



Kit EVALBOT et E/S

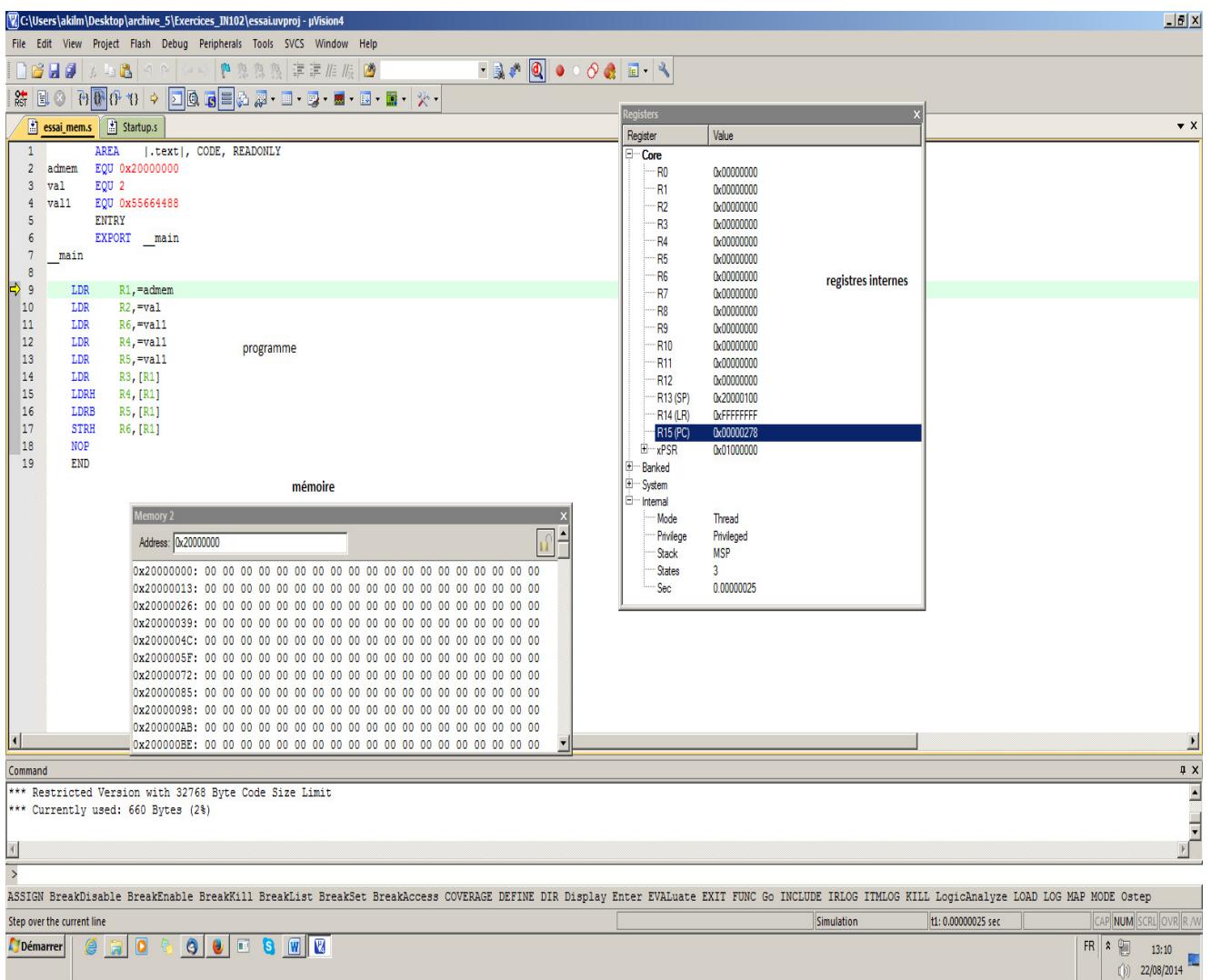
ROSTOM KACHOURI



ESIEE
PARIS

Kit EVALBOT et E/S

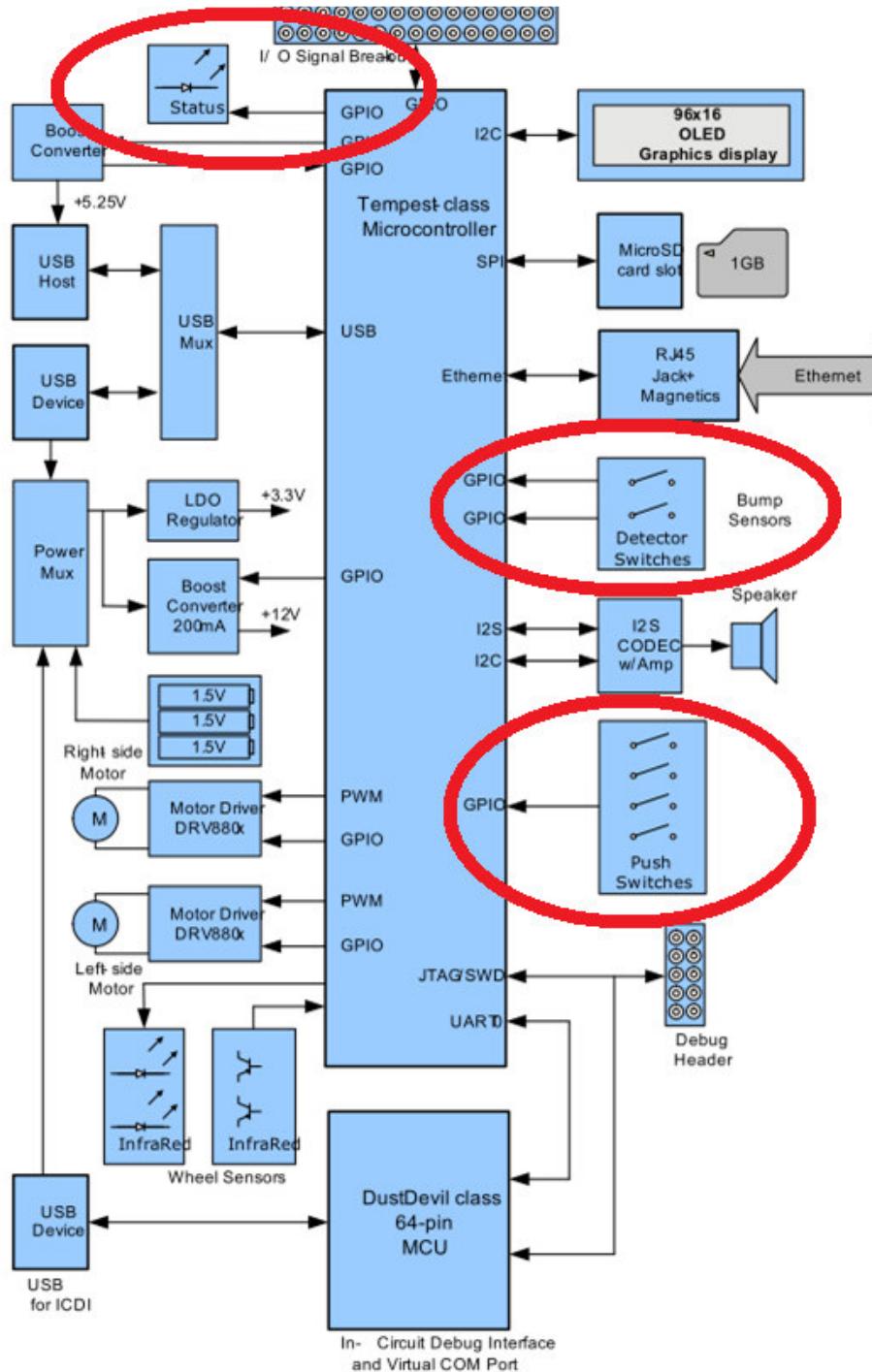
Contexte : Exécution d'un programme par un microprocesseur : exécution Instruction par Instruction, observation des registres internes, de la mémoire, des résultats obtenus (à différentes étapes d'un programme), tests et corrections. Ecriture de programmes, mise en œuvre de ces programmes en utilisant l'environnement de simulation KEIL - uVISION 4. Cette phase est importante pour programmer le ROBOT EVALBOT, afin de le « piloter » (faire avancer avec différents scénarios : suivre une trajectoire, détecter des obstacles, changer d'orientation, etc.).



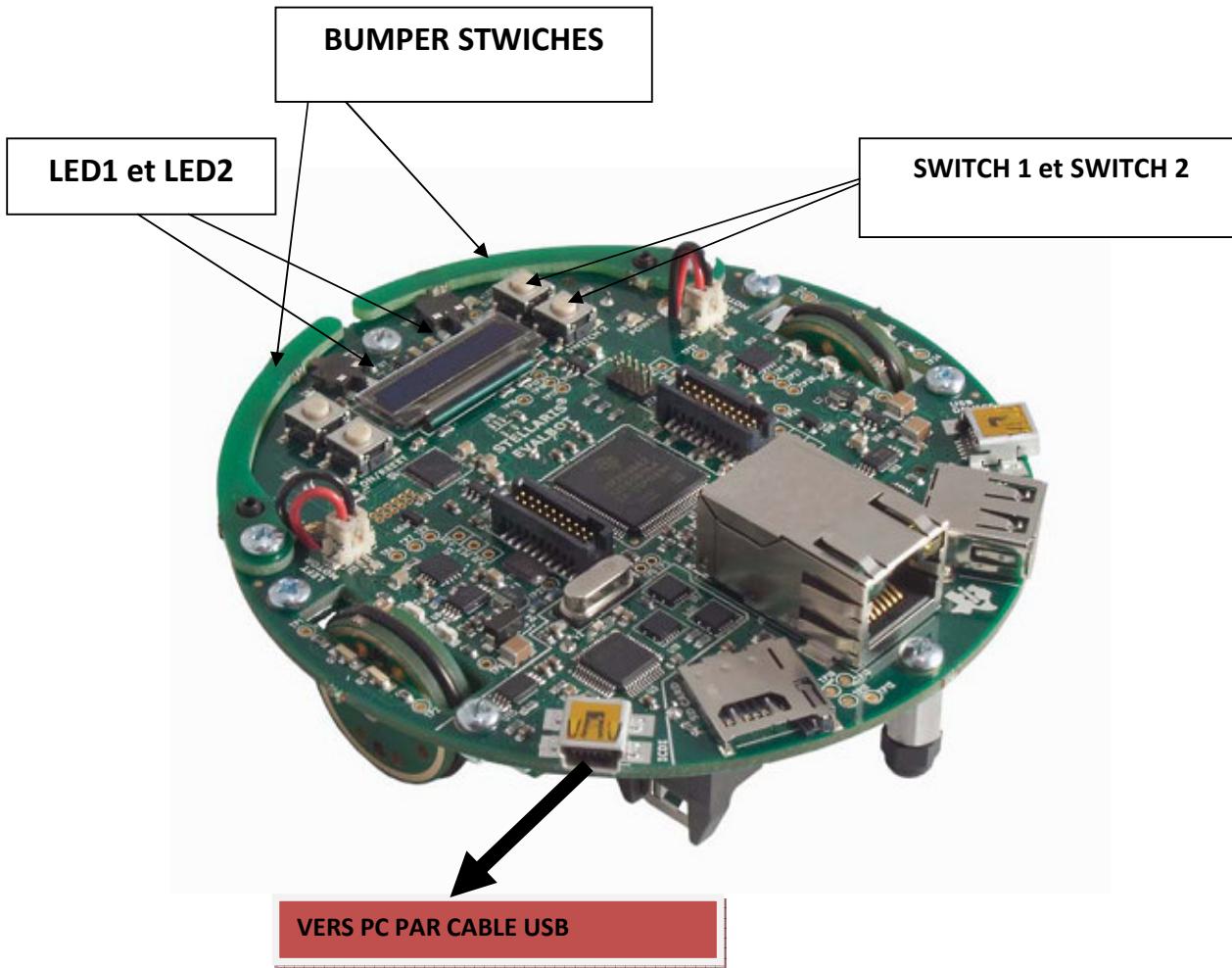
Carte EVALBOT :
Gestion des ports d'Entrée –Sortie (IO)

Organisation fonctionnelle de la carte EVALBOT

Cette carte est basée autour du Microcontrôleur LM3S9B92 Stellaris. Elle dispose de différents périphériques, comme : écran lumineux à OLED bleu 96 x 1, deux moteurs.



Le kit EVALBOT est aussi doté de boutons poussions (SWITCHES), par exemple les « switches pare – choc » (BUMPER SWITCHES) et des boutons poussoirs : **SWITCH 1**, **SWITCH 2**, **OFF** et **ON/RESET**.



La carte EVALBOT dispose aussi de LEDs (Light – Emitting Diode : LED), par exemple les LEDs : LED1 et LED2.

Pour échanger : échanges de données (envoi/réception de données) et des commandes, le microcontrôleur LM3S9B92 dispose de signaux, ces signaux sont accessibles via des broches (appelées pins).

Lignes d'Entrée – Sortie (IO pins)

Parmi ces signaux, on distingue 64 lignes d'Entrée/sortie (appelées I/O : Input/Output). Ces lignes (broches ==pins) sont regroupés en Ports E/S (I/O Ports). On distingue 9 ports : Port

A/B/C/D/E/FH/J. Les ports A/B/C/D/E/H/J sont des ports 8 bits (8-pin ports). Le port F est un port 6 bits (6-pin port) et le Port G est un port 3 bits (3-pin port)

Ces ports sont configurables (c'est-à-dire qu'on peut choisir leur fonctionnement) par programmation. Pour chaque port, on dispose de registres qui permettent d'une part de fixer son fonctionnement et d'autre par échanger des données avec le périphérique qui lui est connecté (réception de la donnée == lecture par le microcontrôleur ou envoie de la donnée == écriture par le microcontrôleur).

Chaque registre est identifié par un nom et il est accessible par une adresse. Il existe différents registres, par exemple pour le port A :

L'adresse de base du port A est 0x4000 4000, on peut donc calculer l'adresse de chaque registre du port A à partir de l'adresse de base + offset de chaque registre, par exemple :

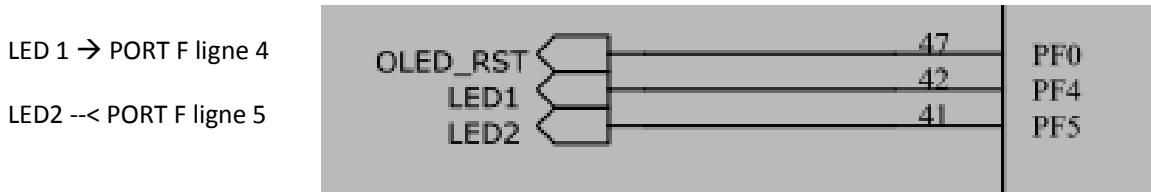
nom	offset	Adresse = adresseBase+Offset	rôle
GPIODATA_PortA	0x00	0x4000 4000	Les 8 premiers bits permettent de forcer à 0 ou 1 l'état de chaque fil si ils sont en sorties ou de les lire si ils sont en entrées (voir registre ci-dessous)
GPIODIR_PortA	0x400	0x4000 4400	Les 8 premiers bits fixent la direction de chaque ligne (0= Input et 1 = output)

D'autres registres sont nécessaires pour fixer un fonctionnement correct du port : GPIODR2R, GPIODEN, GPIOAMSEL (voir annexe).

Illustration

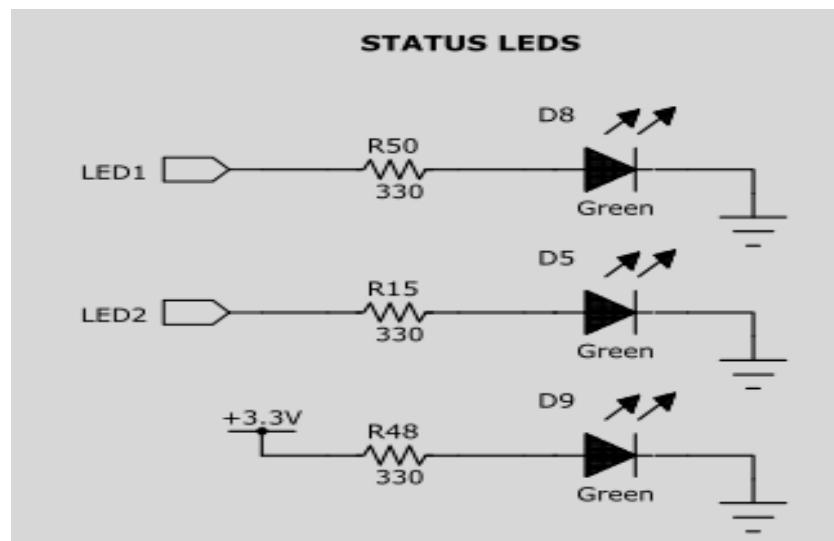
1) Exemple de ports en sorties :

Il faut que les lignes 4 et 5 du PORT F soient en sortie : en effet, d'après les 2 schémas ci-dessous on constate que les leds LED1 et LED2 sont connectées sur ces broches (notées PF4 et PF5). Il faudra donc configurer ces ports en sortie, c'est à dire positionner à 1 (sortie) les bits 4 et 5 du port F grâce au registre GPIODIR_PORTF. Ensuite pour allumer ou éteindre les leds il suffira de changer l'état des bits 4 et 5 du registre GPIODATA_PORTF.



Allumer/éteindre LED

Allumer une LED consiste à mettre 1 sur la ligne correspondante



2) Exemple de ports en entrées :

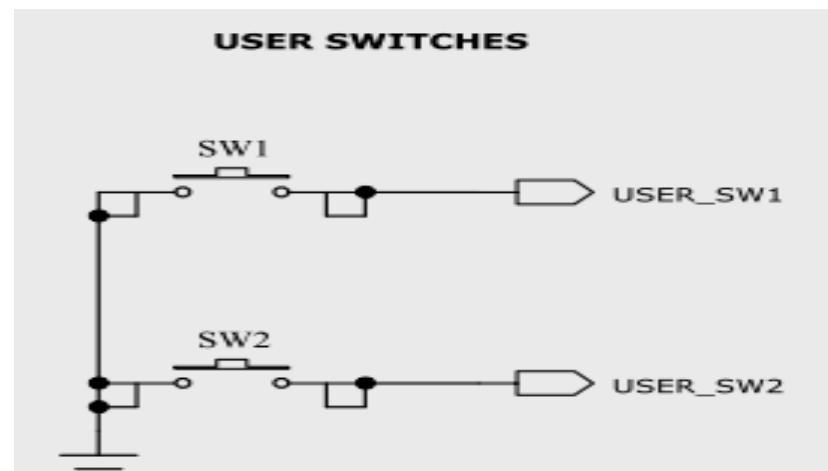
D'après les schémas ci-dessous, les lignes 6 et 7 du Port D devront être configurées en entrées puisque ces lignes sont connectées à des interrupteurs. Puis en lisant sur les bits 6 et 7 du registre GPIO DATA_PORTD on pourra connaitre l'état des interrupteurs SW1 et SW2, c'est-à-dire savoir si ils sont appuyés (niveau 1) ou non (niveau 0).

bouton poussoir (BP) :



Switch (SW)

- SW1 → Port D ligne 6
- SW2 → Port D ligne 7
- Si BP fermé, on lit un 0 sur la ligne correspondante



Les « BUMPER SWITCHES »

BUMPER_R (Right) → Port E ligne 0

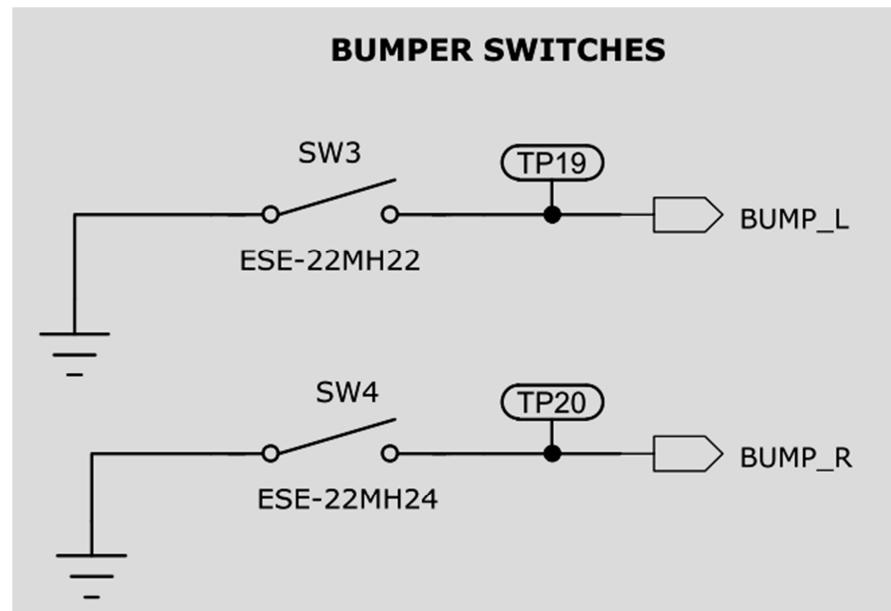
BUMPER_L (Left) → Port E ligne 1



BUMPER actionné : on lit via la

Ligne du port correspondant

l'état logique «0



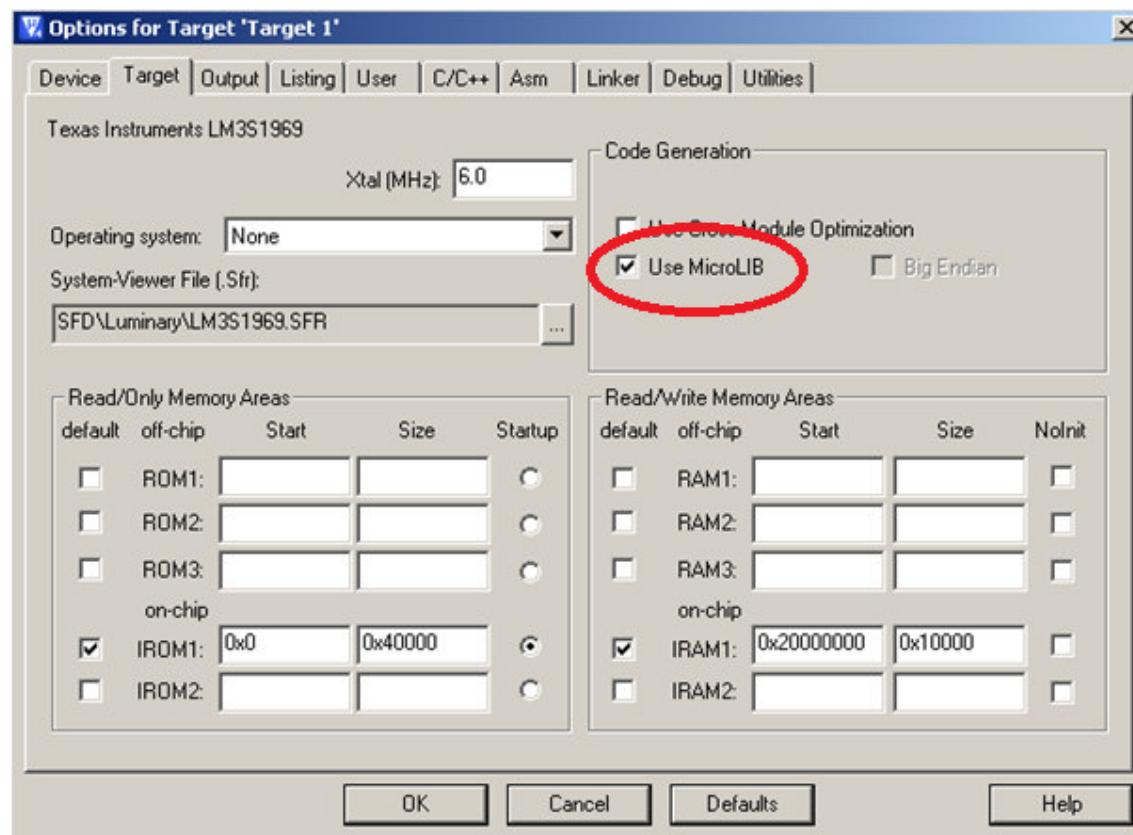
Les séances de travaux pratiques se déroulent en salle informatique – utilisation du simulateur μ VISION4 sur PC et le kit EVALBOT. Il est nécessaire d'aller « au bout des choses » : identifier les erreurs, les corriger, programmer soi-même et s'autoformer.

Comme pour les séances de TD, il faut créer un projet pour le processeur Texas Instrument :

- choisir la cible processeur « LM1968 » (l'Evalbot repose en réalité sur un « LM3S9B92 » mais le « LM1968 » est compatible et simulé)

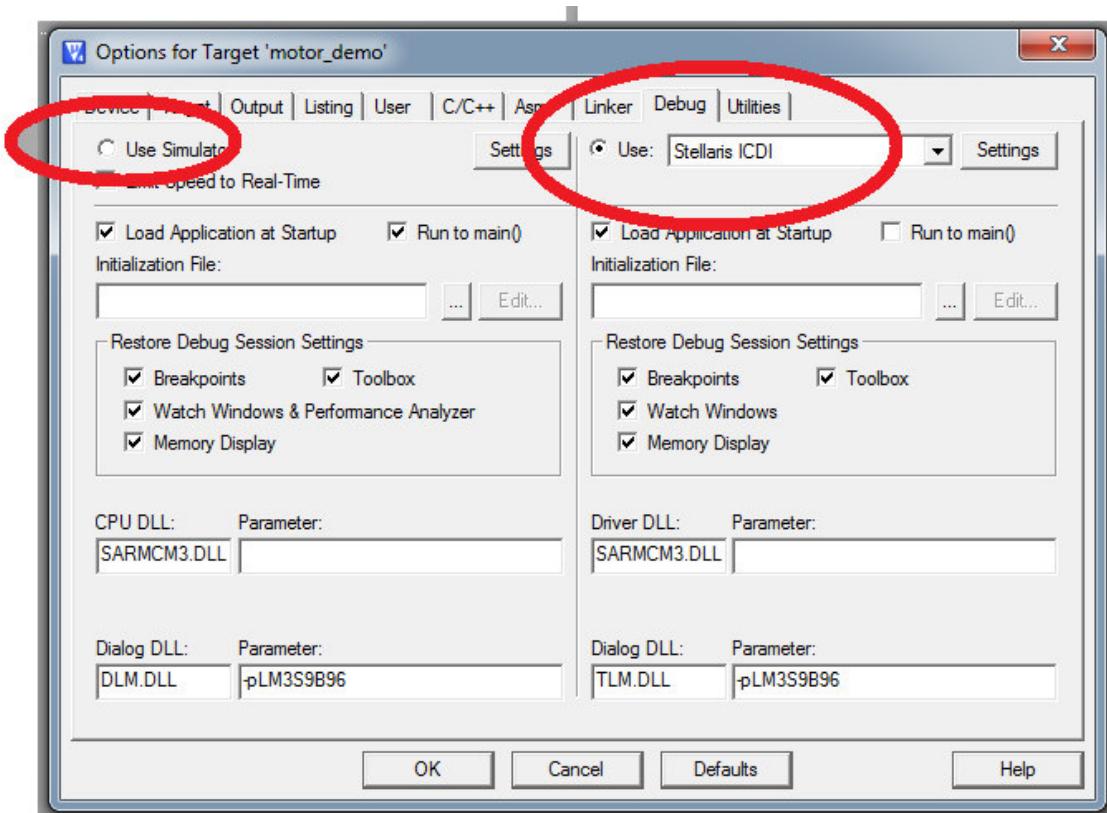
- autorisez la copie du fichier Startup.s proposé par défaut

- cocher « use microlib » dans l'onglet « Target » (pour ne pas rechercher les sections de code spécifique au C)

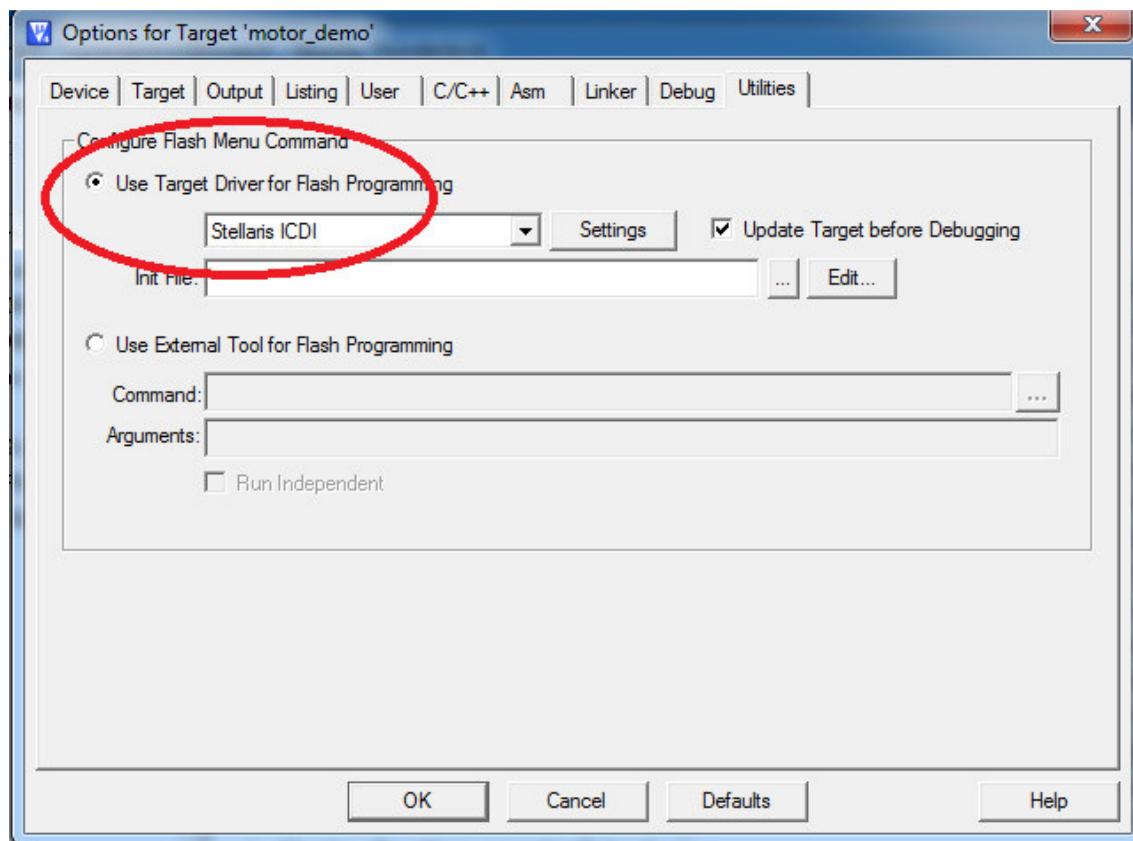


- puis dans les propriétés du projet il faut configurer ainsi pour utiliser l'interface USB de Texas :

- dans onglet « Debug » : use Stellaris ICDI



- dans l'onglet « Utilities » : Stellaris ICDI



Ensuite pour exécuter le code sur le kit Evalbot il faut :

- 1) Brancher l'Evalbot par le câble USB sur le bon port USB (voir page 4)
- 2) Compiler le code et vérifier qu'il n'y a pas d'erreur (Build All)
- 3) Allumer l'Evalbot : bouton ON, il faut alors attendre quelques secondes que le driver USB s'installe
- 4) Transférer le code vers la carte en appuyant sur « Load » :



- 5) Si tout s'est bien passé vous obtenez le message suivant « Programming Done, Verify OK ». Le code est maintenant sur la carte il suffit d'appuyer à nouveau sur le bouton « ON » pour faire exécuter le programme qui vient d'être stocké dans l'Evalbot.

REMARQUE : quand le programme ne fonctionne pas vous pouvez travailler en mode pas à

pas en cliquant sur « Debug » :



Cela transfert votre code dans la mémoire de l'Evalbot (il faut donc l'avoir allumé), puis vous pouvez avancer instruction par instruction en appuyant sur F11 et voir à chaque fois l'instruction qui va s'exécuter, l'état des registres, visualiser le contenu de la mémoire.

Vous pouvez aussi cliquez sur run puis sur stop pour voir où en est votre programme.

Si quand vous faites stop vous constatez que le processeur est bloqué sur l'instruction suivante, cela veut dire qu'il y a eu un accès mémoire invalide.

```
Disassembly
284: ;
285: ; This is the code that gets called
286: ; interrupt. This simply enters an
287: ; for examination by a debugger.
288: ;
289: ;*****
290: IntDefaultHandler
→ 0x00000272 E7FE      B      0x00000272
 291:           B      IntDefaultHandler
 0x00000274 E7FE      B      0x00000274
```

Chaque exercice doit être validé en séance par un des enseignants. Les programmes sources doivent être commentés.

En commentaire, 1^{ère} ligne mettez les noms – prénoms de votre binôme, votre numéro de groupe.

Imprimer les codes sources de vos exercices et rendez-les à la fin de la séance à un des enseignants.

Séance 1 – première partie - Prise en main :

Comprendre, exécuter et assimiler le programme permettant de faire clignoter La LED1, connectée à la broche 4 du port GPIOF. Voir code en annexe.

Séance 1 deuxième partie et séance 2 : Exécuter chacun des programmes des exercices ci-dessous.

Exercices à faire :

Écrire en langage d'assemblage ARM Cortex – M3 les programmes suivants. Exécuter ces programmes et vérifier leur bon fonctionnement sur la carte « Stellaris EVALBOT ».

1. Programme permettant de faire clignoter les deux LED1 et 2, connectées respectivement aux broches 4 et 5 du port GPIOF.
2. Programme permettant de faire allumer les deux LED1 et 2, un appui sur l'un des 2 pare-chocs (bumper) de devant éteint la LED correspondante.
3. Même question que le 2. Les pare-chocs (bumper) sont remplacés par les deux boutons poussoirs SW1 et SW2.
4. Programme permettant de garder les deux LED1 et LED2 éteintes, un appui sur l'un des 2 boutons poussoirs allume la LED correspondante.
5. Programme permettant de commander une LED à l'aide d'un bouton poussoir, un appui sur le bouton poussoir SW1 allume la LED1, un deuxième appui sur le même bouton l'éteint.
6. Même question que le 5 en commandant les deux LEDs. Un appui sur le bouton poussoir SW1 allume les deux LEDs, un deuxième appui les éteint.
7. Programme permettant de changer inversement l'état des deux LEDs. Un appui sur le bouton poussoir SW1 allume la LED1 et éteint la LED2, un deuxième appui change l'état des deux LEDs.

```

; M. Akil, T. Grandpierre, R. Kachouri : département IT - ESIEE Paris -
; 11/2012 - Evalbot (Cortex M3 de Texas Instrument
; programme - clignotement LED1 connectée au port GPIOF

AREA      | .text |, CODE, READONLY

; This register controls the clock gating logic in normal Run mode
; SYSCTL_RCGC2_R (p291 datasheet de lm3s9b92.pdf

SYSCTL_PERIPH_GPIOF      EQU      0x400FE108

; The GPIODATA register is the data register
; GPIO Port F (APB) base: 0x4002.5000 (p416 datasheet de lm3s9B92.pdf

GPIO_PORTF_BASE          EQU      0x40025000

; configure the corresponding pin to be an output
; all GPIO pins are inputs by default
; GPIO Direction (p417 datasheet de lm3s9B92.pdf

GPIO_O_DIR                EQU      0x00000400

; The GPIODR2R register is the 2-mA drive control register
; By default, all GPIO pins have 2-mA drive.
; GPIO 2-mA Drive Select (p428 datasheet de lm3s9B92.pdf)

GPIO_O_DR2R                EQU      0x00000500

; Digital enable register
; To use the pin as a digital input or output, the corresponding; GPIODEN
bit must be set. GPIO Digital Enable (p437 datasheet de ;lm3s9B92.pdf)

GPIO_O_DEN                 EQU      0x0000051C

; Port select - LED1 sur la ligne 4 du port

PORT4                      EQU      0x10

; blinking frequency

DUREE                      EQU      0x002FFFFF

        ENTRY
        EXPORT      __main
__main

; Enable the Port F peripheral clock by setting bit 5 (0x20 == 0b100000)-
; (p291 datasheet de lm3s9B96.pdf), (GPIO::FEDCBA)

        ldr r6, = SYSCTL_PERIPH_GPIOF           ; RCGC2

; Enable clock sur GPIO F où sont branchés les leds (0x20 ==
; 0b100000)
        mov r0, #0x00000020
        str r0, [r6]

```

```

; "There must be a delay of 3 system clocks before any GPIO reg. access
(p413 datasheet de lm3s9B92.pdf)
; tres très important....; pas nécessaire en simulation ou en debug step
; by step...
    nop
    nop
    nop

; CONFIGURATION LED
; une broche (Pin) du portF en sortie (broche 4 : 00010000)

    ldr r6, = GPIO_PORTF_BASE+GPIO_O_DIR
    ldr r0, = PORT4
    str r0, [r6]

; Enable Digital Function

    ldr r6, = GPIO_PORTF_BASE+GPIO_O_DEN
    ldr r0, = PORT4
    str r0, [r6]

; Choix de l'intensité de sortie (2mA)

    ldr r6, = GPIO_PORTF_BASE+GPIO_O_DR2R
    ldr r0, = PORT4
    str r0, [r6]

    mov r2, #0x0000          ; pour éteindre LED

; allumer la led broche 4 (PORT4)

    mov r3, #PORT4           ; Allume portF broche 4 : 00010000

; @data Register = @base + (mask<<2) ==> LED

    ldr r6, = GPIO_PORTF_BASE + (PORT4<<2)

; Fin configuration LED

; CLIGNOTEMENT LED1

loop
    str r2, [r6]             ; Eteint LED car r2 = 0x00

; pour la durée de la boucle d'attente1 (wait1)

    ldr r1, = DUREE

wait1
    subs r1, #1
    bne wait1

; Allume portF broche 4 : 00010000 (contenu de r3)

    str r3, [r6]

;pour la durée de la boucle d'attente2 (wait2)

```

```
ldr r1, = DUREE  
wait2    subs r1, #1  
        bne wait2  
  
        b loop  
  
nop  
END
```


Register 1: GPIO Data (GPIO DATA), offset 0x000

The **GPIO DATA** register is the data register. In software control mode, values written in the **GPIO DATA** register are transferred onto the GPIO port pins if the respective pins have been configured as outputs through the **GPIO Direction (GPIO DIR)** register (see page 418).

In order to write to **GPIO DATA**, the corresponding bits in the mask, resulting from the address bus bits [9:2], must be set. Otherwise, the bit values remain unchanged by the write.

Similarly, the values read from this register are determined for each bit by the mask bit derived from the address used to access the data register, bits [9:2]. Bits that are set in the address mask cause the corresponding bits in **GPIO DATA** to be read, and bits that are clear in the address mask cause the corresponding bits in **GPIO DATA** to be read as 0, regardless of their value.

A read from **GPIO DATA** returns the last bit value written if the respective pins are configured as outputs, or it returns the value on the corresponding input pin when these are configured as inputs. All bits are cleared by a reset.

GPIO Data (GPIO DATA)

GPIO Port A (APB) base: 0x4000.4000

GPIO Port A (AHB) base: 0x4005.8000

GPIO Port B (APB) base: 0x4000.5000

GPIO Port B (AHB) base: 0x4005.9000

GPIO Port C (APB) base: 0x4000.6000

GPIO Port C (AHB) base: 0x4005.A000

GPIO Port D (APB) base: 0x4000.7000

GPIO Port D (AHB) base: 0x4005.B000

GPIO Port E (APB) base: 0x4002.4000

GPIO Port E (AHB) base: 0x4005.C000

GPIO Port F (APB) base: 0x4002.5000

GPIO Port F (AHB) base: 0x4005.D000

GPIO Port G (APB) base: 0x4002.6000

GPIO Port G (AHB) base: 0x4005.E000

GPIO Port H (APB) base: 0x4002.7000

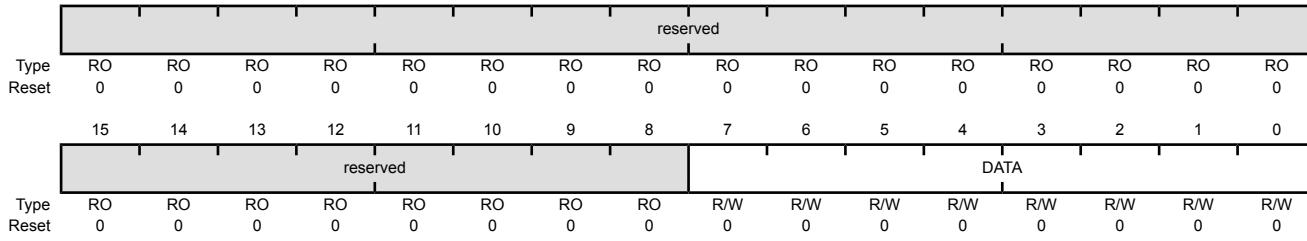
GPIO Port H (AHB) base: 0x4005.F000

GPIO Port J (APB) base: 0x4003.D000

GPIO Port J (AHB) base: 0x4006.0000

Offset 0x000

Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DATA	R/W	0x00	<p>GPIO Data</p> <p>This register is virtually mapped to 256 locations in the address space. To facilitate the reading and writing of data to these registers by independent drivers, the data read from and written to the registers are masked by the eight address lines [9:2]. Reads from this register return its current state. Writes to this register only affect bits that are not masked by ADDR[9:2] and are configured as outputs. See "Data Register Operation" on page 410 for examples of reads and writes.</p>

Register 2: GPIO Direction (GPIODIR), offset 0x400

The **GPIODIR** register is the data direction register. Setting a bit in the **GPIODIR** register configures the corresponding pin to be an output, while clearing a bit configures the corresponding pin to be an input. All bits are cleared by a reset, meaning all GPIO pins are inputs by default.

GPIO Direction (GPIODIR)

GPIO Port A (APB) base: 0x4000.4000

GPIO Port A (AHB) base: 0x4005.8000

GPIO Port B (APB) base: 0x4000.5000

GPIO Port B (AHB) base: 0x4005.9000

GPIO Port C (APB) base: 0x4000.6000

GPIO Port C (AHB) base: 0x4005.A000

GPIO Port D (APB) base: 0x4000.7000

GPIO Port D (AHB) base: 0x4005.B000

GPIO Port E (APB) base: 0x4002.4000

GPIO Port E (AHB) base: 0x4005.C000

GPIO Port F (APB) base: 0x4002.5000

GPIO Port F (AHB) base: 0x4005.D000

GPIO Port G (APB) base: 0x4002.6000

GPIO Port G (AHB) base: 0x4005.E000

GPIO Port H (APB) base: 0x4002.7000

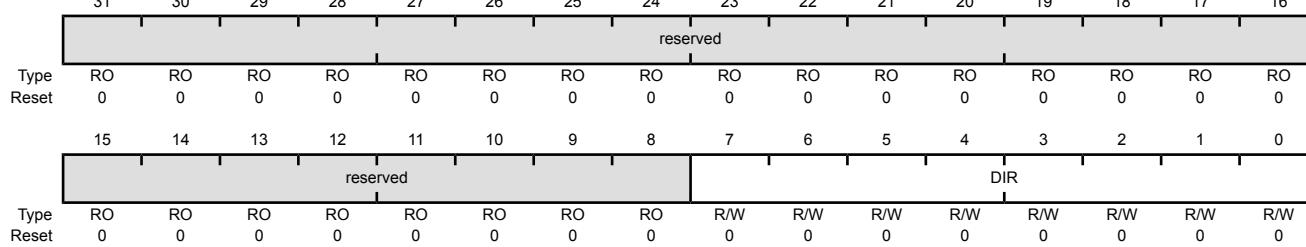
GPIO Port H (AHB) base: 0x4005.F000

GPIO Port J (APB) base: 0x4003.D000

GPIO Port J (AHB) base: 0x4006.0000

Offset 0x400

Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DIR	R/W	0x00	GPIO Data Direction
Value Description				
0 Corresponding pin is an input.				
1 Corresponding pins is an output.				

Register 11: GPIO 2-mA Drive Select (GPIODR2R), offset 0x500

The **GPIODR2R** register is the 2-mA drive control register. Each GPIO signal in the port can be individually configured without affecting the other pads. When setting the **DRV2** bit for a GPIO signal, the corresponding **DRV4** bit in the **GPIODR4R** register and **DRV8** bit in the **GPIODR8R** register are automatically cleared by hardware. By default, all GPIO pins have 2-mA drive.

GPIO 2-mA Drive Select (GPIODR2R)

GPIO Port A (APB) base: 0x4000.4000

GPIO Port A (AHB) base: 0x4005.8000

GPIO Port B (APB) base: 0x4000.5000

GPIO Port B (AHB) base: 0x4005.9000

GPIO Port C (APB) base: 0x4000.6000

GPIO Port C (AHB) base: 0x4005.A000

GPIO Port D (APB) base: 0x4000.7000

GPIO Port D (AHB) base: 0x4005.B000

GPIO Port E (APB) base: 0x4002.4000

GPIO Port E (AHB) base: 0x4005.C000

GPIO Port F (APB) base: 0x4002.5000

GPIO Port F (AHB) base: 0x4005.D000

GPIO Port G (APB) base: 0x4002.6000

GPIO Port G (AHB) base: 0x4005.E000

GPIO Port H (APB) base: 0x4002.7000

GPIO Port H (AHB) base: 0x4005.F000

GPIO Port J (APB) base: 0x4003.D000

GPIO Port J (AHB) base: 0x4006.0000

Offset 0x500

Type R/W, reset 0x0000.00FF



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

7:0	DRV2	R/W	0xFF	Output Pad 2-mA Drive Enable
-----	------	-----	------	------------------------------

Value Description

- 1 The corresponding GPIO pin has 2-mA drive.
- 0 The drive for the corresponding GPIO pin is controlled by the **GPIODR4R** or **GPIODR8R** register.

Setting a bit in either the **GPIODR4** register or the **GPIODR8** register clears the corresponding 2-mA enable bit. The change is effective on the second clock cycle after the write if accessing GPIO via the APB memory aperture. If using AHB access, the change is effective on the next clock cycle.

Register 18: GPIO Digital Enable (GPIODEN), offset 0x51C

Note: Pins configured as digital inputs are Schmitt-triggered.

The **GPIODEN** register is the digital enable register. By default, all GPIO signals except those listed below are configured out of reset to be undriven (tristate). Their digital function is disabled; they do not drive a logic value on the pin and they do not allow the pin voltage into the GPIO receiver. To use the pin as a digital input or output (either GPIO or alternate function), the corresponding GPIODEN bit must be set.

Important: All GPIO pins are configured as GPIOs and tri-stated by default (**GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, **GPIOPUR**=0, and **GPIOPCTL**=0, with the exception of the pins shown in the table below. A Power-On-Reset ($\overline{\text{POR}}$) or asserting $\overline{\text{RST}}$ puts the pins back to their default state.

Table 8-11. GPIO Pins With Non-Zero Reset Values

GPIO Pins	Default State	GPIOAFSEL	GPIODEN	GPIOPDR	GPIOPUR	GPIOPCTL
PA[1:0]	UART0	0	0	0	0	0x1
PA[5:2]	SSIO	0	0	0	0	0x2
PB[3:2]	I ² C0	0	0	0	0	0x3
PC[3:0]	JTAG/SWD	1	1	0	1	0x1

Note: The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is provided for the NMI pin (PB7) and the four JTAG/SWD pins ($\text{PC}[3:0]$). Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 427), **GPIO Pull Up Select (GPIOPUR)** register (see page 433), **GPIO Pull-Down Select (GPIOPDR)** register (see page 435), and **GPIO Digital Enable (GPIODEN)** register (see page 438) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 440) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 441) have been set.

GPIO Digital Enable (GPIODEN)

GPIO Port A (APB) base: 0x4000.4000

GPIO Port A (AHB) base: 0x4005.8000

GPIO Port B (APB) base: 0x4000.5000

GPIO Port B (AHB) base: 0x4005.9000

GPIO Port C (APB) base: 0x4000.6000

GPIO Port C (AHB) base: 0x4005.A000

GPIO Port D (APB) base: 0x4000.7000

GPIO Port D (AHB) base: 0x4005.B000

GPIO Port E (APB) base: 0x4002.4000

GPIO Port E (AHB) base: 0x4005.C000

GPIO Port F (APB) base: 0x4002.5000

GPIO Port F (AHB) base: 0x4005.D000

GPIO Port G (APB) base: 0x4002.6000

GPIO Port G (AHB) base: 0x4005.E000

GPIO Port H (APB) base: 0x4002.7000

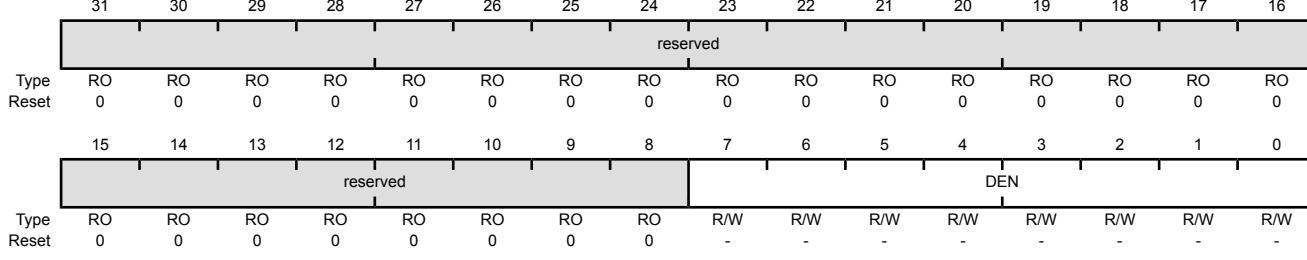
GPIO Port H (AHB) base: 0x4005.F000

GPIO Port J (APB) base: 0x4003.D000

GPIO Port J (AHB) base: 0x4006.0000

Offset 0x51C

Type R/W, reset -



Bit/Field Name Type Reset Description

31:8 reserved RO 0x0000.00 Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

7:0 DEN R/W - Digital Enable

Value Description

0 The digital functions for the corresponding pin are disabled.

1 The digital functions for the corresponding pin are enabled.

The reset value for this register is 0x0000.0000 for GPIO ports that are not listed in Table 8-1 on page 404.

Register 15: GPIO Pull-Up Select (GPIOPUR), offset 0x510

The **GPIOPUR** register is the pull-up control register. When a bit is set, a weak pull-up resistor on the corresponding GPIO signal is enabled. Setting a bit in **GPIOPUR** automatically clears the corresponding bit in the **GPIO Pull-Down Select (GPIOPDR)** register (see page 434). Write access to this register is protected with the **GPIOCR** register. Bits in **GPIOCR** that are cleared prevent writes to the equivalent bit in this register.

Important: All GPIO pins are configured as GPIOs and tri-stated by default (**GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, **GPIOPUR**=0, and **GPIOPCTL**=0, with the exception of the pins shown in the table below. A Power-On-Reset (**POR**) or asserting **RST** puts the pins back to their default state.

Table 8-9. GPIO Pins With Non-Zero Reset Values

GPIO Pins	Default State	GPIOAFSEL	GPIODEN	GPIOPDR	GPIOPUR	GPIOPCTL
PA[1:0]	UART0	0	0	0	0	0x1
PA[5:2]	SSI0	0	0	0	0	0x2
PB[3:2]	I ² C0	0	0	0	0	0x3
PC[3:0]	JTAG/SWD	1	1	0	1	0x1

Note: The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is provided for the NMI pin (**PB7**) and the four JTAG/SWD pins (**PC[3:0]**). Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 426), **GPIO Pull Up Select (GPIOPUR)** register (see page 432), **GPIO Pull-Down Select (GPIOPDR)** register (see page 434), and **GPIO Digital Enable (GPIODEN)** register (see page 437) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 439) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 440) have been set.

GPIO Pull-Up Select (GPIOPUR)

GPIO Port A (APB) base: 0x4000.4000

GPIO Port A (AHB) base: 0x4005.8000

GPIO Port B (APB) base: 0x4000.5000

GPIO Port B (AHB) base: 0x4005.9000

GPIO Port C (APB) base: 0x4000.6000

GPIO Port C (AHB) base: 0x4005.A000

GPIO Port D (APB) base: 0x4000.7000

GPIO Port D (AHB) base: 0x4005.B000

GPIO Port E (APB) base: 0x4002.4000

GPIO Port E (AHB) base: 0x4005.C000

GPIO Port F (APB) base: 0x4002.5000

GPIO Port F (AHB) base: 0x4005.D000

GPIO Port G (APB) base: 0x4002.6000

GPIO Port G (AHB) base: 0x4005.E000

GPIO Port H (APB) base: 0x4002.7000

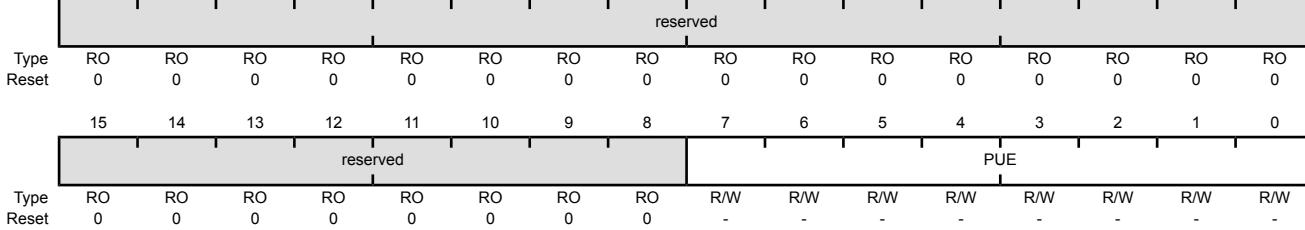
GPIO Port H (AHB) base: 0x4005.F000

GPIO Port J (APB) base: 0x4003.D000

GPIO Port J (AHB) base: 0x4006.0000

Offset 0x510

Type R/W, reset -



Bit/Field	Name	Type	Reset	Description						
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
7:0	PUE	R/W	-	Pad Weak Pull-Up Enable						
				<table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>The corresponding pin has a weak pull-up resistor.</td></tr> <tr> <td>0</td><td>The corresponding pin is not affected.</td></tr> </tbody> </table> <p>Setting a bit in the GPIOPDR register clears the corresponding bit in the GPIOPUR register. The change is effective on the second clock cycle after the write if accessing GPIO via the APB memory aperture. If using AHB access, the change is effective on the next clock cycle. The reset value for this register is 0x0000.0000 for GPIO ports that are not listed in Table 8-1 on page 403.</p>	Value	Description	1	The corresponding pin has a weak pull-up resistor.	0	The corresponding pin is not affected.
Value	Description									
1	The corresponding pin has a weak pull-up resistor.									
0	The corresponding pin is not affected.									