

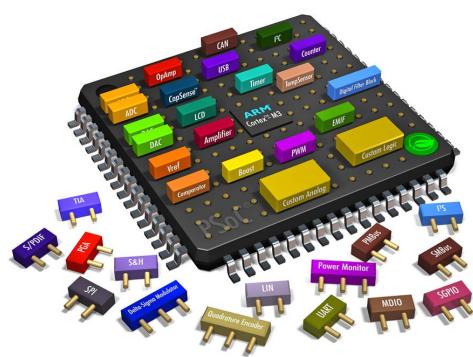
TÉLÉCOM SAINT-ETIENNE

UNIVERSITÉ JEAN MONNET DE SAINT-ETIENNE



ELEC 3

SYSTÈMES EMBARQUÉS À MICROPROCESSEURS



V. SZYMANSKI

version 1.7

Table des matières

1 Systèmes à processeur microprocesseur bare metal	4
1.1 Quelques caractéristiques des PSoC 5[LP]	4
1.2 L'écosystème PSoC	10
1.2.1 l'environnement PSoC Creator	10
1.2.2 Présentation du kit CY8CKIT-059	10
1.2.3 Installations préliminaires	13
1.2.4 Creation d'un premier projet : blinking led	13
1.2.5 Compilation, Programmation, Debuggage	20
1.3 Les entrées/sorties GPIO	20
1.3.1 Présentation	20
1.3.2 Rappels	21
1.3.3 Modes de pilotage des broches	22
1.4 Les interruptions	23
1.4.1 Introduction	23
1.4.2 Architecture du mécanisme d'interruptions sur PSoC 5LP	23
1.4.3 Caractéristiques	24
1.4.4 Mise en oeuvre des interruptions	24
1.4.5 Communication entre le programme principal et l'ISR	28
1.4.6 Gestion de la priorité des interruptions	28
1.5 DMA : Direct Memory Access	28
1.5.1 Introduction	28
1.5.2 Concept de base de DMA	28
1.5.3 Configuration DMA	30
1.6 Applications	31
1.7 Création d'un RTOS	32
1.7.1 Ecriture d'un RTOS	32
1.7.2 Ecriture du RTOS	32
1.8 Bibliographie	32
2 Systèmes à processeurs hosted : le cas linux	1
2.1 GNU Free Documentation License	1
GNU Free Documentation License	1
1. APPLICABILITY AND DEFINITIONS	1
2. VERBATIM COPYING	2
3. COPYING IN QUANTITY	2
4. MODIFICATIONS	2
5. COMBINING DOCUMENTS	2
6. COLLECTIONS OF DOCUMENTS	3
7. AGGREGATION WITH INDEPENDENT WORKS	3
8. TRANSLATION	3
9. TERMINATION	3
10. FUTURE REVISIONS OF THIS LICENSE	3
11. RELICENSING	3
ADDENDUM : How to use this License for your documents	3

Copyright © 2017– Vincent SZYMANSKI

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation ; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

##+TOC : headlines 2

1 Systèmes à processeur microprocesseur bare metal

1.1 Quelques caractéristiques des PSoC 5[LP]

Objectifs :

- A partir de la connaissance de base de la structure des composants PSoC 5, comprendre le mécanisme de construction d'un projet sous PSoC Creator.
- Savoir construire, compiler, programmer et debugger des applications PSoC Creator.
- Savoir créer des applications basiques utilisant les périphériques traditionnels : GPIO, LCD Char, ADC, DAC.

Configuration matérielle :

- processeur ARM Cortex M3 [32 bits data bus]
- Universal Digital Blocks (UDB)
- périphériques : GPIO, ADC, DAC, UART, SPI, I2C, CAN, ...
- Mémoires : SRAM, EEPROM, FLASH
- Direct Memory Access

Rappel concernant les processeurs et applications embarquées :

Pour les applications embarquées on distingue principalement 2 types de processeurs :

- ceux possédant une MMU (Memory Management Unit), c'est à dire une unité de gestion de la mémoire, ce qui permet le support d'un noyau Linux et d'un OS (Operating System). On parle de system HOSTED, pour lesquels DATABUS \geq 32 bits.
- ceux ne possédant pas de MMU et donc ne supportant pas un noyau Linux. On parle alors de système **BARE METAL** ou **FREE STANDING UNIT**. Pour la plupart de ces processeurs DATA BUS \leq 32 bits.

Les PSoC 5[LP] sont des processeurs **FREE STANDING UNIT**. La structure des applications est donc constituée d'un boucle infinie, interrompue par des interruptions matérielles.

Processeur :

- Architecture Harvard à 3 niveaux de pipeline - RISC
- Configurable Nested Vectored Interrupt Controller (NVIC)
- Wake-up interrupt controller (WIC)
- Bit-banding : gestion des entrées/sortie au bit unitaire
- Memory map : cartographie de la mémoire

Architecture du processeur :

My Processor belongs to which architecture

Processor core	Architecture
• ARM7TDMI family <ul style="list-style-type: none">– ARM720T, ARM740T	v4T
• ARM9TDMI family <ul style="list-style-type: none">– ARM920T, ARM922T, ARM940T	v4T
• ARM9E family <ul style="list-style-type: none">– ARM946E-S, ARM966E-S, ARM926EJ-S	v5TE, v5TEJ
• ARM10E family <ul style="list-style-type: none">– ARM1020E, ARM1022E, ARM1026EJ-S	v5TE, v5TEJ
• ARM11 family <ul style="list-style-type: none">– ARM1136J(F)-S– ARM1156T2(Γ)-S– ARM1176JZ(F)-S	v6
• Cortex family <ul style="list-style-type: none">– ARM Cortex-A8– ARM Cortex-R4– ARM Cortex-M3	v6T2 v6Z v7A v7R v7M

FIGURE 1 – Architecture de l'ARM Cortex M3 du PSoC 5[LP]

Processor vs MCU

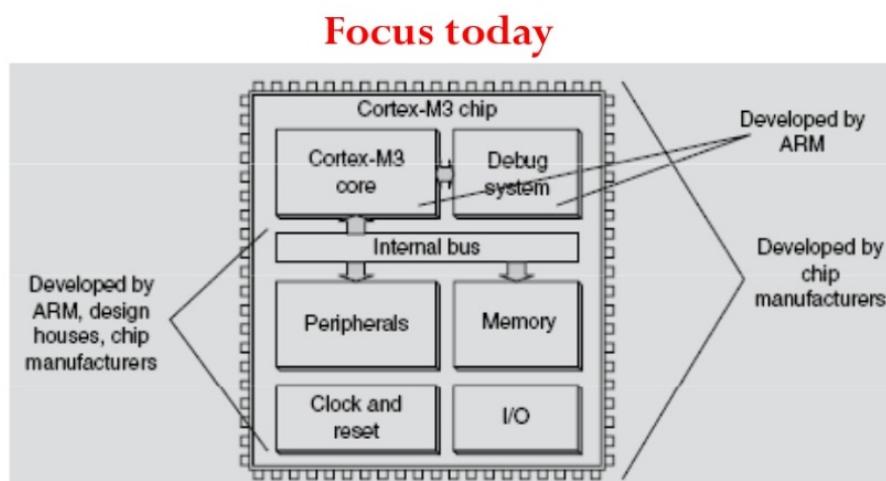


FIGURE 2 – ARM Cortex M3 vs Manufacturers design

ARM Architecture road map

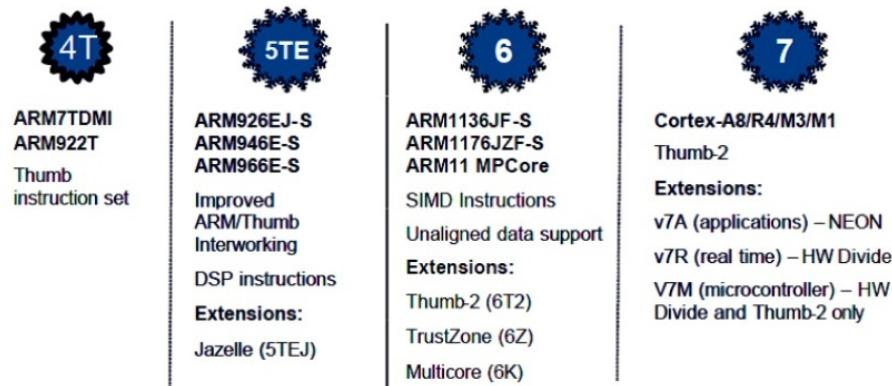


FIGURE 3 – Catégories et feuille de route des processeurs ARM

CORTEX M3 CORE

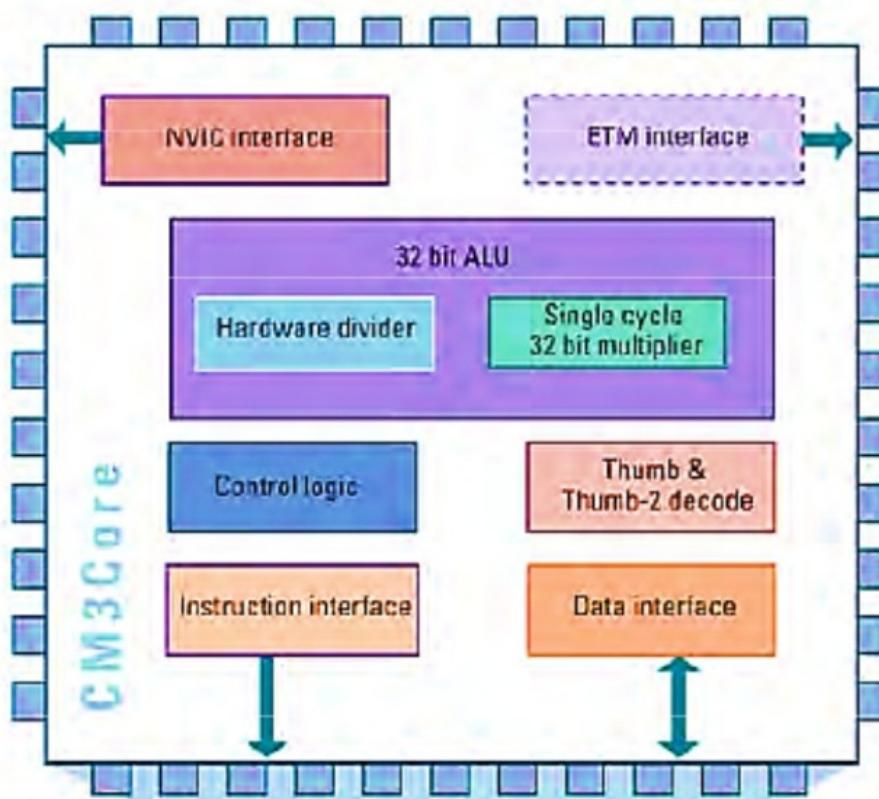


FIGURE 4 – Architecture du cœur ARM Cortex M3

ARM Cortex Pipeline

Like the ARM7 and ARM9 CPUs the Cortex M3 has a three stage pipeline. However, the Cortex-M3 also has branch prediction to minimise the number of pipeline flushes.

Whilst one instruction is being executed, the next is being decoded and a third is being fetched from memory. This works very well for linear code, but when a branch is encountered the pipeline must be flushed and refilled before code can continue to execute. In the ARM7 and ARM9 CPUs branches are very expensive in terms of code performance. In the Cortex CPU the three stage pipeline is enhanced with branch prediction. This means that when a conditional branch instruction is reached, a speculative fetch is performed, so that both destinations of the conditional instruction are available for execution without incurring a performance hit. The worst case is an indirect branch where a speculative fetch cannot be made and the only course of action is to flush the pipeline. While the pipeline is key to the overall performance of the Cortex CPU, no special considerations need to be made in the application code.

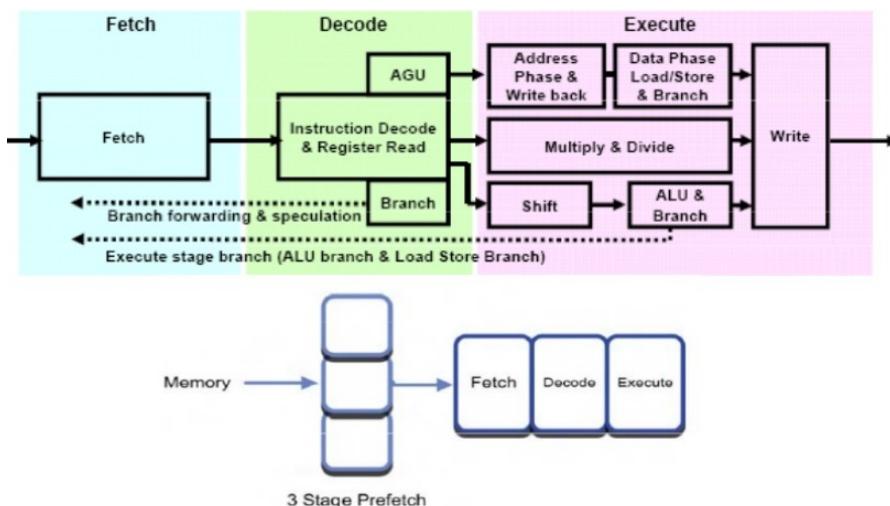


FIGURE 5 – Structure du pipeline ou chaîne de traitement des instructions

Memory map : cartographie mémoire

Unaligned Memory Access

The ARM7 and ARM9 instruction sets are capable of accessing byte, half word and word signed and unsigned variables. This allows the CPU to naturally support integer variables without the need for the sort of software library support typically required in 8 and 16-bit microcontrollers. However, the earlier ARM CPUs do suffer from a disadvantage in that they can only do word or half-word aligned accesses. This restricts the compiler linker in its ability to pack data into the SRAM and some valuable SRAM will be wasted. (This can be as much as 25% depending on the mix of variables used.)

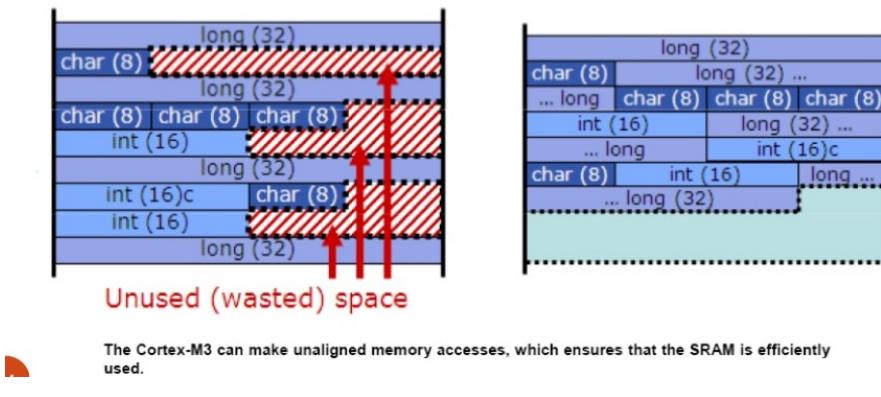


FIGURE 6 – Cartographie mémoire

Bit banding

Les premiers CPU ARM7 et ARM9 étaient seulement capable de réaliser des manipulation de bit sur la RAM et les périphériques mémoires en utilisant des opérations logiques ET et OU. Cela nécessitait une opération READ MODIFY WRITE qui est très coûteuse en terme de nombre de cycles d'horloge pris pour activer et effacer des bits individuellement et en tant qu'espace pris par le code programme pour chaque manipulation de bit. Pour surmonter cette limitation il aurait été possible d'ajouter un bit dédié aux instructions SET et CLEAR ou un processeur purement booléen, mais cela aurait augmenté la taille et la complexité du processeur Cortex. A la place une technique appelée bit banding (regroupement de bits) qui permet la manipulation directe de bits sur des sections d'espace mémoire SRAM ou de périphériques sans avoir besoin d'instructions spécifiques. Les régions adressables par bit de l'espace mémoire du Cortex sont composées de la région bit band (qui va jusqu'à 1 Mo de mémoire réelle ou de registres de périphériques).

La carte mémoire possède 2 régions d'alias de 32 Mo qui accèdent chacune à une région de bit band de 1 Mo :

- un accès à la région d'alias de 32 Mo qui est mappée à la région de bit band dédiée à la SRAM de 1 Mo
- un accès à la région d'alias de 32 Mo qui est mappée à la région de bit band dédiée périphériques de 1 Mo

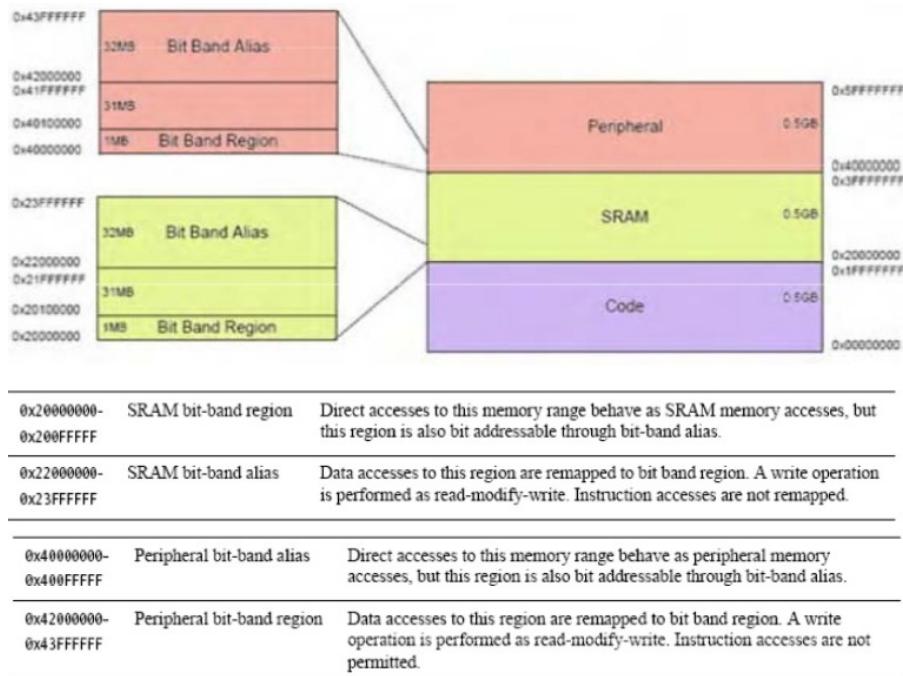


FIGURE 7 – Région de bit band

Accès au bit band et à la région d'alias

Figure 2-1 shows examples of bit-band mapping between the SRAM bit-band alias region and the SRAM bit-band region:

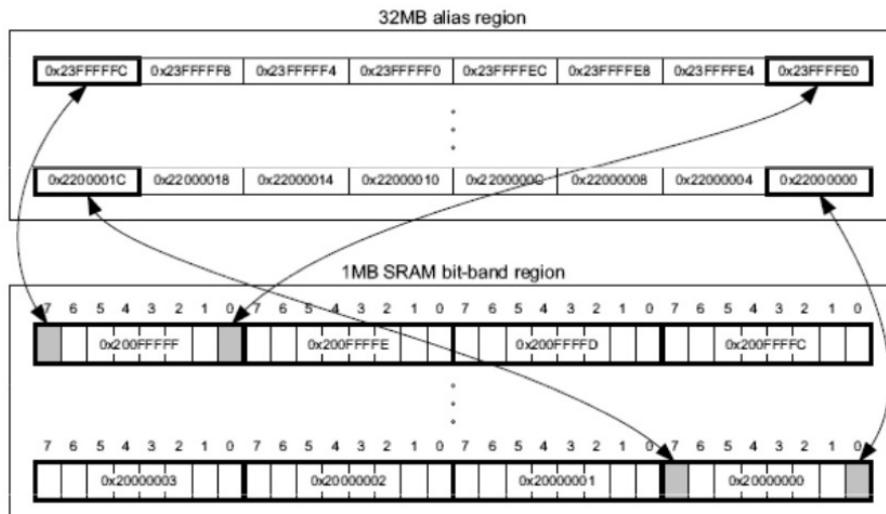


FIGURE 8 – Bit-band mapping

Read Modify Write Vs Bit Banding

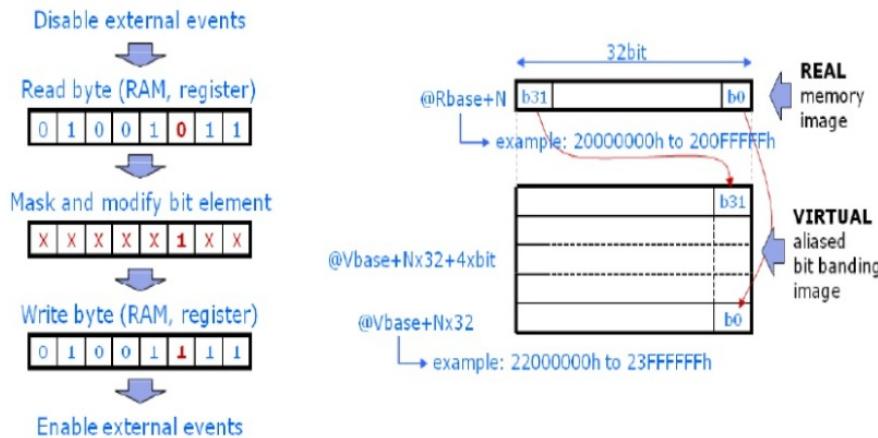


FIGURE 9 – Bit-banding suite

Accès direct à une région de bit-band

1.2 L'écosystème PSoC

1.2.1 l'environnement PSoC Creator

L'environnement de développement intégré (*IDE* : *Integrated Development Environment*) PSoC Creator permet le développement de projets PSoC pour les cibles PSoC 3, PSoC 4, PSoC 5[LP] et PSoC 6. Il permet également le débogage et la programmation de ces différentes puces. Il permet la conception intégrale d'un projet PSoC :

- saisie du schéma électrique
- utilisation et/ou création de périphériques
- écriture du code (en C ou assembleur ARM) de l'application
- compilation du code
- programmation du composant PSoC
- débogage de l'application pas à pas

La dernière version est la 4.1 en octobre 2017

1.2.2 Présentation du kit CY8CKIT-059

Le kit CY8CKIT-059 est représenté ci-dessous :

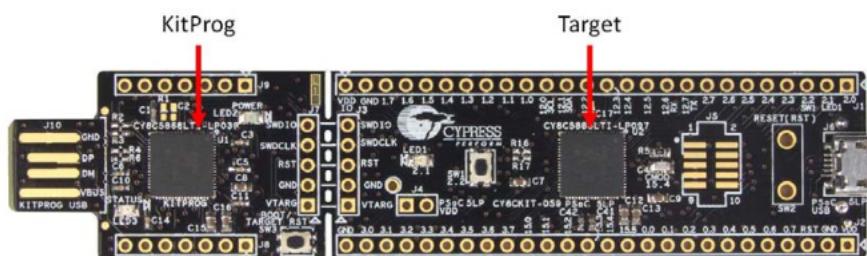


FIGURE 10 – Photo du kit PSoC CY8CKIT-059

On observe que le kit possède 2 PSoC 5[LP] :

- la cible (target) : CY8C5888LTI-LP097 QFN68, PSoC 5[LP] servant à notre application qui sera configurée et programmée
- le programmeur (KitProg) : CY8C5868LTI-LP039 QFN68 servant à programmer et à déboguer notre cible.

Le programmeur possède :

- une connexion USB avec le PC
- une connection avec l'interface SWD de la cible.

Ces 2 PSoC 5[LP] appartiennent à la même famille et sont constitués notamment :

- d'un processeur ARM Cortex M3, architecture RISC 32 bits à 3 niveaux de pipeline avec contrôleur DMA (Direct Memory Access)
- d'un partie reconfigurable UDB
- de périphériques numériques (re/configurables ou non)
- de périphériques analogiques

Attention, ces kits sont fragiles électriquement et mécaniquement. Ils nécessitent certaines précautions :

- Veiller à utiliser une rallonge USB afin de pas exercer de contrainte mécanique sur le connecteur USB.
- A défaut d'utiliser un bracelet antistatique, toucher une prise de terre avec votre main avant d'établir un contact physique avec la carte
- La carte n'est pas protégée contre les surtensions, surcharges et Décharges ElectroStatiques (DES/ESD). Ainsi lorsqu'elle est configurée en 3,3 V en tension d'alimentation, elle ne supporte pas de recevoir du 5V, sur l'une de ses broches d'entrée/sortie

Ci-joint un extrait des caractéristiques de la famille :



PSoC® 5LP: CY8C58LP Family Datasheet

Programmable System-on-Chip (PSoC®)

General Description

PSoC® 5LP is a true programmable embedded system-on-chip, integrating configurable analog and digital peripherals, memory, and a microcontroller on a single chip. The PSoC 5LP architecture boosts performance through:

- 32-bit ARM Cortex-M3 core plus DMA controller and digital filter processor, at up to 80 MHz
- Ultra low power with industry's widest voltage range
- Programmable digital and analog peripherals enable custom functions
- Flexible routing of any analog or digital peripheral function to any pin

PSoC devices employ a highly configurable system-on-chip architecture for embedded control design. They integrate configurable analog and digital circuits, controlled by an on-chip microcontroller. A single PSoC device can integrate as many as 100 digital and analog peripheral functions, reducing design time, board space, power consumption, and system cost while improving system quality.

Features

■ Operating characteristics

- Voltage range: 1.71 to 5.5 V, up to 6 power domains
- Temperature range (ambient): -40 to 85 °C^[1]
Extended temperature parts: -40 to 105 °C
- DC to 80-MHz operation
- Power modes
 - Active mode 3.1 mA at 6 MHz, and 15.4 mA at 48 MHz
 - 2-µA sleep mode
 - 300-nA hibernate mode with RAM retention
 - Boost regulator from 0.5-V input up to 5-V output

■ Performance

- 32-bit ARM Cortex-M3 CPU, 32 interrupt inputs
- 24-channel direct memory access (DMA) controller
- 24-bit 64-tap fixed-point digital filter processor (DFB)

■ Memories

- Up to 256 KB program flash, with cache and security features
- Up to 32 KB additional flash for error correcting code (ECC)
- Up to 64 KB RAM
- 2 KB EEPROM

■ Digital peripherals

- Four 16-bit timer, counter, and PWM (TCPWM) blocks
- I²C, 1 Mbps bus speed
- USB 2.0 certified Full-Speed (FS) 12 Mbps peripheral interface (TID#10840032) using internal oscillator^[2]
- Full CAN 2.0b, 16 Rx, 8 Tx buffers
- 20 to 24 universal digital blocks (UDB), programmable to create any number of functions:
 - 8-, 16-, 24-, and 32-bit timers, counters, and PWMs
 - I²C, UART, SPI, I²S, LIN 2.0 interfaces
 - Cyclic redundancy check (CRC)
 - Pseudo random sequence (PRS) generators
 - Quadrature decoders
 - Gate-level logic functions

■ Programmable clocking

- 3- to 74-MHz internal oscillator, 1% accuracy at 3 MHz
- 4- to 25-MHz external crystal oscillator
- Internal PLL clock generation up to 80 MHz
- Low-power internal oscillator at 1, 33, and 100 kHz
- 32.768-kHz external watch crystal oscillator
- 12 clock dividers routable to any peripheral or I/O

■ Analog peripherals

- Configurable 8- to 20-bit delta-sigma ADC
- Up to two 12-bit SAR ADCs
- Four 8-bit DACs
- Four comparators
- Four opamps
- Four programmable analog blocks, to create:
 - Programmable gain amplifier (PGA)
 - Transimpedance amplifier (TIA)
 - Mixer
 - Sample and hold circuit
- CapSense® support, up to 62 sensors
- 1.024 V ±0.1% internal voltage reference

■ Versatile I/O system

- 46 to 72 I/O pins – up to 62 general-purpose I/Os (GPIOs)
- Up to eight performance I/O (SIO) pins
 - 25 mA current sink
 - Programmable input threshold and output high voltages
 - Can act as a general-purpose comparator
 - Hot swap capability and overvoltage tolerance
- Two USBIO pins that can be used as GPIOs
- Route any digital or analog peripheral to any GPIO
- LCD direct drive from any GPIO, up to 46 × 16 segments
- CapSense support from any GPIO
- 1.2-V to 5.5-V interface voltages, up to four power domains

■ Programming, debug, and trace

- JTAG (4-wire), serial wire debug (SWD) (2-wire), single wire viewer (SWV), and Traceport (5-wire) interfaces
- ARM debug and trace modules embedded in the CPU core
- Bootloader programming through I²C, SPI, UART, USB, and other interfaces

■ Package options: 68-pin QFN, 100-pin TQFP, and 99-pin CSP

■ Development support with free PSoC Creator™ tool

- Schematic and firmware design support
- Over 100 PSoC Components™ integrate multiple ICs and system interfaces into one PSoC. Components are free embedded ICs represented by icons. Drag and drop component icons to design systems in PSoC Creator.
- Includes free GCC compiler, supports Keil/ARM MDK compiler
- Supports device programming and debugging

Notes

1. The maximum storage temperature is 150 °C in compliance with JEDEC Standard JESD22-A103, High Temperature Storage Life.
2. This feature on select devices only. See [Ordering Information](#) on page 127 for details.

Questions :

1. Quelle est la taille du bus de données du processeurs PSoC 5[LP]
2. Quelle est la taille du bus d'adresse ?

1.2.3 Installations préliminaires

En avril 2018, PSoC Creator 4.2 est la dernière version officielle de l'outil de développement. Elle est téléchargeable gratuitement sur le site de cypress moyennant l'enregistrement d'un compte cypress à l'adresse :

<http://www.cypress.com/products/psoc-creator-integrated-design-environment-ide>

L'outil n'est disponible que pour Windows. Afin de faciliter la prise en charge du kit CY8CKIT-059, il est conseillé de télécharger et d'installer les outils et exemples spécifiques au kit sur la page suivante :

<http://www.cypress.com/documentation/development-kitsboards/cy8ckit-059-psoc-5lp-prototyping-kit-onboard-programmer-and>

Nous pouvons alors démarrer l'utilisation de PSoC Creator et démarrer un nouveau projet avec pour cible le kit CY8CKIT-059

1.2.4 Creation d'un premier projet : blinking led

1. Présentation Dans cette partie nous allons voir comment procéder pour créer un projet sur PSoC Creator pour un type de PSoC donné. Pour cela lancer PSoC Creator puis aller dans le menu File → New → Project...

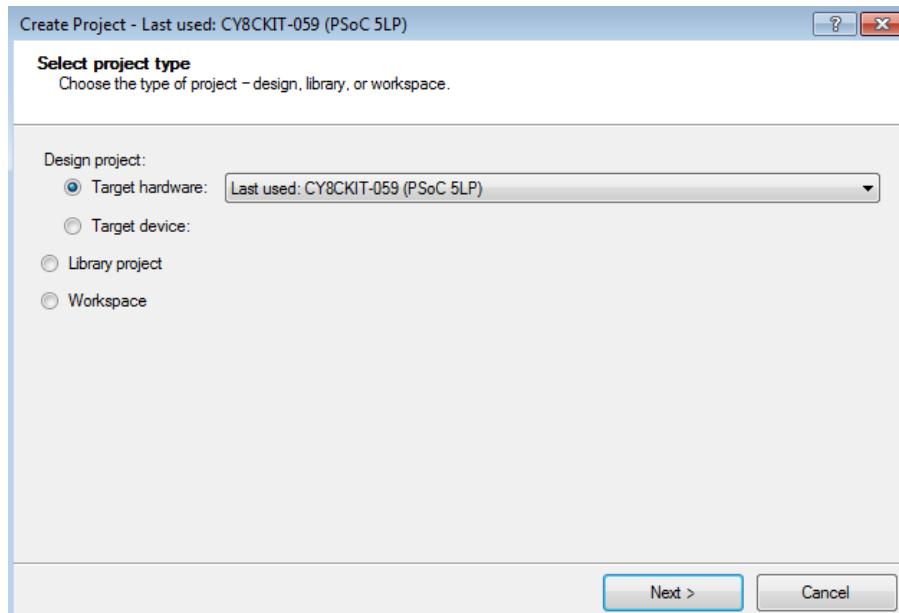


FIGURE 11 – Création du projet PSoC Creator

Sélectionner dans l'onglet **Target hardware** : CY8CKIT-059 puis cliquer sur **Next**. Dans la fenêtre suivante choisir **Empty schematic** puis cliquer à nouveau sur **Next**. Dans la fenêtre suivante, il faut nommer :

- le **Workspace** (espace de travail) qui est un agrégateur de projets : on le nommera **Workspace**

— le Project (projet) que l'on appellera : **blink**

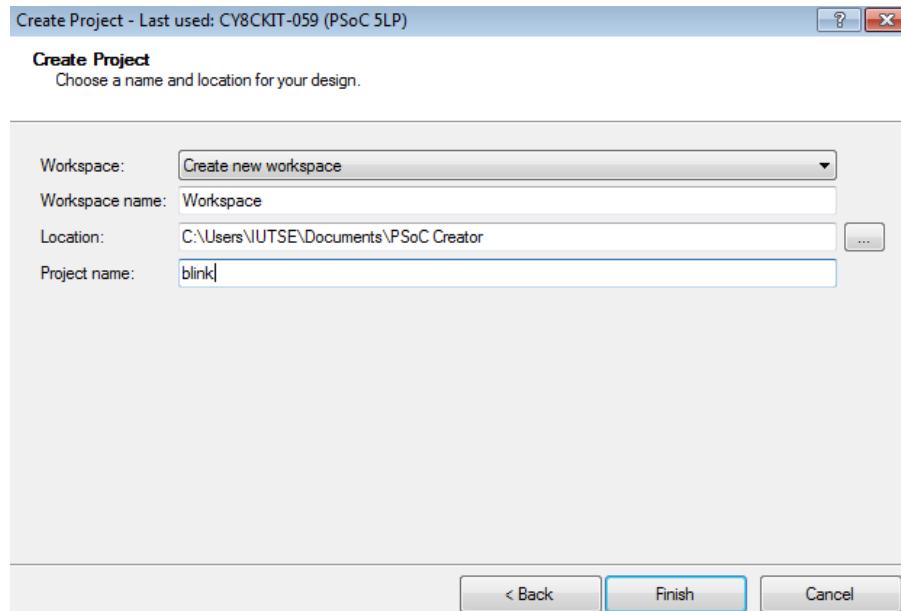


FIGURE 12 – Crédit de la création du projet PSoC Creator

Attention : on évitera les caractères non alphanumériques (excepté l'underscore caractère `<_>`) et on évitera de commencer le nom par un chiffre

On aboutit à l'écran ci-dessous :

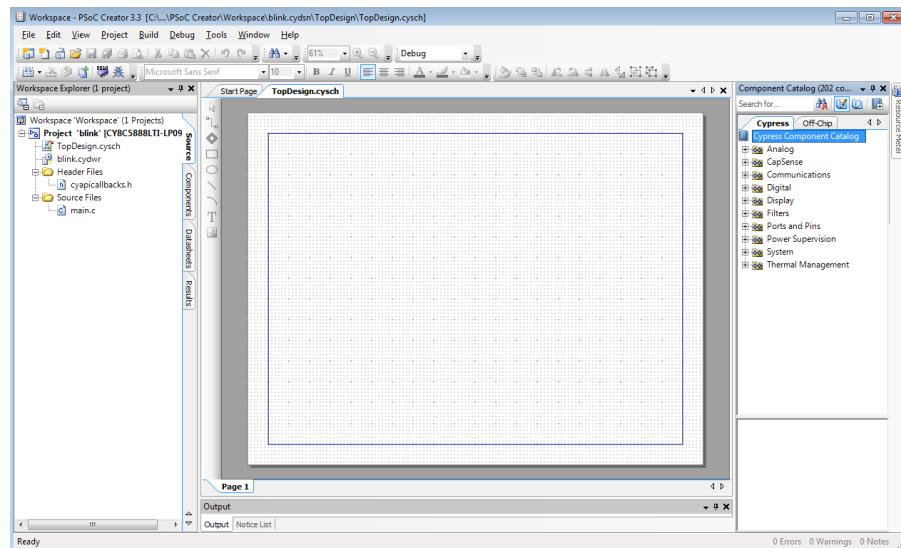


FIGURE 13 – Crédit de la création du projet PSoC Creator

On retrouve sur la partie gauche l'arborescence du Workspace et du Projet. Dans l'onglet vertical Datasheet, on retrouve la datasheet de la famille du composant ainsi que le TRM (Technical Reference Manual). Au centre se situe la partie schématique, il s'agit en fait des blocs matériels utilisés pour le projet. Comme aucun bloc ni périphérique n'ont été utilisés, cette partie est vierge. Sur la partie droite, on retrouve la bibliothèque de composants par défaut.

2. Projet Blinking Led Nous allons créer un projet où l'on souhaite faire clignoter la LED du kit CY8CKIT-059 à la fréquence de 1Hz.

Pour cela, aller dans le **Cypress Component Catalog**, situé sur la partie droite. On y retrouve 2 onglets :

- **Cypress** : qui contient les périphériques et composants disponibles pour le projet actuel INTERNES au PSoC.
- **Off-Chip** : qui contient des schémas de composants pouvant être utilisés dans notre projet mais externes à la puce (chip) PSoC .

Aller dans l'onglet **Cypress** : On retrouve les composants classés par famille. Nous avons besoin :

- d'une broche de sortie numérique
- d'un timer

Réalisation :

- Aller dans le menu **Ports and Pins** → **Digital Output Pin**.
- Faire un clic gauche puis faire glisser le composant dans la partie schéma de PSoC Creator. On observe que le composant Timer possède 3 entrées et 2 sorties.
- Aller dans le menu **Digital** → **PWM**. Faire un clic gauche puis faire glisser le composant dans la partie schéma de PSoC Creator.
- Supprimer le composant **BUSCLK** qui a été placé en entrée du bloc PWM
- Aller dans le menu **System** → **Clock**, faîtes glisser le composant **clock** (horloge) dans la partie schéma de PSoC Creator.

Nous allons relier la sortie **pwm** du composant **PWM** à la broche de sortie **led**. Pour cela sélectionner l'outil fil (wire), cliquer une première fois sur la sortie **pwm** du composant **PWM**, un fil rose apparaît, recliquer lorsque le pointeur de la souris est au niveau de la broche de sortie **led**. Faire de même avec le composant **Clock** que vous relierez à l'entrée **clock** du composant **PWM**.

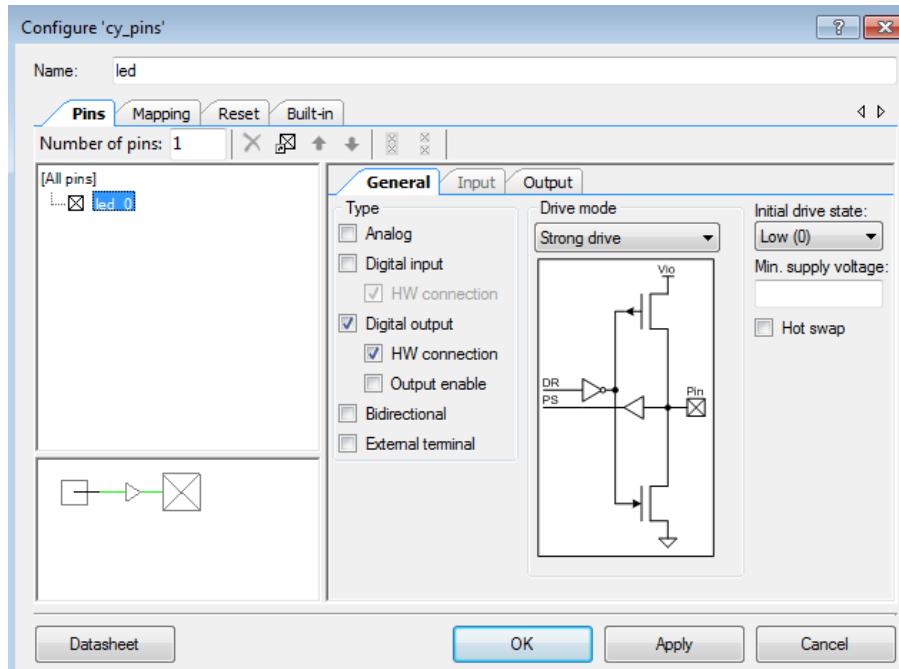


FIGURE 14 – Création du projet PSoC Creator

Configuration des composants Tous les périphériques et composants disponibles pour une puce (chip) donnée dans PSoC Creator ont une Datasheet complète disponible. Il est possible d'y accéder soit :

- en effectuant un click droit sur le composant dans la zone de schéma puis sélectionner **Open Datasheet...**

- en double cliquant sur le composant dans la zone de schéma puis en bas à sélectionner **Datasheet**.

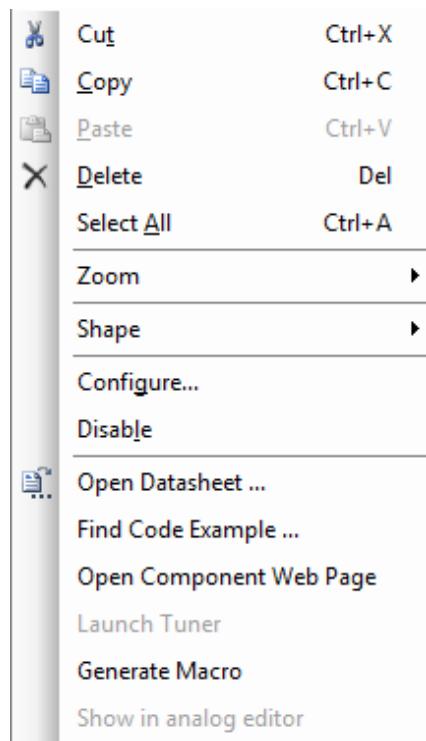


FIGURE 15 – How to open Datasheet

La datasheet s'ouvre dans le logiciel de lecture de pdf (acrobat reader par défaut). On retrouve une description du composant, les différentes API (*A*pplication *P*rogramming *I*nterface) qui sont des fonctions plus ou moins complexes déjà déclarées et définies dans le logiciel PSoC Creator et qui permettent d'éviter de manipuler les registres directement. Pour configurer les composants dans PSoC Creator, il suffit de double-cliquer sur l'un d'eux dans la partie schéma. Chaque composant possède une interface de configuration qui lui est propre. La configuration d'un composant réalise de manière transparente l'écriture dans les registres associés du microcontrôleur. Pour de plus amples informations il faut regarder le TRM (Technical Reference Manual) associé au chip.

Configuration du composant Clock :

- Double cliquer sur le composant Clock₁
- Dans la zone de texte Name, indiquer **Clock**.

ATTENTION PSoC Creator est sensible à la casse des caractères.

- Dans la zone de texte **Period** indiquer **256 Hz**

Configuration du composant PWM :

- Double cliquer sur le composant PWM₁
- Dans la zone de texte associée au nom, actu. ATTENTION PSoC Creator est sensible à la casse des caractères.

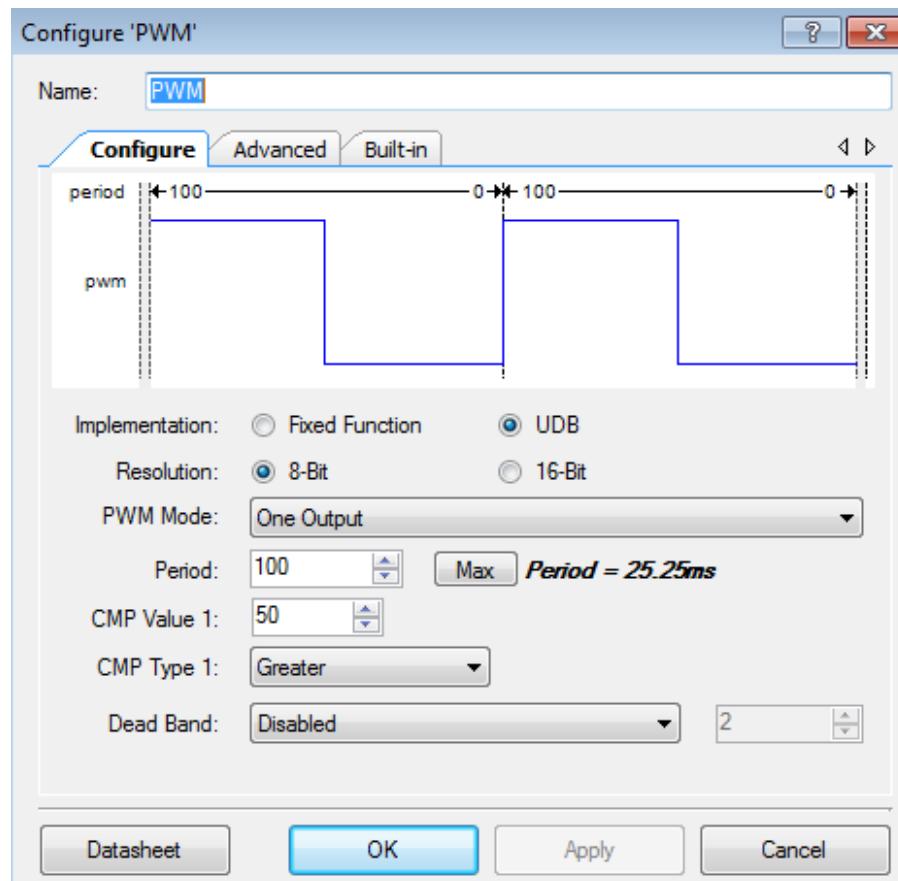


FIGURE 16 – Configuration du composant PWM

- actuellement PWM_1 , renommer celui-ci PWM. Le nom du composant est très important car c'est lui qui sera utilisé avec les API dans le code de l'application.
- Il nous faut ensuite définir la période du composant pwm (period) qui d'après la datasheet, correspond au nombre de cycles d'horloges s'écoulant lors d'une période du PWM (un créneau). Ainsi : $TPWM = \text{period} \cdot Tclock$ avec $Tclock$ la période du signal à la sortie pwm du PWM et $Tclock$ la période de l'horloge sur l'entrée clock du PWM.
- Dans l'onglet Advanced, choisir Software. Cela permet d'activer le composant en utilisant l'API.
- Sélectionner la broche de sortie Pin, et renommer la broche de sortie Pin_1 en led.

La partie configuration de notre projet est terminée.

(a) Les paramètres généraux

Dans l'onglet de l'explorateur de projet sur la fenêtre de gauche (Workspace Explorer) double cliquer sur blink.cydwr. Il s'agit de l'espace de configuration des ressources. Au bas de l'écran, on voit apparaître un bandeau avec plusieurs onglets :

Le premier onglet Pins permet de réaliser le routage physique (mapping) des broches du composant PSoC vers l'application. C'est ici que nous allons indiquer le port et la ligne du port que nous souhaitons utiliser pour la sortie digitale que nous avons appelé led. Afin que cette sortie soit reliée à une led physique nous utiliserons pour des soucis de simplicité la ligne de port P2[1] qui est déjà relié à une led bleue présente sur le kit CY8CKIT-059 :

- l'onglet **Clocks** (horloges) permet de choisir la configuration des horloges déjà présentes dans le composant PSoC.
- l'onglet **Interrupts** permet de gérer les interruptions matérielles, leur priorité.
- l'onglet **DMA** permet la gestion des accès Direct Memory Access.

- l'onglet **System** contient les paramètres globaux d'alimentation et de gestion de mémoire de notre projet.

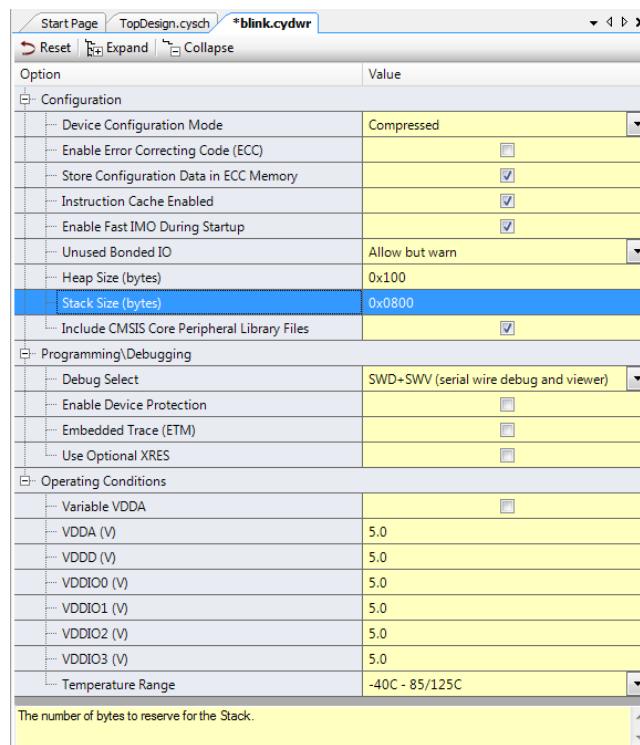


FIGURE 17 – Onglets du Design Wide Ressources

Au sein de l'onglet System, on retrouve 3 listes déroulantes :

- Configuration : on y trouve la gestion des codes correcteurs d'erreurs pour la mémoire FLASH ainsi que la la **taille de la pile (Stack Size)** et la **taille du tas (Heap Size)**
- Programming/Debugging : dans cette partie, le mode de programmation est choisi. On laisse les valeurs par défaut.
- Operating Conditions : contient les informations sur les tensions d'alimentation, des bus numériques et analogiques ainsi que la gamme de température de fonctionnement.

RAPPEL SUR L'ALLOCATION MEMOIRE : 3 stratégies d'allocation mémoire vive (RAM) peuvent être employées :

- allocation statique : Elle s'effectue avant l'exécution du programme, au moment de la création du programme. L'espace mémoire alloué statiquement est déjà réservé dans le fichier exécutable du programme lorsque le système charge le programme en mémoire pour l'exécuter.

Lorsque de la mémoire est alloué de manière statique,

- allocation dynamique sur la pile : elle s'effectue automatiquement L'exécution d'un programme utilise généralement une pile contenant les cadres d'appel aux routines (fonctions) du langage de programmation utilisé. Schématiquement, les variables lexicales, c'est-à-dire les variables définies dans la portée textuelle d'une routine, sont :
 - allouées lors de l'entrée dans la routine, c'est-à-dire que l'espace est réservé pour ces variables ;
 - désallouées automatiquement lors de la sortie (du retour) de la routine, c'est-à-dire que l'espace réservé pour ces variables est dorénavant libre et disponible pour d'autres variables.

- allocation dynamique sur le tas : La plupart des programmes ayant des besoins en mémoire dépendant de l'usage qu'on en fait, il est nécessaire de pouvoir, à des moments arbitraires de l'exécution, demander au système l'allocation de nouvelles zones de mémoire, et de pouvoir restituer au système ces zones (libérer la mémoire). Dans ce cas, l'allocation et la libération de la mémoire sont sous la responsabilité du programmeur. Les fuites de mémoire, ainsi que d'autres erreurs fréquentes dans les programmes à gestion manuelle de la mémoire, ont leur source dans les erreurs d'allocation mémoire sur le tas.

Classiquement, les fonctions de la bibliothèque standard de C malloc et free, les opérateurs du langage C++ new et delete permettent, respectivement, d'allouer et de libérer la mémoire sur le tas. On retiendra que :

- Toute variable globale (et variable locale déclarée "static") est stockée dans le segment de données
- Toute variable locale à une fonction (non "static") est stockée dans la pile (stack).
- Toute zone mémoire allouée par new est stockée dans le tas (heap), zone située après le segment de données.

Question : Combien d'octets sont alloués à la pile (stack) dans la figure précédente ? Combien d'octets sont alloués au tas (heap) ?

Le tas est utilisé par certaines fonctions telles que sprintf(), ainsi que par des applications qui utilisent des blocs de mémoire à l'aide de malloc() et ses API connexes. Si le projet n'utilise pas d'allocation dynamique de mémoire, de l'espace mémoire est simplement gaspillé pour le tas. Il est alors possible de le supprimer en changeant la taille du tas à zéro octet.

La pile est une zone mémoire utilisée lors de l'appel de fonctions. Lorsqu'une fonction est appelée, la pile est utilisé pour l'adresse de retour, pour transmettre des arguments à la fonction appelée, et pour les valeurs de retour. La quantité de mémoire nécessaire pour la pile dépend de la profondeur d'imbriication de l'application est du nombre (et le type) des paramètres transmis à chaque niveau. Tout comme le tas, la taille de la pile se trouve dans l'onglet Système de l'éditeur de ressources (Design Wide resources).

Il existe plusieurs façons de déterminer la bonne quantité d'utilisation de la pile pour une application. La première méthode consiste à examiner le code source. Déterminer les fonctions qui utilisent beaucoup d'espace de pile et où la plus profonde imbrication se produit. A partir de là, il est possible de calculer l'utilisation maximale de la pile.

Il est également possible d'utiliser les lignes de commande fournies avec le compilateur pour obtenir l'utilisation de la pile sur une base par fonction. Par exemple, si vous utilisez GCC compiler, vous pouvez ajouter -fstack-utilisation aux commandes pour générer un fichier supplémentaire avec une extension .swf, qui spécifie la quantité maximale de pile utilisée par chaque fonction. Pour plus d'informations, voir https://gcc.gnu.org/onlinedocs/gnat_ugn/Static-Stack-Usage-Analysis.html.

En phase de développement, il est judicieux d'ajouter un peu plus au maximum calculée de telle sorte d'avoir encore de l'espace pour ajouter une petite quantité de code à l'application sans avoir à changer la taille de la pile.

La seconde méthode est plus empirique. Elle consiste à utiliser le débogueur pour surveiller la mémoire utilisée. Tout d'abord, vous remplissez la pile avec des données inhabituelles. Un nombre arbitraire, comme 0xFEEDBEEF fonctionne bien parce qu'il se remarque bien visuellement, et la chance de votre application à écrire de manière récurrente de ce numéro sur la pile est faible. Ensuite, on exécute simplement l'application et visualise la fenêtre de mémoire pour voir combien de la pile n'a pas été remplacée. Il est important d'exécuter l'application intégralement pour s'assurer que tous les chemins ont été suivis et l'utilisation maximale de la pile est produite.

Une alternative à cette méthode consiste à placer une lecture / écriture (accès) breakpoint

au bas de la pile et voir si elle est toujours atteinte. Si elle est atteinte alors vous avez besoin de plus d'espace de pile. Sinon, il suffit de déplacer le breakpoint vers le haut de la pile jusqu'à ce qu'il soit atteint et on obtient alors l'utilisation optimale.

Enfin, certains outils tiers et IDEs offrent intégré, les fonctions stack-checking, qui peuvent être particulièrement utile lorsque vous utilisez un système d'exploitation en temps réel (RTOS). Dans ces environnements, chaque tâche a généralement sa propre pile dédiée.

Remarque importante Dans le cas de l'utilisation d'un écran LCD 2 lignes de 16 caractères, il faut mettre le heap à 0x0100 car les API du composant LCD en ont besoin dans le cas de l'affichage de variable de type float sur celui-ci.

1.2.5 Compilation, Programmation, Debuggage

La conception de notre projet est terminée, il est nécessaire de le compiler. Cette opération s'effectue grâce à l'icône : 

Cette opération build, réalise la génération du bitstream pour la partie logique programmable, l'édition de liens ainsi que la compilation du code C. Lors de la compilation, on peut observer qu'un certain nombre de fichiers ont été créés.



La programmation du PSoC s'effectue grâce à l'icône :  L'interface de programmation est SWD ou JTAG.



Le débogage s'effectue grâce à l'icône :  Il est possible de visualiser les variables globales. Dans le cas du kit CY8CKIT-059, le programmeur et le debugger est in situ c'est à dire que la puce CY8C5868LTI-LP039 joue ces 2 rôles. Elle est nommée le KitProg.

1.3 Les entrées/sorties GPIO

1.3.1 Présentation

Les broches entrées/sorties sont communément appelées GPIO (*General Purpose Input Output*) ou I/O (*Input Output*). Elles permettent l'interaction entre le monde extérieur et l'environnement PSoC.

Celles-ci peuvent être analogiques ou numériques, configurées, en tant qu'entrée, sortie, ou même bidirectionnelles. Leur interface externe est constituée d'une porte inverseuse CMOS : elle utilise 2 transistors MOS complémentaires. Une entrée/sortie est un <push-pull> avec :

- un transistor MOS canal P actif si la sortie est à l'état 1 : celui conduit lorsque la tension appliquée entre sa grille et sa source (c'est-à-dire Vdd-Ve) est supérieure à 2,5 V.
- un transistor MOS canal N actif si la sortie est à l'état 0 : celui-ci conduit lorsque la tension appliquée entre sa grille et sa source (c'est-à-dire Ve) est supérieure à 2,5 V.

On a donc toujours en régime statique, un transistor bloqué et un transistor passant. Si la sortie est surchargée, la résistance interne limite le courant mais l'échauffement peut être destructeur. Afin d'augmenter le courant de sortie, il est possible dans certains cas de connecter plusieurs sorties en parallèle et de leur assigner le même état logique.

La technologie utilisée est de type CMOS ce qui signifie que les transistors constituants ces portes sont de type MOS canal P pour celui High Side (en haut) et MOS Canal N pour celui Down Side (en bas). Cette technologie présente l'avantage de ne consommer quasiment aucun courant tant que la porte est à l'état de repos.

Schéma bloc d'une broche GPIO

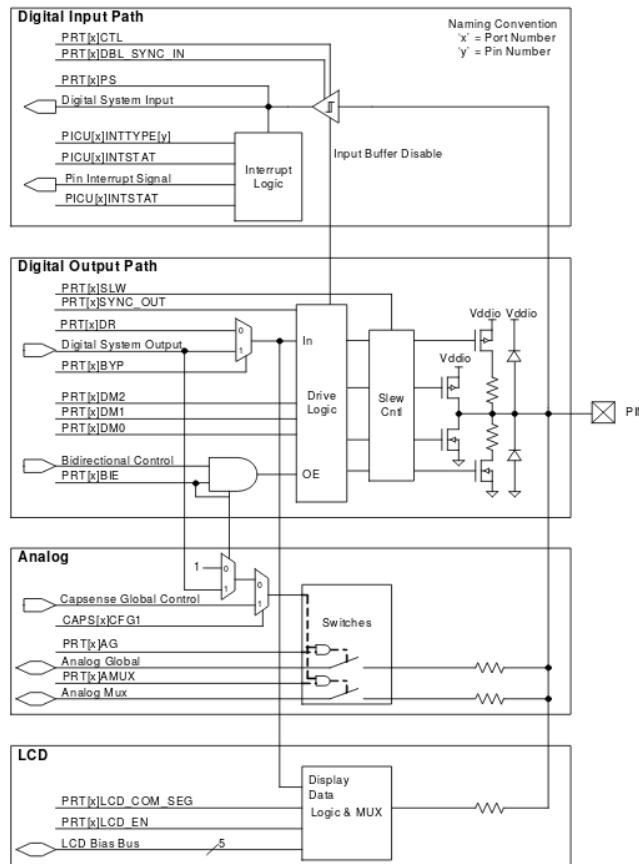


FIGURE 18 – Schéma bloc d'une broche GPIO

L'intégralité des différentes broches IO (GPIO, SIO, USBIO) est représenté pages 152 et suivantes du TRM (*Technical Reference Manual*).

Quel est le courant maximal que peut fournir une broche de sortie du PSoC 5[LP] CY8C58LP ?

1.3.2 Rappels

1. Résistances

- **pull-down** : Une résistance de pull-down impose une tension proche du zéro et évite que la ligne soit flottante, donc avec une valeur logique mal définie.
- **pull-up** : Une résistance de pull-up impose une tension proche de $+V_{cc}$. Les mirr-contrôleurs ont souvent des pull-ups programmables
- **diviseur de tension** : Un dviseur définit une tension intermédiaire.
- **résistance série** : Une résistance série crée une chute de tension proportionnelle au courant donc limite ce courant.

2. Condensateurs

Une capacité de découplage réduit les pointes négatives de tension lors de la commutation des signaux. On met souvent 100 nF sur l'alimentation à proximité du processeur.

3. Diodes et diodes de protection

Une diode ne laisse passer le courant que dans un sens. Comme on peut le voir sur la figure fig :GPIO_{PSOC5LP}, les sorties sont protégées par des diodes. Si la tension est de 6V et que l'on applique 6V sur une entrée, la diode du haut va conduire, chauffer et se détruire si le courant est important. Si l'alimentation du microcontrôleur est à l'envers, ces diodes conduisent et le circuit chauffe. Si l'on coupe le courant à temps, ce n'est pas destructeur.

4. Entrée en Bascule de Schmitt (Schmitt-trigger)

La bascule de Schmitt présente un hystérésis qui fait que les petites fluctuations du signal ne passent pas. C'est l'idéal pour toute ligne connectée à un signal extérieur.

5. Entrée non connectée

Une entrée "en l'air" (non connectée) a une tension mal définie. Il suffit d'approcher la main pour capter des perturbations électromagnétiques (couplage capacitif appelé effet de main) qui font changer l'état logique et si la main ou le fer à souder possède une tension électrostatique élevée, l'entrée peut être détruite. Les bracelets antistatiques et les fils à la masse sont indispensables dans tous les pays chauds et secs où la tension d'ionisation de l'air est relativement basse.

6. Etage de sortie

Une sortie normale est en push-pull avec un transistor MOS canal P actif si la sortie est à l'état 1 et un transistor MOS canal N actif pour l'état 0. Si la sortie est surchargée, la résistance interne d'un transistor R_{DSon} (très faible : quelques $m\Omega$) limite le courant qui provoque un échauffement pouvant être destructeur.

7. Collecteur ouvert

Le transistor MOS canal P manque en général sur une ligne de programmation de microcontrôleur. Dans ce cas, on peut écrire un **0**, mais écrire un **1** laisse la ligne dans un état flottant, comme pour une entrée. Pour imposer l'état logique **1**, il est nécessaire d'ajouter une résistance de pull-up.

1.3.3 Modes de pilotage des broches

Suivant les liaisons extérieures et les organes pilotées ou les types des signaux acquis sur les broches, on utilise un < mode de pilotage > pour celles-ci. Il existe 8 modes de pilotages, représentés ci-dessous que l'on peut retrouver dans la doc des broches en effectuant un click droit sur l'une d'entre elles dans la partie TopDesign (schéma) et en sélectionnant la Datasheet.

Passons en revue ces différents modes de pilotage : Les modes de pilotage des GPIO sont :

- High Impedance Analog (HZ) : mode haute impédance pour des signaux analogiques : impédance d'entrée ou de sortie infinie.
- High Impedance Digital(HZ) : mode haute impédance pour des signaux numériques : impédance d'entrée ou sortie infinie.
- Resistive Pull Up : Une résistance est insérée entre le transistor MOS canal P du haut et la sortie.
- Resistive Pull Down : Une résistance est insérée entre le transistor MOS canal N du bas et la sortie.
- Open Drain, Drives Low : Seul le transistor MOS canal N est commandé. S'il est passant la sortie est à 0. S'il ne l'est pas la sortie est en HZ.

- Open Drain, Drives High : Seule le transistor MOS canal P est commandé. S'il est passant la sortie est à 1. Sinon la sortie est en HZ.
- Strong Drive : c'est le cas où la sortie est directement reliée soit à 0 soit à 1 sans résistance de limitation de courant (Pull-up ou Pull-down).
- Resistive Pull Up & Down

Questions :

- Quel doit être le mode de commande à utiliser pour avoir le courant maximal en sortie sur une broche ?
- Quel doit être le mode commande à utiliser sur une broche d'entrée si celle-ci est reliée à un capteur dont la grandeur de sortie est une tension afin de ne pas consommer de courant ?

1.4 Les interruptions

1.4.1 Introduction

Les interruptions sont d'une importance capitale lors du développement d'un système embarqué à microprocesseur. Elles permettent : Les interruptions sont une partie importante de toute application embarquée. Elles libèrent le CPU d'avoir à interroger en continu de la survenue d'un événement spécifique et au lieu, aviser le CPU uniquement lorsque cet événement se produit. Dans les architectures de systèmes sur puces (SoC : System-On-Chip) telles que les PSoC, les interruptions sont très fréquemment utilisées pour communiquer l'état des périphériques internes au processeur (CPU). C'est grâce aux interruptions matérielles, que des contraintes dites "temps-réels" peuvent être intégrées sur des systèmes informatiques.

1.4.2 Architecture du mécanisme d'interruptions sur PSoC 5LP

Les PSoC 5[LP] sont dotés de 32 lignes d'interruptions : int[0] à int[31]. Chaque ligne d'interruption peut se voir assigner un des 8 niveaux de priorité (de 0 à 7), où 0 correspond à la plus haute priorité.

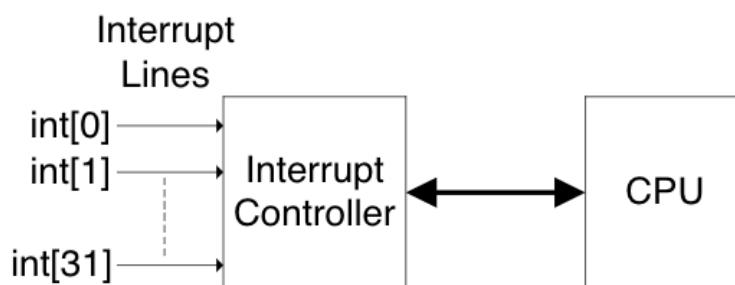


FIGURE 19 – Lignes d'interruption sur PSoC 5[LP]

Chaque ligne d'interruption est assignée à une adresse de vecteur d'interruption qui se réfère à l'adresse de départ du code d'interruption. Le processeur se connecte à cette adresse après avoir reçu un code d'interruption qui est dénommé ISR (Interrupt Service Routine).

Le contrôleur d'interruption assure l'interface entre les lignes d'interruptions et le CPU. Il envoie l'adresse du vecteur d'interruption d'une ligne d'interruption au CPU avec le signal de demande d'interruption. Le contrôleur d'interruption reçoit les signaux d'accusé de réception de la part

du CPU sur les conditions d'entrées et de sortie d'interruption. Le contrôleur d'interruption gère la priorité des interruptions dans le cas de demandes sur des lignes d'interruption multiples.

1.4.3 Caractéristiques

Caractéristiques des interruptions dans le PSoC 5[LP]

L'adresse du vecteur d'interruption est configurable : Avec les PSoC il est possible de configurer dynamiquement l'adresse du vecteur d'interruption. Le programme exécuté par le processeur peut se connecter directement à n'importe quel code de programme d'interruption (ISR = Interrupt Service Routine). Cela permet au PSoC d'avoir une latence d'exécution d'interruption réduite par rapport aux micro-contrôleurs traditionnels. Pour changer l'adresse du vecteur d'interruption, il suffit d'utiliser l'API SetVector() associée à l'interruption.

Les sources d'interruption sont flexibles : Dans les micro-contrôleurs traditionnels, la source d'interruption est routée physiquement à chaque ligne d'interruption. Les PSoC apportent la flexibilité en permettant de choisir la source d'interruption pour chaque ligne d'interruption. La flexibilité de l'architecture permet à n'importe quel signal numérique d'être configuré comme une source d'interruption.

Interruption à déclenchement sur front, interruption à déclenchement sur niveau
 Les PSoCs supportent aussi bien les interruptions à déclenchement sur niveau (level triggered interrupts) que les interruptions à déclenchement sur front (edge triggered interrupts). La classification d'une interruption en <interruption à déclenchement sur front> ou en <interruption à déclenchement sur niveau> est basé sur le signal d'interruption généré par la source d'interruption.

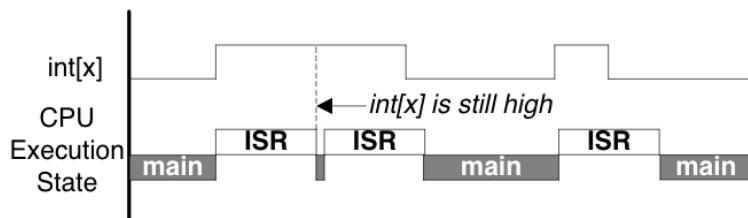


FIGURE 20 – Interruption à déclenchement sur niveau - trigger-level interrupt

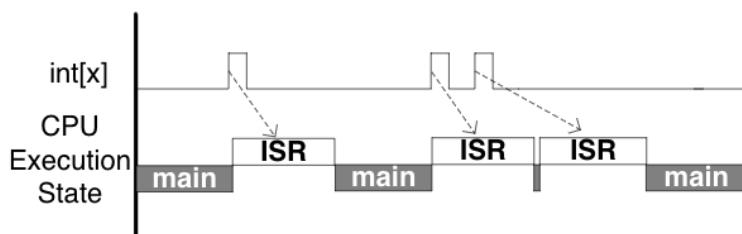


FIGURE 21 – Interruption à déclenchement sur font - edge-level interrupt

1.4.4 Mise en oeuvre des interruptions

La mise en oeuvre d'une interruption sur le PSoC peut s'effectuer de 2 manières :

- en utilisant le composant **isr** que l'on retrouve dans la bibliothèque standard des composants dans le dossier **System**. Le composant **isr** peut être connecté directement (généralement sur une broche de sortie) du périphérique considéré de la fenêtre TopDesign. Il est alors nécessaire de reconstruire le projet avec l'icône build afin que les fichiers *isr.c* et *isr.h* (si le nom de l'interruption dans la fenêtre TopDesign est *isr*) soient créés. Le code inséré dans ces fichiers doit respecter certaines règles sous peine de disparaître lors de la compilation. En effet PSoC Creator reconstruit ces fichiers à chaque **build**.



Le code ne peut être inséré qu'entre les balises :

```
/* `#START isr_intc` */
/* `#END` */
```

exemple :

```
*****  
* Place your includes, defines and code here  
*****  
/* `#START isr_intc` */  
#include <device.h>  
extern uint8 period;  
extern uint8 compare;  
extern uint8 i;  
/* `#END` */  
CY_ISR(isr_Interrupt)  
{  
    #ifdef ISR_INTERRUPT_INTERRUPT_CALLBACK  
        isr_Interrupt_InterruptCallback();  
    #endif /* ISR_INTERRUPT_INTERRUPT_CALLBACK */  
    /* Place your Interrupt code here. */  
    /* `#START isr_Interrupt` */  
    if (i>=10){  
        if (compare < period)  
        {  
            compare++;  
        }  
        else compare =0;  
        i=0;  
    }  
    else i++;  
    PWM_1_WritePeriod(period);  
    PWM_1_WriteCompare(period/2);  
    //PWM_2_WritePeriod(period);  
    PWM_2_WriteCompare(compare);  
    /* `#END` */  
}
```

- en créant sa propre routine d'interruption Afin de rendre plus facilement exportable et polyvalent le code développé sous PSoC Creator en C, il peut être judicieux de concentrer les routines d'interruptions dans un seul est même fichier. Les étapes permettant de créer sa propre fonction, par exemple, *MyCustomISR*, c'est à dire, la routine d'interruption associée au composant *isr*, sont :
 - Déclarer la fonction utilisant la macro *CY_ISR_PROTO* :
 - CY_ISR_PROTO(MyCustomISR)*
 - Définir la fonction utilisant la macro *CY_ISR* :
 - Dans le code de démarrage du fichier *main.c*, ajouter un appel à la fonction de l'API *isr_Start_Ex()*. Cette fonction est similaire à la fonction de l'API *isr_Start()* sauf que *isr_Start_Ex()* prend pour paramètre la fonction ISR créée : *isr_Start_Ex(MyCustomISR)*. La fonction *isr_Start_Ex()* définit l'adresse du vecteur d'interruption de la fonction ISR.

Exemple :

fichier *main.c*

```
#include <project.h>

/* Header file containing the custom ISR prototypes */
#include "InterruptRoutines.h"
void main()
{
    /* Initialize the two custom defined ISRs */
    isr_1_StartEx(PICU_ISR) ;
    isr_2_StartEx(Tick_ISR) ;

    CyGlobalIntEnable ; /* Enable global interrupts. */

    for( ; ;)
    {
        /* Do nothing in the main loop ; code to do something
           is in the ISRs */
    }
}
```

fichier *InterruptRoutines.h*

```
#ifndef INTERRUPT_ROUTINES_HEADER
#define INTERRUPT_ROUTINES_HEADER

/* including project.h gives access to all component APIs and other generated source files */
#include <project.h>

/* defines the debounce time in milliseconds, max value is 255 */
#define DEBOUNCE_TIME 50

/* ISR function prototype declarations */
CY_ISR_PROTO(Tick_ISR) ;
CY_ISR_PROTO(PICU_ISR) ;

#endif
```

fichier *InterruptRoutines.c*

```
#include "interruptRoutines.h"

/* register bit masks for the switches */
#define SW1_MASK 1
#define SW2_MASK 2
#define TIMED_OUT 0

// switch debounce timeout variables decremented by Tick_ISR(), and monitored and reset by PICU_ISR()
static volatile uint8 CYDATA switch_1_timeout;
static volatile uint8 CYDATA switch_2_timeout;

/********************* FUNCTION NAME : Tick_ISR *****************/
FUNCTION NAME : Tick_ISR
Summary : Interrupt once per millisecond. If timeout, toggle LED.
Parameters : none; it's an ISR function
Return : none; it's an ISR function
/********************* CY_ISR(Tick_ISR) *********************/
CY_ISR(Tick_ISR)
{
    if(switch_1_timeout != 0)
    {
        // Come here if an edge was detected by the PICU. Decrement and check timeout.
        if(--switch_1_timeout == TIMED_OUT)
        {
            // Timed out. Toggle LED only on a switch press (= 0).
            if((Switches_Read() & SW1_MASK) == 0)
            {
                LED1_Write(~LED1_Read()); // toggle the LED
            }
        }
    }
    /* repeat for switch 2 */
    if(switch_2_timeout != 0)
    {
        if(--switch_2_timeout == TIMED_OUT)
        {
            if((Switches_Read() & SW2_MASK) == 0)
            {
                LED2_Write(~LED2_Read()); /* toggle the LED */
            }
        }
    }
}

/********************* FUNCTION NAME : PICU_ISR *****************/
FUNCTION NAME : PICU_ISR
Summary : Interrupt only occurs on falling edge of either pin, i.e., when
switch is pressed. Checks and initiates debounce timeout.
Parameters : none; it's an ISR function
Return : none; it's an ISR function
/********************* CY_ISR(PICU_ISR) *********************/
CY_ISR(PICU_ISR)
{
    /* copies of PICU registers */
    uint8 CYDATA temp_stat;

    // read the PICU interrupt status register, with a clear on read
    temp_stat = Switches_ClearInterrupt();

    // Process the PICU event on SW1 only if any ongoing debounce period has timed out
    if(((temp_stat & SW1_MASK) != 0) && (switch_1_timeout == TIMED_OUT))
    {
        // reset the debounce timer for this button
        switch_1_timeout = DEBOUNCE_TIME;
    }

    // Process the PICU event on SW2 only if any ongoing debounce period has timed out
    if(((temp_stat & SW2_MASK) != 0) && (switch_2_timeout == TIMED_OUT))
    {
        /* reset the debounce timer for this button */
        switch_2_timeout = DEBOUNCE_TIME;
    }
}
/* [] END OF FILE */
```

1.4.5 Communication entre le programme principal et l'ISR

La communication entre la fonction d'interruption et le programme principal *main* peut se réaliser de 2 manières :

- utilisation d'un drapeau (*flag*)
- utilisation de variables globales

```
#include <stdio.h>
int main()
{
    printf("Hello\n");
    return 0;
}
```

1.4.6 Gestion de la priorité des interruptions

La fenêtre Design Wide Resources (*project_name.cydwr*) du projet sur PSoC Creator possède un onglet Interrupt.

1.5 DMA : Direct Memory Access

1.5.1 Introduction

Le contrôleur DMA (DMAC = DMA Controller) du PSoC 5LP permet le transfert des données d'une source vers une destination sans intervention du processeur (CPU). Cela permet au CPU de gérer d'autres tâches pendant que le DMA fait le transfert de données, ce qui permet la réalisation d'un environnement multi-tâches/ multi-traitements <temps-réel>. Le DMAC du PSoC est très flexible. Il peut parfaitement transférer des données entre la mémoire et les périphériques de la puce, y compris ADCs, DACs, Filtre, USB, UART et SPI.

1.5.2 Concept de base de DMA

Le DMAC dans le PSoC 5LP est une partie d'un bus central appelé Peripheral HUB (PHUB) qui inter-connecte les périphériques de la puce comme indiqué dans la figure ci-dessous :

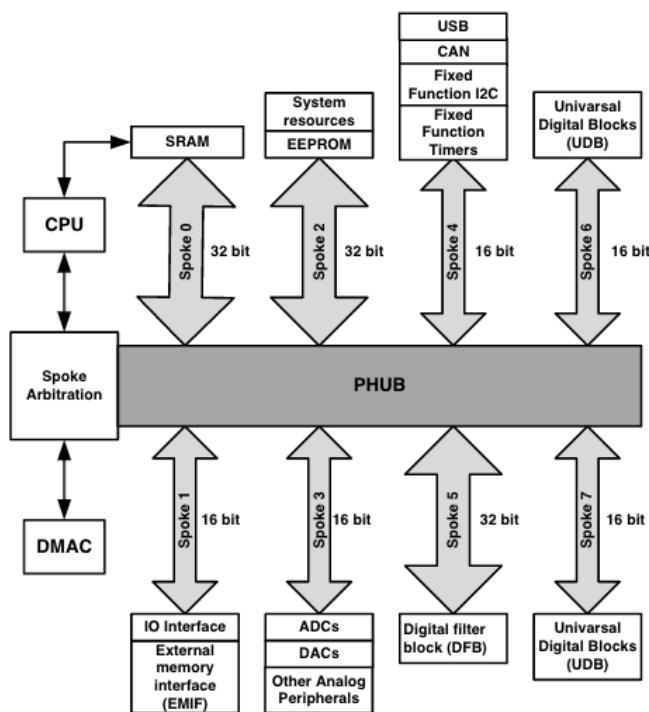


FIGURE 22 – bus central PHUB

Le PHUB dispose de huit bus de données qui sont appelées rayons (spoke en anglais). Chaque rayon relie la CPU et le DMAC une ou plusieurs périphériques. Les rayons peuvent avoir des largeurs soit de 16 bits, ou soit de 32 bits. Les périphériques connectés à un rayon peuvent avoir une largeur de 8 bits, 16 bits ou 32 bits. La taille des données d'un périphérique est généralement inférieure ou égale à la largeur de données du rayon auquel il est attaché. Si la taille de données du périphérique est supérieure à celle du rayon attaché, le PHUB peut effectuer des transactions avec le périphérique au niveau de la largeur du rayon.

Le PHUB possède 2 maîtres de bus :

- le CPU
- le DMAC

La CPU et le DMAC peuvent accéder à différents rayons de PHUB en même temps. Si la CPU et DMAC tentent d'accéder au même rayon en même temps, l'arbitrage de bus se produit. Voir le manuel de référence technique du PSoC 5LP pour plus de détails (Technical Reference Manual). Chacun des 24 canaux DMA peuvent transférer de façon indépendante des données. Chaque canal possède un descripteur de chaîne de transaction (TD), comme le montre la figure ci-dessous :

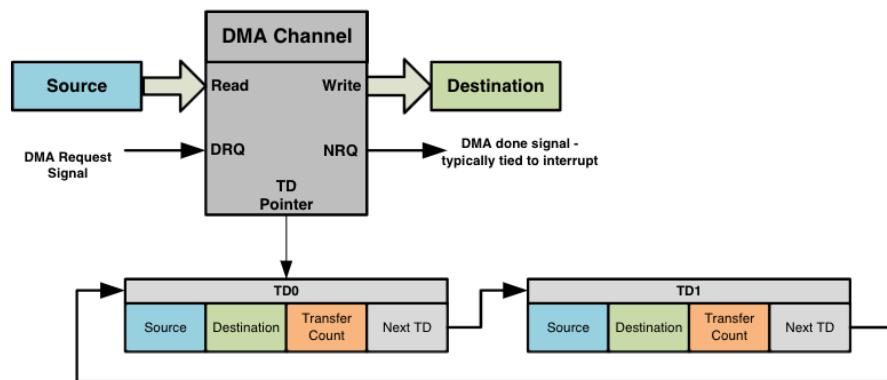


FIGURE 23 –

Le TD contient des informations telles que l'adresse source, adresse de destination, le transfert compteur, et le prochain TD dans la chaîne. Il peut y avoir jusqu'à 128 TDs. La combinaison de canal et TD décrit le transfert complet DMA.

Chaque canal de DMA a une entrée de demande DMA distincte qui initie une transaction. Une demande de DMA peut être initiée par le CPU ou par un périphérique. Lorsqu'une requête DMA est reçue, le DMAC accède aux rayons fixés à la source et à la destination et déplace les données comme configurées dans le canal et le TD associés.

1.5.3 Configuration DMA

Un transfert DMA est configuré en utilisant les registres associé au canal DMA au TD.

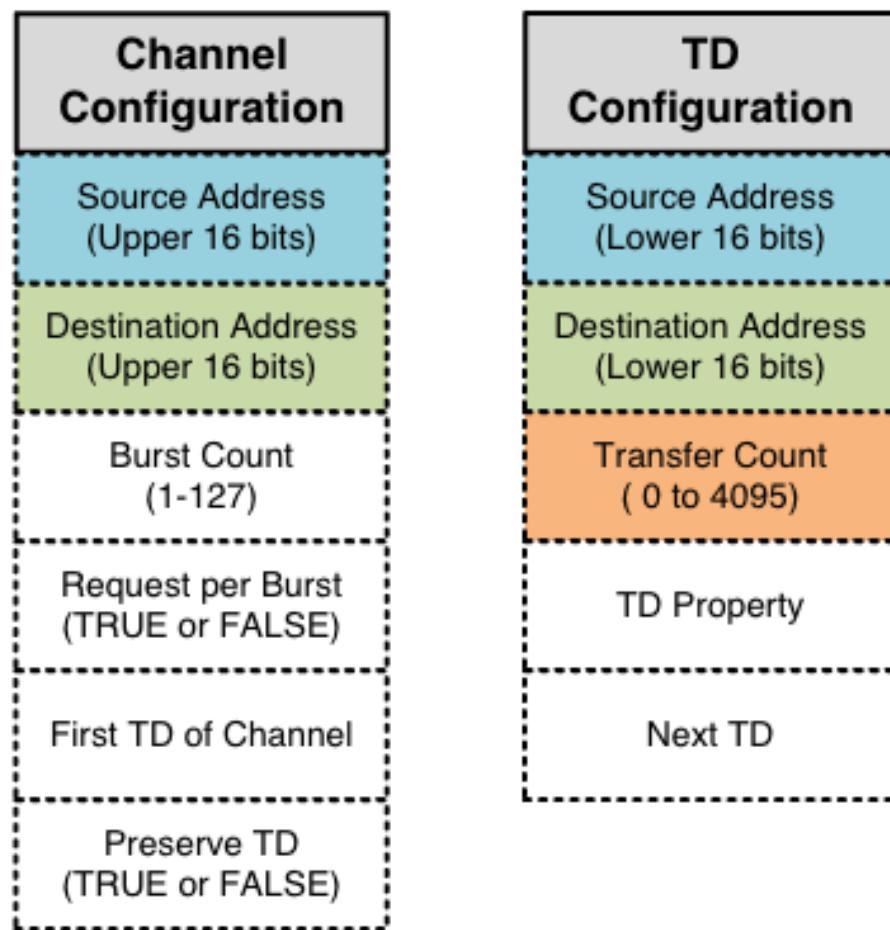


FIGURE 24 –

1.6 Applications

1. Blinking LED sans interruption (classique)
2. Blinking LED avec interruption.
3. Blinking LED avec interruption (avec fichier spécifique pour ISR).
4. Création d'un mini-RTOS

1.7 Creation d'un RTOS

1.7.1 Ecriture d'un RTOS

Objectif : Ecriture d'un petit systeme d'exploitation temps-reel (RTOS = *Real Time Operating System*).

Aspect materiel : Le RTOS sera utilisable sur les cartes PSoC de la famille CY8C58LP.

Algorithme du sequenceur

```

Debut Fonction Sequence
    variable debut <- valeur courante du timer
    Tache_10ms_1() //1ere tache appele toutes les 10 ms
    Tache_10ms_2() //2eme tache appele toutes les 10 ms
    .....
    .....
    Selon(nb_sequences %2) //(avons nous 2 squences ou des multiples de 2 ?)
        cas 0 : Tache_20ms_1()
        cas 1 : Tache_20ms_1()
        ...
    Fin Selon

    Selon(nb_sequences %10) //(avons nous 10 squences ou multiples de 10 ?)
        cas 0 : Tache_100ms_1()
        cas 1 : Tache_100ms_2()
        cas 2 : Tache_100ms_3()
        cas 3 : Tache_100ms_4()
        ...
    Fin Selon

    Si ((nb_sequences % 100)==0) //(avons nous 100 squences ou multiples ?)
        Tache_1s_1()
    Fin Si

```

Il peut tre utile de rappeler que l'operateur modulo % retourne le reste de la division. Dans ce sequenceur, il est donc essentiel dans le fonctionnement. C'est le reste qui definit apres chaque "Selon" le cas a traiter.

Algorithme du programme principal

1.7.2 Ecriture du RTOS

Ecrire l'algorithme du sequenceur gerant :

- 1 fonction toutes les 100 ms
- 2 fonctions toutes les 200 ms
- 2 fonctions toutes les 1000 ms
- 1 fonction toutes les 10 secondes

Dans un premier temps, ne pas ecrire l'algorithme d'ajustement du temps de sequence.

1.8 Bibliographie

2 Systèmes à processeurs hosted : le cas linux

```
import matplotlib, numpy
matplotlib.use('Agg')
import matplotlib.pyplot as plt
fig=plt.figure(figsize=(4,2))
x=numpy.linspace(-15,15)
plt.plot(numpy.sin(x)/x)
fig.tight_layout()
plt.savefig('images/python-matplot-f1.png')
return 'images/python-matplot-f1.png' # return filename to org
```

@@latex @@ nice thing !

2.1 GNU Free Documentation License

Version 1.3, 3 November 2008

Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc.

<<https://fsf.org/>>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document “free” in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of “copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The “**Document**”, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as “**you**”. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A “**Modified Version**” of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A “**Secondary Section**” is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The “**Invariant Sections**” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The “**Cover Texts**” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A “**Transparent**” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not “Transparent” is called “**Opaque**”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The “**Title Page**” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

The “**publisher**” means any person or entity that distributes copies of the Document to the public.

A section “**Entitled XYZ**” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as “**Acknowledgements**”, “**Dedications**”, “**Endorsements**”, or “**History**”.) To “**Preserve the Title**” of such a section when you modify the Document means that it remains a section “Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled "History". Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an “aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled “Acknowledgements”, “Dedications”, or “History”, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <https://www.gnu.org/licenses/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License “or any later version” applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy’s public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

“Massive Multiauthor Collaboration Site” (or “MMC Site”) means any World Wide Web server that publishes copyrighted works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A “Massive Multiauthor Collaboration” (or “MMC”) contained in the site means any set of copyrighted works thus published on the MMC site.

“CC-BY-SA” means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

“Incorporate” means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is “eligible for relicensing” if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright © YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with ... Texts." line with this:

with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.