Compare accumulator A
CMPB	Compare accumulator B
COM	One’s complement on memory
COMA	One’s complement on accumulator A
COMB	One’s complement on accumulator B
CPD	Compare accumulator D
CPS	Compare register SP
CPX	Compare index register X
CPY	Compare index register Y
DAA	Decimal adjust accumulator A
DBEQ	Decrement counter and branch if equal to zero
DBNE	Decrement counter and branch if not equal to zero
DEC	Decrement memory location
DECA	Decrement accumulator A
DECB	Decrement accumulator B
DES	Decrement register SP
DEX	Decrement index register X
DEY	Decrement index register Y
EDIV	Division 32-bits/16 bits (unsigned)
EDIVS	Division 32-bits/16 bits (signed)
EMACS	Multiply and accumulate signed
EMAXD	Maximum of 2 unsigned integer in accumulator D
EMAXM	Maximum of 2 unsigned integer in memory
EMIND	Minimum of 2 unsigned integer in accumulator D
EMINM	Minimum of 2 unsigned integer in memory
EMUL	16-bit * 16-bit multiplication (unsigned)
EMULS	16-bit * 16-bit multiplication (signed)
EORA	Exclusive or with accumulator A
EORB	Exclusive or with accumulator B
ETBL	Table Lookup and Interpolate
EXG	Exchange register contents
FDIV	16-bit / 16-bits fractional divide
IBEQ	Increment and branch if equal to zero
IBNE	Increment and branch if not equal to zero
IDIV	16-bit / 16-bit integer division (unsigned)
IDIVS	16-bit / 16-bit integer division (signed)
INC	Increment memory location
INCA	Increment accumulator A
INCB	Increment accumulator B
INS	Increment register SP
INX	Increment index register X
INY	Increment index register Y
JMP	Jump
JSR	Jump to subroutine
LBCC	Long branch if carry clear
LBCS	Long branch if carry Set
LBEQ	Long branch if equal
LBGE	Long branch if greater than or equal
LBGT	Long branch if greater than
LBHI	Long branch if higher
LBHS	Long branch if higher or same
LBLE	Long branch if Less Than or equal
LBLO	Long branch if lower
LBLS	Long branch if lower or same
LBLT	Long branch if less than
LBMI	Long branch if Minus
LBNE	Long branch if not equal
LBPL	Long branch if plus
LBRA	Long branch always
LBRN	Long branch never
LBVC	Long branch if overflow clear
LBVS	Long branch if overflow set
LDAA	Load accumulator A
LDAB	Load accumulator B
LDD	Load accumulator D
LDS	Load register SP
LDX	Load index register X
LDY	Load index register Y

LEAS	Load SP with effective address
LEAX	Load X with effective address
LEAY	Load Y with effective address
LSL	Logical shift left in memory
LSLA	Logical shift left accumulator A
LSLB	Logical shift left accumulator B
LSLD	Logical shift left accumulator D
LSR	Logical shift left in memory
LSRA	Logical shift right accumulator A
LSRB	Logical shift right accumulator B
LSRD	Logical shift right accumulator D
MAXA	Get maximum of 2 unsigned byte in accumulator A
MAXM	Get maximum of 2 unsigned byte in memory
MEM	Membership function
MINA	Get minimum of 2 unsigned byte in accumulator A
MINM	Get minimum of 2 unsigned byte in memory
MOVB	Memory to memory byte move
MOVW	Memory to memory word move
MUL	8*8-bit unsigned multiplication
NEG	Negate memory (2’s complement)
NEGA	Negate accumulator A (2’s complement)
NEGB	Negate accumulator B (2’s complement)
NOP	No operation
ORAA	Inclusive or with accumulator A
ORAB	Inclusive or with accumulator B
ORCC	Inclusive or CCR with mask
PSHA	Push accumulator A
PSHB	Push accumulator B
PSHC	Push register CCR
PSHD	Push accumulator D
PSHX	Push index register X
PSHY	Push index register Y
PULA	Pop accumulator A
PULB	Pop accumulator B
PULC	Pop register CCR
PULD	Pop accumulator D
PULX	Pop index register X
PULY	Pop index register Y
REV	Rule Evaluation for 8-bits values
REVW	Rule Evaluation for 16-bits values
ROL	Rotate memory left
ROLA	Rotate left accumulator A
ROLB	Rotate left accumulator B
ROR	Rotate right memory
RORA	Rotate right accumulator A
RORB	Rotate right accumulator B
RTC	Return from call
RTI	Return from interrupt
RTS	Return from subroutine
SBA	Subtract accumulators
SBCA	Subtract with carry from accumulator A
SBCB	Subtract with carry from accumulator B
SEC	Set carry bit
SEI	Set interrupt bit
SEV	Set two’s complement overflow bit
SEX	Sign extend into 16-bit register
STAA	Store accumulator A
STAB	Store accumulator B
STD	Store accumulator D
STOP	Stop processing
STS	Store register SP
STX	Store index register X
STY	Store index register Y
SUBA	Subtract without carry from accumulator A
SUBB	Subtract without carry from accumulator B
SUBD	Subtract without carry from accumulator D
SWI	Software interrupt
TAB	Transfer accumulator A to accumulator B
TAP	Transfer accumulator A to CCR
TBA	Transfer accumulator B to accumulator A
TBEQ	Test counter and branch if equal to zero
TBL	8-Bit Table Lookup and Interpolate
TBNE	Test counter and branch if not equal to zero
TFR	Transfer register to register
TPA	Transfer CCR to accumulator A
TRAP	Software interrupt
TST	Test memory
TSTA	Test accumulator A
TSTB	Test accumulator B
TSX	Transfer SP to index register X
TSY	Transfer SP to index register Y
TXS	Transfer index register X to SP
TYS	Transfer index register Y to SP
WAI	Wait for Interrupt
WAV	Weighted Average Calculation
XGDX	Exchange accumulator D with index register X
XGDY	Exchange accumulator D with index register Y

DESCRIPTION DU CONVERTISSEUR ANALOGIQUE NUMERIQUE

Avertissement: Ce document n'est pas une description exhaustive du convertisseur, mais contient ce que vous avez besoin de connaître pour le TP.

Description

Le processeur est doté de 2 Convertisseur Analogiques Numérique (CAN en français), d'une résolution de 10 bits, chacun doté de 8 entrées analogiques. En anglais : ATD (*Analog To Digital converter*).

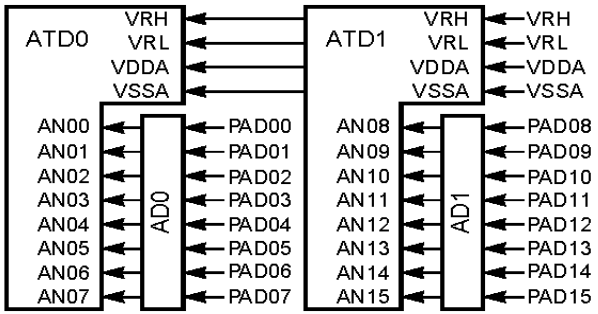
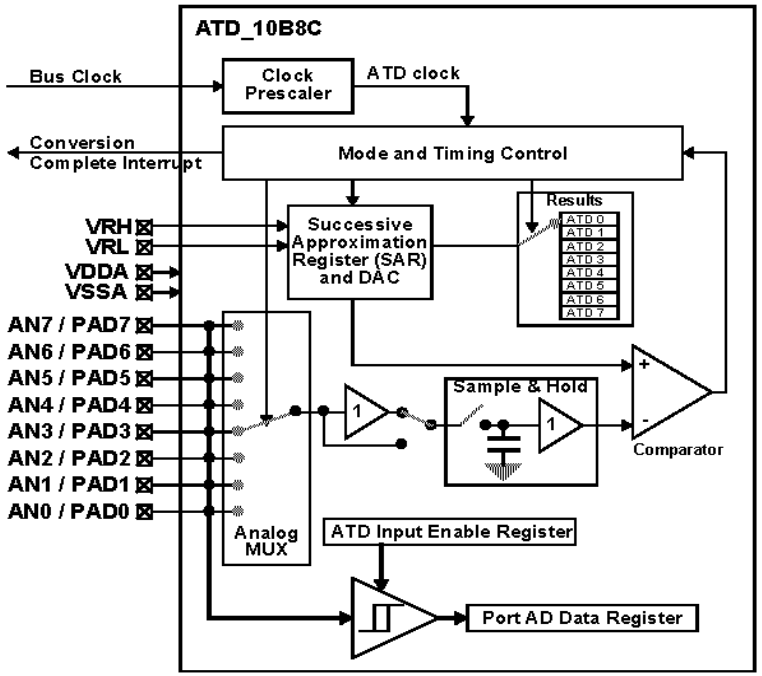


Schéma bloc de chaque convertisseur



- Ces convertisseurs sont pilotés à travers des registres dédiés, on en distingue 3 types :
  - 5 Registres de **contrôle** (8 bits)
  - 2 Registres d'**état** ("*Status Registers*") (8 bits)
  - 8 Registres de **résultat** (16 bits)

- Tous ces registres sont préfixés par les lettres 'ATD' (pour Analog To Digital)

- Les 2 convertisseurs sont indépendants: ils ont chacun leur jeu de registres distincts, identifiés par '0' ou'1' après 'ATD'. Par exemple, le registre décrit ci-après comme 'ATDCTL2' s'appellera en fait ATD**0**CTL2 ou ATD**1**CTL2

- Pour les registres de contrôle et les registres d'état, chaque bit est à considérer séparément. Si le rôle d'un bit vous est inconnu, on peut alors le laisser à sa valeur par défaut (voir la ligne "Reset")

- Il existe un certains nombres de registres de test, dont l'usage est réservé, et qui ne sont pas décrits ici. (Ceci explique la numérotation qui commence à 2 pour les registres de contrôle.)

Utilisation

Le convertisseur peut fonctionner en mode "conversion continue" ou en mode "conversion commandée". Dans ce dernier mode, le démarrage peut se faire soit par logiciel (écriture dans un registre), soit par matériel (front ou niveau externe). Dans ce cas, le signal de synchronisation est à envoyer sur la broche AN7.

La résolution est de 10 bits en standard, mais peut être ramenée à 8 bits, permettant d'accélérer la conversion si besoin.

A chaque démarrage d'une séquence de conversion, le convertisseur effectue automatiquement 'n' conversions (entre 1 et 8, 4 par défaut), et stocke le résultat dans les 8 registres résultats. La fin de la séquence de conversion est signalée par l'activation (=1) du flag SCF (registre ATDSTAT0). On peut aussi détecter individuellement la fin de chaque séquence via le registre ADSTAT1.

Description des registres

1 - Registres de controle

1.1 - ATD Control Register 2 (ATDCTL2)

Ce registre gère la mise en marche du convertisseur, les interruptions, et la synchronisation externe. Une écriture dans ce registre interrompt la conversion en cours, mais ne provoque pas de nouvelle conversion.

	7	6	5	4	3	2	1	0
R W	ADPU	AFFC	AWAI	ETRIGLE	ETRIGP	ETRIGE	ASCIE	ASCIF
RESET:	0	0	0	0	0	0	0	0

ADPU - ATD Power Up
Ce bit permet de désactiver l'alimentation électrique du bloc ATD, permettant ainsi de réduire la consommation globale s'il n'est pas utilisé.

- 1 - Alimentation activée
- 0 - Alimentation désactivée

AFFC — ATD Fast Flag Clear All

- 1 - Remise à zéro rapide des drapeaux de conversion (registre ADSTAT1). Une lecture dans un registre de résultat provoque la RAZ du drapeau correspondant.
- 0 - La RAZ se fait de façon normale: lecture du registre d'état suivi d'une lecture du registre de résultat

AWAI — ATD Power Down in Wait Mode

ETRIGLE — External Trigger Level/Edge Control
Ce bit permet de spécifier si la synchronisation externe se fait sur un front ou sur un niveau

ETRIGP — External Trigger Polarity
Ce bit permet de spécifier la polarité du signal de synchronisation externe

ETRIGLE	ETRIGP	External TriggerSensitivity
0	0	front descendant
0	1	front montant
1	0	niveau bas
1	1	niveau haut

ETRIGE — External Trigger Mode Enable
Ce bit permet de spécifier si on utilise un signal de synchronisation externe

- 0 - Pas de synchronisation externe
- 1 - Synchronisation externe validée

ASCIE - ATD Sequence Complete Interrupt Enable
Ce bit valide ou non la demande d'interruption quand ASCIF = 1

- 0 - Pas d'interruptions
- 1 - Interruptions validées

ASCIF - ATD Sequence Complete Interrupt Flag
Ce bit est identique à SCF si ASCIE est à 1, toujours 0 sinon

1.2 - ATD Control Register 3 (ATDCTL3)

Ce registre gère la longueur de la séquence de conversion, et la façon dont les valeurs sont sauvegardées dans les registres résultats. Une écriture dans ce registre interrompt la conversion en cours, mais ne provoque pas de nouvelle conversion.

	7	6	5	4	3	2	1	0
R W	0	S8C	S4C	S2C	S1C	FIFO	FRZ1	FRZ0
RESET:	0	0	1	0	0	0	0	0

S8C, S4C, S2C, S1C — Conversion Sequence Length
Ces bits gèrent le nombre de conversions réalisées à chaque séquence. Par défaut, le convertisseur réalise 4 conversions à chaque fois qu'une séquence de conversion est initiée.

FIFO — Result Register FIFO Mode

- 0 (mode normal): le résultat de chaque conversion est écrit dans le registre correspondant: résultat de la 1ère conversion dans le 1er registre, 2è dans le 2è; etc.
- 1 (mode FIFO [1]) : les valeurs sont sauvegardées de façon successive dans les registres résultats

FRZ1, FRZ0 — Background Debug Freeze Enable

1.3 - ATD Control Register 4 (ATDCTL4)

Ce registre gère la fréquence de l'horloge de conversion, et la résolution de la conversion (8 ou 10 bits). Une écriture dans ce registre interrompt la conversion en cours, mais ne provoque pas de nouvelle conversion.

	7	6	5	4	3	2	1	0
R W	SRES8	SMP1	SMP0	PRS4	PRS3	PRS2	PRS1	PRS0
RESET:	0	0	0	0	0	1	0	1

= Unimplemented or Reserved

ATD Control Register 4 (ATDCTL4)

SRES8 — A/D Resolution Select
Ce bit définit la résolution de la conversion
1 = 8 bits 0 = 10 bits

SMP1, SMP0 — Sample Time Select

PRS4, PRS3, PRS2, PRS1, PRS0 — ATD Clock Prescaler
Tous ces bits permettent de spécifier la durée de la séquence de conversion, définie à partir d'un rapport de prédivision de la fréquence d'horloge interne du processeur.

[1] FIFO: "*First In, First Out*", correspond à un mode de gestion de mémoire, ou la première valeur entrée est obligatoirement la première sortie.

1.4 - ATD Control Register 5 (ATDCTL5)

Ce registre précise le type de séquence de conversion, ainsi que l'entrée échantillonnée. Une écriture dans ce registre interrompt la conversion en cours, et relance une nouvelle conversion.

	7	6	5	4	3	2	1	0
R W	DJM	DSGN	SCAN	MULT	0	CC	CB	CA
RESET:	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

ATD Control Register 5 (ATDCTL5)

DJM - Result Register Data Justification
Ce bit gère l'alignement du résultat dans le registre-résultat

- 1 - valeur alignée à droite
- 0 - valeur alignée à gauche

Présentation des valeurs dans les registres résultats selon l'alignement :

Alignement à droite:

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
conversion 10 bis	0	0	0	0	0	0	bit9	bit8	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
conversion 8 bits	0	0	0	0	0	0	0	0	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0

Alignement à gauche:

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
conversion 10 bis	bit9	bit8	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	0	0	0	0	0	0
conversion 8 bits	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	0	0	0	0	0	0	0	0

DSGN - Result Register Data Signed or Unsigned Representation
Ce bit permet de spécifier si on souhaite un résultat signé ou non-signé

- 1 - Résultat signé
- 0 - Résultat non signé

SCAN - ce bit définit si la conversion est effectuée en continu, ou une seule fois

- 1 = conversion en continu (mode "*scan*")
- 0 = conversion unique

MULT - Multi Channel Sample Mode

- 0 = Le convertisseur convertit une seule entrée analogique, indiquée par les bits CC, CB, CA
- 1 = Le convertisseur convertit plusieurs entrées, dont le nombre est défini par la longueur de la séquence de conversion (bits S8C, S4C, S2C, S1C). La première entrée est déterminée par les bits CC, CB, CA.

CC, CB, CA — Analog Input Channel Select Code
Ces bits permettent de spécifier quelle est l'entrée analogique sélectionnée parmi les 8 possibles.
Par exemple :

000 : entrée AN0
111 : entrée AN7

2 - Registres d'état ("*Status Registers*")

2.1 - ATD Status Register 0 (ATDSTAT0)

Ce registre contient le drapeau signalant la fin de la séquence de conversion (SCF), avec d'autres bits relatifs au fonctionnement synchronisé.

	7	6	5	4	3	2	1	0
R W	SCF	0	ETORF	FIFOR	0	CC2	CC1	CC0
RESET:	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

ATD Status Register 0 (ATDSTAT0)

SCF - Sequence Complete Flag
Ce bit est mis à 1 par le processeur lorsque la séquence de conversion est terminée. Il est remis à zéro par **l'un** des évènements suivants:

- Ecriture de 1 sur SCF
- Ecriture dans ATDCTL5 (démarrage d'une nouvelle conversion)

Les autres bits sont sans importance pour nous.

2.2 - ATD Status Register 1 (ATDSTAT1)

Ce registre contient les "*flags*" (drapeaux) signalant la fin de la conversion. Lors d'une séquence de conversion, après chaque conversion, un de ces drapeaux est activé ("*Set*"), en commençant pas CCF0.

	7	6	5	4	3	2	1	0
R W	CCF7	CCF6	CCF5	CCF4	CCF3	CCF2	CCF1	CCF0
RESET:	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

3 - Registres Résultat

Pour chaque convertisseur, on dispose de 8 registres **16 bits** "résultat", nommés:

ATDDR0, ATDDR1, ATDDR2, ATDDR3, ..., ATDDR7

On y accèdera :

- En assembleur, via un registre 16 bits (`ldd ATD0DR0` par exemple)
- En C, via un entier (`int a = ATD0DR0;` par exemple)

TABLE ASCII (7 bits)

En hexadécimal

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
10	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
20	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
30	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
40	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
50	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
60	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
70	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

Exemple : 'A' = 65 = $41

* Codes de contrôle (0 à $1F)

NUL (Null)	DLE (Data link escape)
SOH (Start of heading)	DC1 (Device control 1)
STX (Start of text)	DC2 (Device control 2)
ETX (End of text)	DC3 (Device control 3)
EOT (End of transmission)	DC4 (Device control 4)
ENQ (Enquiry)	NAK (Negative acknowledgement)
ACK (Acknowledge)	SYN (Synchronous idle)
BEL (Bell)	ETB (End of transmission block)
BS (Backspace)	CAN (Cancel, annulation)
TAB (Tabulation horizontale)	EM (End of medium, fin du médium)
LF (Line Feed, saut de ligne)	SUB (Substitute, substitut)
VT (Vertical tabulation, tabulation verticale)	ESC (Escape, caractère d'échappement)
FF (Form feed)	FS (File separator, séparateur de fichier)
CR (Carriage return, retour à la ligne)	GS (Group separator, séparateur de groupe)
SO (Shift out)	RS (Record separator, séparateur d'enregistrement)
SI (Shift in)	US (Unit separator, séparateur d'enregistrement)

TD : Conversion de bases numériques

Conversion Hexa -> décimal

$ABC

$12 \times 16^0 =$ 12

$11 \times 16^1 = 11 \times 16 =$ 176

$10 \times 16^2 = 10 \times 256 =$ 2560

TOTAL = 2748

Conversion décimal -> Binaire

21 / 2 = 10, reste 1

10 / 2 = 5, reste 0

5 / 2 = 2, reste 1

2 / 2 = 1, reste 0

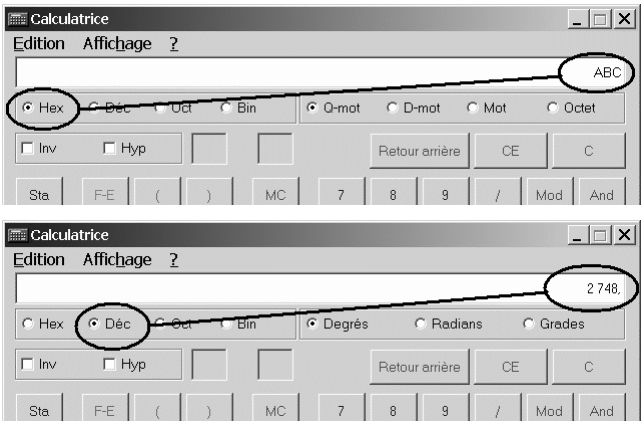
10101

Conversion binaire -> décimal

512	256	128	64	32	16	8	4	2	1
1	**0**	**0**	**1**	**0**	**0**	**1**	**0**	**0**	**0**

512 + 64 + 8 = 584

En pratique (en TP): Calculette Windows



Exercice

	hexa	binaire	décimal
1	**$45**		
2	**$ab**		
3	**$f0**		
4	**$101**		
5	**$15**		
6		**%10001010**	
7		**%11111**	
8		**%10000**	
9		**%10101010**	
10		**%101010101**	
11			**100**
12			**254**
13			**129**
14			**33**
15			**66**
16			**200**

REPRESENTATION DES NOMBRES REELS SUR UN SYSTEME INFORMATIQUE

1) GENERALITES – CHANGEMENT DE BASE DE NUMERATION

-1.1) Tout nombre N peut être représenté en virgule flottante, dans une base de numération quelconque :

$N = M \times B^E$ Avec M: mantisse, B: base de numération, E: Exposant

Exemples : $N = 3{,}1415 . 10^0$ $N = 1011{,}11 . 2^{-4}$

-1.2) Quelle que soit la base de numération, un nombre peut se définir comme une série :

$N = a_n . B^n + a_{n-1} . B^{n-1} + ... + a_0 . B^0 + a_{-1} . B^{-1} + ... + a_{-p} . B^{-p}$

En base 10, on aura par exemple :

$123{,}45 = 1.10^2 + 2.10^1 + 3.10^0 + 4.10^{-1} + 5.10^{-2}$

En base 2 :

$101{,}01 = 1.2^2 + 0.2^1 + 1.2^0 + 0.2^{-1} + 1.2^{-2}$

$= 4 + 0 + 1 + 0 + 1/4 = 5{,}25$

2) CONVERSION DE BASE x EN BASE 10

Exemple en base 2 : 1010,001

Binaire	1	0	1	0	,	0	0	1		
Poids	2^3	2^2	2^1	2^0		2^{-1}	2^{-2}	2^{-3}		
Total	8		2					0,125	=	10,125

Exemple en base 16 : $ABE,C

Hexa	**A**	**B**	**E**	**,**	**C**		
Decimal	10	11	14		12		
Poids	$16^2 = 256$	$16^1 = 16$	$16^0 = 1$		$16^{-1} = 0{,}0625$		
Total	2560	176	14		0,75	=	**2750,75**

3) CONVERSION DE BASE 10 EN BASE 2

Le problème consiste à trouver les coefficients de la série représentant le nombre en base 2. On traite **séparément** la partie entière et la partie fractionnaire

Exemple : 12,125

<u>1) Partie entière</u> : 12

On effectue une suite de division entières par 2, jusqu'à obtenir un nombre inférieur à 2

12 / 2 = 6 reste : 0 (3)
6 / 2 = 3 reste : 0 (2)
3 / 2 = 1 (0) reste : 1 (1)

Puis on regroupe les bits **dans le sens inverse d'obtention**: $12_{(10)} = 1100_{(2)}$
(Le 1er bit est toujours un 1)

<u>2) Partie fractionnaire</u> : 0,125

On effectue une suite de multiplications par 2, jusqu'à obtenir 1 tout rond comme résultat. A chaque étape, on ne prend que **la partie fractionnaire du résultat** (tant que celui ci est supérieur à 1).

Etape			Résultat	Partie entière
1	0,125	x 2 =	0,25	0
2	0,25	x 2 =	0,5	0
3	0,5	x 2 =	1	1

On récupère ensuite les bits **en commençant par le 1er**, et en rajoutant la virgule: 0,001 (le dernier bit est toujours un 1). Le nombre complet **12,125** s'écrit donc en binaire **1100,001** On le vérifie aisément :

$1100{,}001 = 8 + 4 + 1/8 = 8 + 4 + 0{,}125 = 12{,}125$

<u>Rermarque</u> : en général, on n'arrive jamais « pile » à 1. On s'arrête à un nombre de bit donné. Une représentation fractionelle est toujours une **approximation**.

Exemple N=0,2

Etape			Résultat	Partie entière
1	0,2	x 2 =	0,4	0
2	0,4	x 2 =	0,8	0
3	0,8	x 2 =	1,6	1
4	0,6	x 2 =	1,2	1
5	0,2	...		

4) REPRESENTATION DES REELS EN BINAIRE : NORME IEEE 754

-4.1) Mantisse

Une fois la conversion en binaire effectuée, on **normalise** la mantisse : on décale l'exposant jusqu'à avoir un '1' **à gauche** de la virgule :

Exemples :

5,125 (b10) = 101,001 . 2^0 = 1,01001 . 2^2

19,75 (b10) = 10011,11 . 2^0 = 1,001111 . 2^4

On va stocker en mémoire la mantisse **sans le premier '1' à gauche de la virgule**, puisqu'il est toujours là (économie d'un bit).

-4.2) Exposant

L'exposant obtenu est ensuite **biaisé** (décalé), en utilisant un **décalage de référence**, dépendant du nombre de bits utilisés pour représenter l'exposant, en appliquant la formule :

Exposant + décalage = Exposant décalé

On stocke ensuite en mémoire l'**exposant décalé** (on économise ainsi un bit de signe pour l'exposant).

-4.3) Signe mantisse

Le bit de signe est à '0' pour les nombres positifs, à '1' pour les négatifs.

-4.4) Formats de stockage de la norme

La norme **IEEE 754** définit principalement 2 formats de stockage :

- **Simple précision** (type 'C' **`float`**) sur 32 bits :

23 bits de mantisse, 8 bit d'exposant, 1 bit de signe
Exposant de référence (décalage) = 127

- **Double précision** (type 'C' **`double`**) sur 64 bits :

52 bits de mantisse, 11 bit d'exposant, 1 bit de signe

-4.5) Exemples de représentation

-4.5.1) soit à représenter N = 10,5 (base 10) en simple précision

Conversion en binaire : N = 1010,1 . 2^0 = 1,0101 . 2^3,

Mantisse = 1,0101 et on la stocke sans le premier 1 => 01010000000000...

L'exposant est 3, le décalage est 127, donc l'exposant décalé est

3 + 127 = 130 (base 10) = 10000010 (base 2)

La représentation globale sera donc :

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
S	Exposant								Mantisse																						
0	1	0	0	0	0	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Soit, codé en hexa :																															
4		1							2		8					0		0				0		0							

-4.5.2) Soit à retrouver la valeur en base 10 du nombre codé en simple précision par :

$C0 - $D4 - $00 - $00

On convertit en binaire :

	C		0						D		4					0		0				0		0				
1	1	0	0	0	0	0	0	1	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S	Exposant								Mantisse																			

Exposant décalé = 129 Décalage = 127 => Exposant = 2

Mantisse = 0,10101 + 1 = 1,10101 (on doit **rajouter** le 1 à gauche de la virgule !)

soit : N = 1,10101 . 2^2 = 110,101 = 4 + 2 + 0 + 1/2 + 0 + 1/8 = 6,625

Note : on peut tout à fait convertir la mantisse en base 10, et la multiplier **ensuite** par la puissance de 2 :

N = 1,10101 . 2^2 = (1 + 1/2 + 0 + 1/8 + 0 + 1/32) . 4

= (1 + 0,5 + 0,125 + 0,03125) . 4

= 1,65625 . 4 = **6,625**

CODAGE DES INFORMATIONS : EXERCICES

1) Convertir la suite d'octets suivants en caractères ASCII

$49	$45	$45	$45	$20	$37	$35	$34

2) Convertir les mots suivants en codes ASCII (en hexa)

I	**U**	**T**		**G**	**.**	**E**	**.**	**I**	**.**	**I**	**.**

3) Convertir les nombres suivants :

Décimal	Binaire	Hexadécimal
	110101,0001	
47,46875		
		3D,D

4) Convertir le nombre réel 43,75 en représentation IEEE 754 simple précision

Conversion en binaire : 43 = 0,75 =

43, 75 =

Mantisse normalisée = Exposant =

Exposant décalé =

Représentation sur 32 bits :

31	30 29 28 27 26 25 24	23	22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2
S	Exposant		Mantisse

Soit, codé en hexa :

5) Opération inverse : quel nombre est codé par les octets suivants :

$B4 $2B $C0 $00

S	Exposant			Mantisse			

Exposant décalé = Exposant =

Mantisse :

1	,									
1		1/2	1/4	1/8	1/16	1/32	1/64	1/128	1/256	1/512

Mantisse (base 10) =

$2^{exposant}$ =

Soit N =

6) Effectuer les additions/soustractions suivantes en binaires, sur 8 bits (vous utiliserez le complément à deux pour transformer les soustractions en additions de nombres négatifs). Le résultat est-il correct ?

6.1) 45 - 18

	45								
+	-18								

6.2) - 78 - 19

	- 78								
+	- 19								

6.3) 145 + 123

	145								
+	123								