

PROJET DE RÉALISATION TECHNOLOGIQUE 4GE 2015-S2

Rapport Final

Badr BOUSLIKHIN

Douglas RAILLARD

Tuteur : Thomas GRENIER

7 juin 2015

Résumé

Ce projet est avant tout un projet personnel destiné à développer les compétences associées à la création d'une carte de développement sur laquelle un système GNU/Linux serait à même de fonctionner. Afin de rendre la carte plus simple à réaliser (design et fabrication) et modifiable, un microcontrôleur a été choisi au lieu d'utiliser un microprocesseur tel qu'un ARM Cortex A8 comme on en trouve sur les cartes BeagleBone Black [5] ou encore le Cortex A7 de la Raspberry Pi [21]. En effet, la complexité de ces circuits impose de fortes contraintes telles que l'utilisation de RAM DDR/DDR2/DDR3, des boîtiers BGA, faisant exploser les coûts de fabrication au passage. En effet au vu des densités de ces Integrated Circuits (ICs), leur implémentation en boîtier BGA nécessite des finesse de pistes importantes ainsi que des vias de diamètre très faible (vias laser en général), ce qui augmente très fortement la complexité et le coût des Printed Circuit Boards (PCBs). De plus ces microprocesseurs requièrent en général un séquencement des alimentations nécessitant une Power Management Unit (PMU) pour générer les tensions nécessaires au démarrage du processeur dans le bon ordre.

Cette réalisation combine donc autant l'aspect matériel que logiciel, les deux étant fortement couplés via des contraintes dans les deux sens (matériel sur logiciel et logiciel sur matériel).

Table des matières

Résumé	1
Introduction	5
1 Contexte	5
2 Cahier des Charges	5
3 Hardware	6
3.1 Microcontrôleur	6
3.1.1 Choix du microcontrôleur	6
3.1.2 Présentation du STM32F429ZI	7
3.2 Mémoire vive externe (RAM)	7
3.2.1 Principe de fonctionnement	7
3.2.2 Solution retenue	11
3.2.3 Interface des SDRAM	11
3.3 Mémoire de stockage de masse (Flash)	12
3.3.1 Comparatif	13
3.3.2 Interface des NOR	13
3.4 Périphériques et interfaces	14
3.4.1 IHM - Écran LCD	14
3.4.2 M2M	16
3.5 Outils utilisés	16
3.5.1 STM32CubeMX	16
3.5.2 Altium	17
3.5.3 SVN	17
3.6 Mise en œuvre du PCB	18
3.6.1 Schématique	18
3.6.2 Routage	23
3.6.3 Procédure de test et board bring-up	27
4 Software	27
4.1 Architecture générale	29
4.1.1 Noyau	29
4.1.2 Bootloader	30
4.1.3 Espace utilisateur	30
4.1.4 Système de fichier ROMFS	30
4.2 Solutions existantes	31
4.3 Outils de développement	31
4.4 Construction du firmware	32
4.4.1 Paramétrage du noyau uClinux	32
4.4.2 Paramétrage du bootloader U-Boot	32
4.4.3 Paramétrage de BusyBox	33
4.4.4 Utilisation des différentes mémoires	33
4.5 Validation du hardware	33
4.6 Prévision des performances	33

Conclusion	34
Annexe 1 : Schémas électriques de la carte	35
Annexe 2 : Bill Of Material de la carte	43
Glossaire	45
Références	48

Table des figures

1	Organisation d'une DRAM	9
2	Cellule mémoire d'une DRAM	10
3	Chronogramme d'un accès en lecture à la SDRAM Micron choisie [24] . . .	12
4	Chronogramme d'un accès en lecture à la NOR Spansion choisie [23] . . .	14
5	Timings vidéos d'une trame de 640x480	15
6	Timings vidéos LCD	16
7	Interface de STM32CubeMX	17
8	Schéma-bloc de la carte	18
9	Schéma utilisé pour la simulation	20
10	Tension de sortie	21
11	Ligne d'électrolyse à l'usine de Cirly	23
12	Vue 3D du PCB	24
13	Stack-up utilisé pour le PCB	25
14	Exemple de placement et routage du Buck	26
15	Place de la libc dans un système GNU/Linux	28
16	Stabilité des interfaces du noyau Linux	29
17	Carte du noyau Linux	30

Introduction

Ce document présente les différents aspects du projet ainsi que les problèmes auxquels nous avons été confrontés. Les choix techniques sont présents à toutes les étapes, du cahier des charges (afin de le rendre réalisable et cohérent) à la mise en œuvre finale. Il présente aussi dans les grandes lignes les étapes de validation de la carte telles qu'elles sont envisagées. Ce document est donc structuré en quatre parties :

- La première partie présente le contexte de ce projet ainsi que ses objectifs pédagogiques
- La deuxième partie présente le cahier des charges qui a été à l'origine de ce projet personnel
- La troisième partie détaille les choix techniques liés au matériel
- La quatrième partie détaille les choix des composants logiciels sélectionnés afin d'exploiter cette carte

1 Contexte

Ce projet est réalisé dans le cadre du Projet de Réalisation Technologique de 4^e année du département de Génie Électrique de l'INSA de Lyon. Il a pour objectif de développer des compétences pratiques dans la réalisation d'un système électronique embarqué. Il a fait l'objet d'une réflexion en amont afin d'identifier les objectifs et a ensuite été proposé en projet de PRT avec des objectifs atteignables dans l'état actuel de nos connaissances.

2 Cahier des Charges

L'objectif est de réaliser une carte de développement construite autour d'un ARM Cortex-M4 dans le but d'y utiliser un système GNU/Linux. La première contrainte physique qui en découle est la nécessité d'une RAM externe (la RAM interne de 256Ko étant trop petite pour faire tourner un tel système). Il faut également de la mémoire Flash de stockage externe (la mémoire Flash interne de 2Mo est tout juste capable de stocker un système de base composé d'un bootloader, d'un noyau et de BusyBox [7]). Afin de rendre le projet plus intéressant du point de vue des applications possibles du résultat, les périphériques suivants ont été implémentés :

- Ecran LCD couleur 480x272
- USB On-The-Go (hôte et esclave)
- UART
- Un certain nombre de General Purpose Input/Outputs (GPIOs), également utilisables pour d'autres usages tels que un bus I2C

Le microcontrôleur de cette carte est un ARM Cortex-M4 produit par STMicroelectronics, le STM32F429ZI. Un portage de uClinux 2.6.33 (février 2010) sur cette plate-forme est disponible et maintenu par la société Emcraft, ce qui a entre autres poussé au choix de ce microcontrôleur.

3 Hardware

3.1 Microcontrôleur

De nombreux microcontrôleurs sont disponibles aujourd’hui et se confrontent sur l’arène de l’électronique embarquée, on peut citer notamment les familles suivantes :

- Atmel AVR (8-bit), AVR32 (32-bit)
- Microchip Technology PIC, (8-bit PIC16, PIC18, 16-bit dsPIC33 / PIC24), (32-bit PIC32)
- Texas Instruments MSP430 et C2000
- Processeurs/microcontrôleurs à base d’ARM

Un acteur tire son épingle du jeu et se démarque particulièrement des autres : ARM. ARM est une société britannique spécialisée dans le développement d’architectures 32 bits de type RISC. C’est une entreprise *fabless* dont le modèle économique particulier de la microélectronique : la conception de Intellectual Property (IP). Ainsi il n’est pas possible d’acheter un processeur/microcontrôleur directement à ARM comme c’est le cas pour Intel, des licences sont attribuées à d’autres fabricants de semi-conducteurs qui proposent donc des puces basées sur la propriété intellectuelle ARM dans leurs catalogues. Les coeurs ARM sont le plus souvent intégrés au sein de System on Chip (SoC) complets. Ils sont très présents dans les systèmes embarqués (téléphone mobile, console portable, tablette électronique). En 2010, ARM était présent dans plus de 95% des smartphones, 10% des ordinateurs portables et 35% des télévisions connectées et set-top box. [3] En 2014, plus de 50 milliards de chipsets à base d’ARM ont été produits. ARM vise aujourd’hui à s’implanter sur le marché des ordinateurs personnels et des serveurs, notamment avec des projets tels qu’Euroserver [13]. Les perspectives de croissances sont prometteuses.

Au vu des données ci-dessus, il paraît évident que la connaissance des technologies ARM devient de plus en plus importante pour un ingénieur génie électrique aujourd’hui. C’est la raison principale pour laquelle nous avons choisi de travailler sur un ARM.

ARM a commencé à développer une gamme de Cortex-M, depuis 2003. Ce sont des microcontrôleurs relativement simples (en opposition aux Cortex A par exemple), ils sont facilement intégrables pour les raisons citées en introduction.

Année d’annonce	Core
2004	Cortex-M3
2007	Cortex-M1
2009	Cortex-M0
2010	Cortex-M4
2012	Cortex-M0+
2014	Cortex-M7

Dans la série des Cortex-M, notre choix s’est porté sur un Cortex-M4 car c’est l’un des plus puissants de la gamme, derrière les Cortex-M7. Il implémente l’architecture ARM v7-M. Ce qui le différencie particulièrement des autres Cortex inférieurs est que le M4 dispose d’instructions de Digital Signal Processor (DSP) et d’une Floating Point Unit (FPU) optionnelle. Conceptuellement, un Cortex-M4 est un Cortex-M3 avec ces deux éléments

en plus. Un Cortex-M4 embarquant ledit FPU est nommé Cortex-M4F

De nombreux fabricants proposent aujourd’hui des microcontrôleurs basés sur un ARM Cortex-M4/M4F, on peut citer :

- Atmel SAM4L, SAM4N, SAM4S, SAM4C (dual core), SAM4E, SAMG
- Freescale Kinetis K, W2
- NXP LPC4000, LPC4300 (dual core, Cortex-M4F + one Cortex-M0)
- STMicroelectronics STM32 F3, F4, F7 (nouvelle famille basée sur le cœur ARM Cortex M7 dotée de caches)
- Texas Instruments LM4F, TM4C et plus récemment le MSP432

Ces chipsets se valent pratiquement tous et proposent les mêmes fonctionnalités. Le STM32F429ZI a été retenu, car c'est l'un des moins chers du marché (12,54 €) à être cadencé à 180 MHz, et supporter les périphériques que l'on souhaite implémenter (SDRAM, NOR Flash, LCD, USB). De plus, la documentation proposée par ST est relativement exhaustive (datasheets, notes d'applications abondantes) et le portage de uClinux déjà réalisé.

Autre avantage de ce chipset : une carte d'évaluation existe déjà embarquant une grande partie des périphériques que nous souhaitons implémenter. Nous avons acquis cette carte d'évaluation en début du projet pour paralléliser les développements hardware et software.

3.1.2 Présentation du STM32F429ZI

Le microcontrôleur choisi est donc le STM32F429ZI qui est donc un Cortex-M4F cadencé à 180 MHz embarquant un DSP et une FPU. Il dispose de 2MB de Flash interne organisée en 2 banques permettant un *read-while-write*, 256KB de SRAM, un Flexible Memory Controller (FMC) permettant l'interfaçage du MCU avec différents types de mémoires externes : SRAM, PSRAM, SDRAM/LPSDR SDRAM, Compact Flash/NOR/-NAND, un contrôleur LCD parallèle permettant de driver des écrans en RGB565/666, 3×12-bit, 2.4 MSPS ADC, 2×12-bit DAC, un contrôleur DMA 16-canaux, 17 Timers. En termes de périphériques d'IO, le STM32F429ZI dispose de 4 entrées pour encodeurs, 3xI2C, 4 USARTs/4 UARTs, 6 SPIs avec 2 interfaces I2S full-duplex pour l'audio, 1 SAI (serial audio interface), 1 contrôleur USB 2.0 full-speed¹ device/host/OTG avec un PHY embarqué et 1 contrôleur USB 2.0 high-speed/full-speed²device/host/OTG avec DMA dédié à utiliser avec un PHY externe (interface UTMI+ low pin interface (ULPI) avec ce dernier), 1 contrôleur 10/100 Ethernet MAC avec DMA dédié à utiliser avec un PHY externe (interface (Reduced) Media-independent interface (MII/RMII) avec ce dernier). Deux interfaces de debug sont implémentées : le Joint Test Action Group (JTAG) et le Single Wire Debug (SWD).

3.2 Mémoire vive externe (RAM)

3.2.1 Principe de fonctionnement

Il y a deux technologies de fabrication des RAM : statiques et dynamiques.

1. USB 2.0 full-speed : 12 Mbit/s.
2. USB 2.0 high-speed : 480 Mbit/s.

- La SRAM ou RAM Statique est la plus ancienne. Les bits y sont mémorisés par des bascules électroniques dont la réalisation nécessite six transistors par bit à mémoriser. Les informations y restent mémorisées tant que le composant est sous tension et n'ont pas besoin d'être rafraîchies. Chaque bit d'une SRAM est formé par une bascule (latch) constituée de 4 à 6 transistors.

Le bascule RS comporte : deux entrées notées R (Reset) et S (Set) et deux sorties désignées par Q et -Q qui ont des valeurs inverses.

S	R	Sortie Q
0	0	Inchangée
0	1	$Q = 0$
1	0	$Q = 1$
1	1	-

La SRAM est très rapide et est pour cette raison le type de mémoire qui sert aux mémoires cache. Mais compte tenu du nombre de transistors nécessaires pour stocker un bit de données, son coût est prohibitif et sa densité faible.

- La DRAM pour RAM dynamique est plus simple à réaliser que la SRAM. Cela permet de faire des composants de plus haute densité et dont le coût est moindre. Chaque bit d'une DRAM est mémorisé par une charge électrique stockée dans un petit condensateur (dont la capacité est de l'ordre de la dizaine de fF). Ce dispositif offre l'avantage d'être très peu encombrant, mais a l'inconvénient de ne pas pouvoir garder l'information longtemps. Le condensateur se décharge au bout de quelques millisecondes (ms). Pour ne pas perdre le bit d'information qu'il contient, il faut un dispositif qui lit la mémoire et qui la réécrit de suite pour recharger les condensateurs. Ces RAM sont dites dynamiques, car cette opération de rafraîchissement ou de lecture doit être répétée régulièrement (64ms ou 16ms dans le domaine de l'automobile).

Il nous a semblé important de bien comprendre le principe de fonctionnement des DRAM avant de nous lancer dans son implantation sur notre carte.

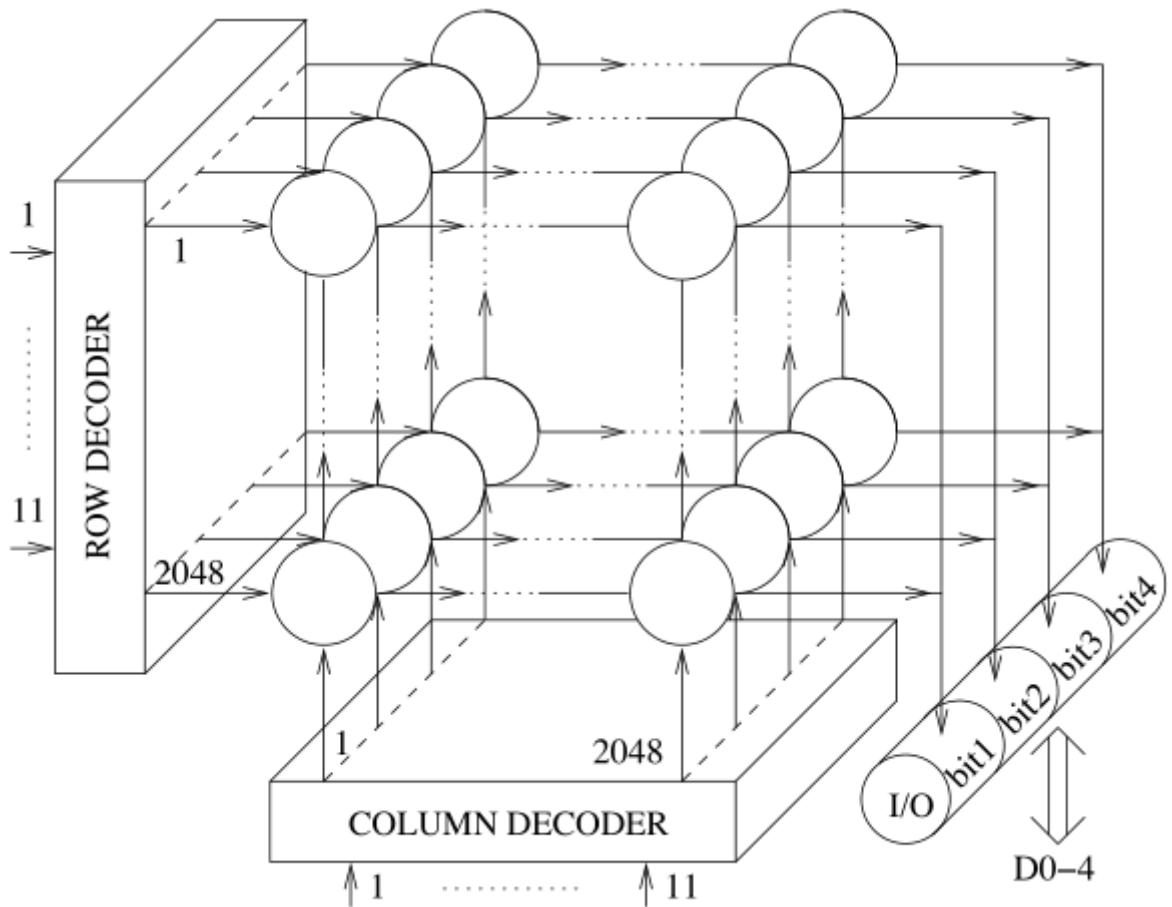


FIGURE 1 – Organisation d'une DRAM

La figure 1 est une représentation simplifiée de la structure d'une DRAM. Chaque cercle représente une cellule mémoire, les lignes sortant des *decoders* sont utilisées pour sélectionner la cellule mémoire à activer et les lignes sortant des cellules mémoires sont utilisées pour transférer les données stockées dans le buffer I/O. La DRAM représentée ici fait $2048 \times 2048 \times 4 = 16$ MBits.

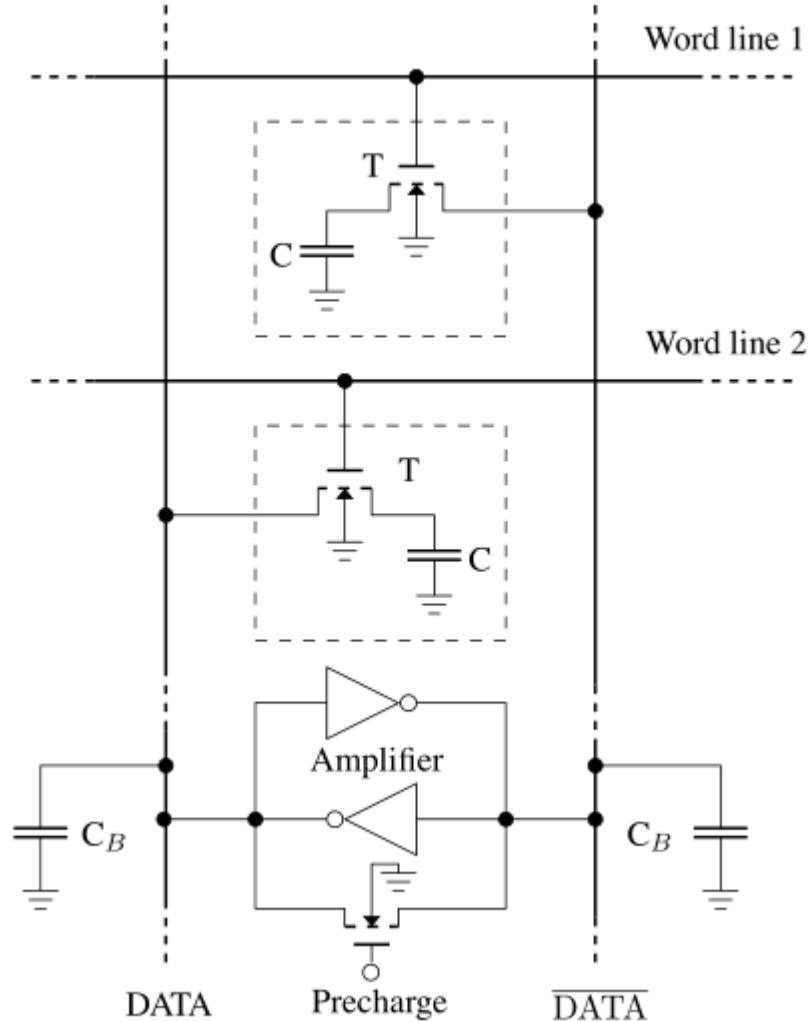


FIGURE 2 – Cellule mémoire d'une DRAM

Chaque cellule mémoire a un condensateur enregistrant la donnée sous forme d'une charge électrique et un transistor d'accès agissant comme interrupteur pour sélectionner le condensateur. La grille du transistor d'accès est connectée à la *word line*. Une cellule mémoire contient autant de *word line* que de lignes mémoire. Les *bit line* sont connectées à DATA et #DATA. Il y a autant de *bit lines* que de colonne mémoires.

Avant d'accéder accès à la mémoire, le contrôleur de la DRAM effectue une opération dite de précharge : les *bit line* complémentaires sont portées à VCC/2. Ce mécanisme sert à gagner du temps lors des accès mémoires. Le contrôleur effectue ensuite une opération dite d'égalisation : les *bit line* sont court-circuitées avec un transistor pour être exactement au même potentiel. Le temps nécessaire à la précharge et l'égalisation est appelée *RAS precharge time* et le contrôleur ne peut accéder aux données qu'une fois ces opérations finies.

Quand le contrôleur veut accéder à une cellule mémoire, il fournit le signal *row address* pour sélectionner la ligne, ce signal passe par le *row buffer* puis est transféré au *row decoder*. Ce dernier décode l'adresse de la ligne et active la *word line* correspondante à la ligne à laquelle on désire accéder. A ce niveau, tous les transistors d'accès de la ligne sélectionnée sont passants. Les charges contenues dans

les condensateurs de stockage s'écoulent vers les condensateurs C_B .

Le problème est que les valeurs des condensateurs de stockage sont plus faibles que les capacités des *bit line* donc le potentiel de ces dernières varie peu (de l'ordre de la centaine de mV seulement) et dans un intervalle de temps très court. Pour régler ce problème, des amplificateurs ont été rajoutés. Si le condensateur de stockage est vide, le potentiel de la *bit line* correspondante décroît, si le condensateur de stockage est chargé, le potentiel de la *bit line* croît. L'amplificateur amplifie la différence de potentiel entre la paire de *bit line*. Dans le premier cas, le potentiel de la *bit line* connectée au condensateur est mis à la masse et l'autre à VCC. Dans l'autre cas, c'est l'inverse qui se passe.

Sans précharge ni égalisation, l'amplificateur devrait porter augmenter/diminuer le potentiel de VCC au lieu de VCC/2.

Les signaux amplifiés sont transférés à l'*IO gate block*. Le *column decoder* décode l'adresse de la colonne à laquelle on souhaite accéder et transfère finalement le bit choisi au bloc IO.

On notera que l'accès aux données est destructif puisque le condensateur se vide. C'est là qu'intervient le condensateur C_B . Nous avons initialement copié la valeur originale dedans avant l'accès mémoire.

L'accès à une seule cellule mémoire entraîne le rafraîchissement de la ligne entière.

3.2.2 Solution retenue

Les SDRAM sont des variantes synchrones des DRAM fonctionnant sur le même principe. Elles sont économiques, d'une plus grande densité (5 à 10 fois que les SRAM) mais sont également plus lentes.

La grande majorité des SDRAM vendues aujourd'hui embarquent des contrôleurs gérant les aspects physiques liés au rafraîchissement, précharge, égalisation etc. facilitant de fait l'intégration de ce type de mémoires. A titre indicatif, le prix d'une SRAM de 16 Mbit (1M X 16 bit) est de 12€ sur Farnell contre 2,7€ pour une SDRAM de même capacité. La référence retenue est donc une Micron MT48LC32M8A2P-7E [24] de 256 Mbit cadencée à 133 MHz et de boîtier TSOP2 54 broches. Elle s'organise en 4 banques internes de 8Mx8x4.

3.2.3 Interface des SDRAM

L'interface des SDRAM est standardisée par le JEDEC [14]. Ceci permet une bonne interopérabilité des différentes mémoires avec les différents contrôleurs. Les signaux sont les suivants [24] :

- CLK : horloge de synchronisation
- CKE : Clock Enable, permet d'entrer dans un mode de basse consommation
- CS# : Chip Select, actif bas, permet d'activer la SDRAM pour signifier qu'elle doit lire une commande. Ceci permet de partager bus de donnée et d'adresse avec d'autres mémoires, comme c'est le cas avec le FMC (mémoire SDRAM et NOR)
- CAS#, RAS#, WE# : Column Address Strobe, Row Address Strobe, Write Enable : commande permettant respectivement de sélectionner une colonne, une ligne, et enfin d'informer d'un accès en lecture (WE haut) ou en écriture (WE bas).

- BA[1 :0] : sélection de la banque interne sur laquelle s'appliqueront les commandes
- A[12 :0] : bus d'adresse
- DQ[8 :0] : bus de données

Lors de l'accès à une banque donnée, le contrôleur envoie une commande de préchargement afin de désélectionner la ligne précédemment sélectionnée si ce n'est pas la bonne. Il envoie ensuite une commande d'activation d'une ligne (RAS#) en précisant son adresse sur le bus d'adresse, en même temps que la banque interne désirée au bout d'un temps t_{RP} . Au bout d'un nombre de cycles t_{RCD} , une commande CAS# est envoyée, avec l'adresse de la colonne sur le bus d'adresse. Au bout d'un temps t_{CL} , les données sont accessibles sur le bus de donnée. Une écriture nécessite l'usage du signal WE#.

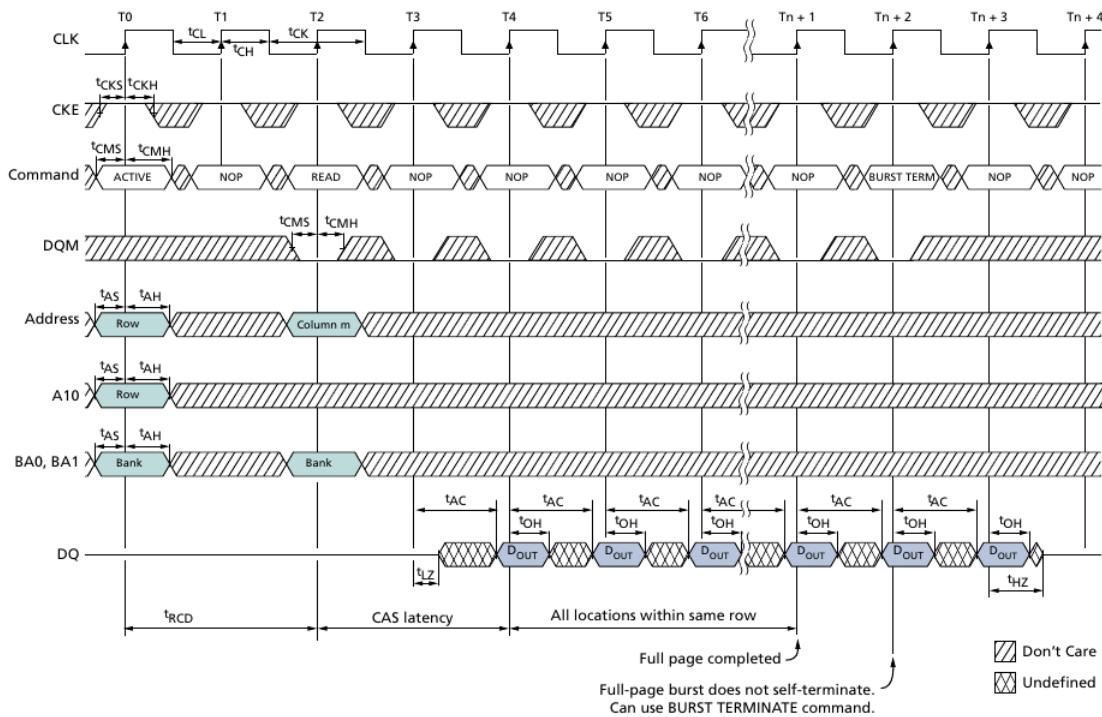


FIGURE 3 – Chronogramme d'un accès en lecture à la SDRAM Micron choisie [24]

En plus de ce fonctionnement de base, les SDRAM possèdent généralement plusieurs modes. En effet, la SDRAM est capable de fournir plus qu'un mot lors de la lecture d'une colonne sélectionnée. Ceci est appelé un *burst*, dont la taille est configurable à l'initialisation de la SDRAM.

3.3 Mémoire de stockage de masse (Flash)

Afin de pouvoir stocker le *firmware* de notre carte, il est nécessaire de rajouter de la mémoire de stockage de masse externe, car la Flash interne du microcontrôleur n'est pas suffisante (2Mo).

3.3.1 Comparatif

Deux familles de mémoire Flash sont disponibles : les NOR et les NAND.

Les NOR permettent un accès mot par mot, alors que les NAND utilisent un protocole plus complexe et ne permettent d'accéder qu'à des pages. La NOR permet donc l'eXecute In Place (XIP), car le cœur du microcontrôleur peut accéder aux instructions mot à mot, de manière transparente grâce au contrôleur mémoire FMC qui est un esclave Advanced High-performance Bus (AHB).

En contrepartie, les NAND ont une meilleure densité et sont donc moins chères. Les NAND possèdent également un système de stockage de métadonnées associées à chaque page permettant de marquer les pages qui ne sont plus utilisables ainsi que des codes correcteurs d'erreur.

En effet, toutes les mémoires flash sont sensibles à un phénomène d'usure compté en nombre de cycles érasement/écriture qui s'élève généralement de 10 000 à 100 000³. Cette fonctionnalité permet de relâcher les contraintes sur la qualité de production, et donc de baisser les prix, en augmentant la difficulté d'utilisation.

Une particularité des mémoires Flash est qu'une écriture ne peut que transformer un bit 1 en bit 0. Pour transformer un bit 0 en bit 1, il faut écraser un secteur, ce qui se fait avec généralement une granularité moins fine que les opérations de lecture dans le cas de la NOR.

La mémoire retenue est une mémoire NOR Spansion S29GL256P90TFIR20 de 32Mo avec un bus de donnée de 16 bits. Ce choix permet d'utiliser l'XIP qui évite de copier le code des programmes en RAM avant l'exécution, permettant donc de gagner une place notable pour les données. Le bus de 16 bit est particulièrement adapté sur cette plate-forme, car les instructions encodées en THUMB font généralement 16 bits. Ceci permet donc de lire une instruction en un cycle de lecture de la NOR.

3.3.2 Interface des NOR

L'interface des mémoires NOR est également standardisée par le JEDEC [14]. Les signaux suivants sont nécessaires au fonctionnement de la NOR [23] :

- A25–A0 : Bus d'adresse. Une adresse est associée à un mot de 16 bits et non à un octet
- DQ15–DQ0 : Bus de données 16 bits
- CE# : Chip Enable, actif bas, permet de sélectionner la NOR car elle partage le bus d'adresse et de données avec la SDRAM
- WE# : Write Enable, permet de spécifier une commande d'écriture
- BYTE# : Sélectionne la largeur du bus de données à utiliser (16 bits dans notre cas)
- RESET#

3. http://en.wikipedia.org/wiki/Flash_memory

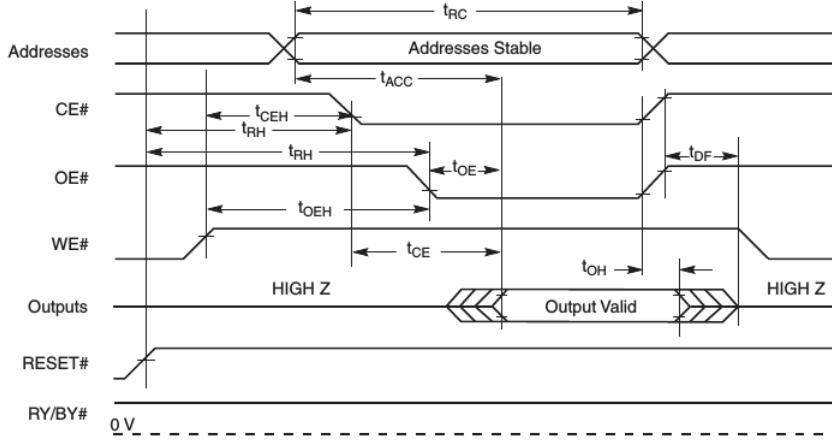


FIGURE 4 – Chronogramme d'un accès en lecture à la NOR Spansion choisie [23]

L'écrasement d'un secteur est une opération longue et sa durée peut être variable d'une puce à l'autre. Il est donc nécessaire de prévenir le processeur lorsque cette opération est finie. L'interface Common Flash Interface (CFI) permet à un pilote générique d'interroger la mémoire NOR afin de récupérer les valeurs maximales des temporisations d'écrasement. Ainsi, un pilote pour une NOR compatible CFI n'aura pas besoin d'être recodé pour chaque mémoire NOR, uniquement pour chaque contrôleur de mémoire.

3.4 Périphériques et interfaces

3.4.1 IHM - Écran LCD

L'écran que nous allons utiliser pour ce projet a été récupéré au ClubElek sur d'anciennes cartes de développement de TI (AM335x Starter Kit). Il s'agit d'un écran LCD fabriqué par Newhaven Display, NHD-4.3-480272MF-ATXI#-T-1 [16]. C'est un écran LCD de résolution 480x272, affichant jusqu'à 16.7M couleur. Il est également équipé d'une dalle tactile résistive et d'un rétroéclairage intégré. Le driver implémenté sur l'écran est un Orise OTA5180A supportant que le RGB888 [18]. L'écran est autosuffisant : il suffit de lui fournir une tension de 3V3 pour l'alimentation, une tension de 21V pour le rétroéclairage et des signaux vidéos avec les bons timings. Toute l'implémentation hardware du driver est déjà réalisée par Newhaven Display. Le protocole vidéo utilisé est un protocole LCD parallèle relativement standard. Les différents signaux à fournir sont :

- PLCK : horloge de synchronisation
- DISP : allumage/extinction de l'écran
- HSYNC : signal de synchronisation de lignes
- VSYNC : signal de synchronisation de trames
- DE : sélection de l'activation des données
- R[7 :0] : données de couleur rouge
- G[7 :0] : données de couleur verte
- B[7 :0] : données de couleur bleue

La difficulté de ce genre d'interfaces n'est pas tant l'intégration hardware (l'horloge vidéo est à 9 MHz) c'est plutôt les drivers software générant les bons timings vidéos. Le

boîtier 144 broches de notre microcontrôleur a cependant posé un problème : en effet, certains signaux nécessaires au fonctionnement de la mémoire NOR utilisent les mêmes broches que des bits de poids faible des bus de couleur de l'écran (signaux R, G et B). Il a donc été décidé de relier ces bits de poids faibles à la masse afin de les utiliser pour la NOR. On obtient alors une configuration RGB666.

Ces timings sont assez communs dans le monde de la vidéo (CVBS, MIPI, ...). Il s'agit des *Front porch*, *Back porch* horizontaux et verticaux. La spécification de ces timings se trouve dans la datasheet de l'Orise OTA5180A [18].

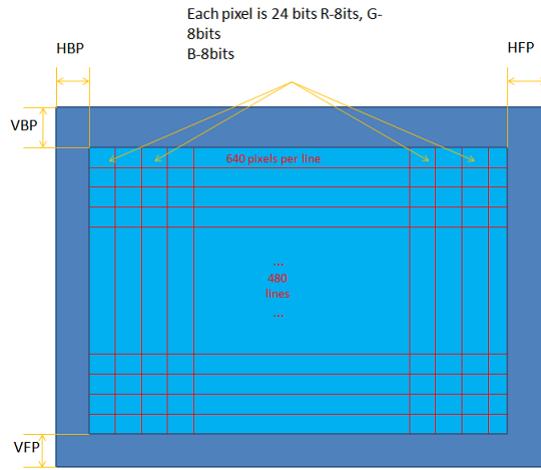


FIGURE 5 – Timings vidéos d'une trame de 640x480

Ces paramètres sont un héritage des vieux standards de télévision que sont le PAL et le NTSC. À l'époque des télévisions à tube cathodique, c'est un faisceau d'électrons qui balaie l'écran et, en frappant une surface recouverte de matériaux phosphorescents, génère une lumière sur l'écran. La déviation de ce faisceau était faite par un champ magnétique généré par des bobines. Deux signaux électriques sont principalement appliqués pour se faire :

- Un signal commandant le balayage horizontal («en x»), sur les bobines situées au-dessus et en dessous du faisceau ;
- Un signal commandant le balayage vertical («en y»), sur les bobines situées à gauche et à droite du faisceau.

Les temps de *blanking* ont été introduits pour synchroniser le faisceau sur la partie dite «active» de l'écran. Ainsi le *Vertical Blanking Interval* ou *Vertical Front/Back Porch* ont été introduits pour laisser le temps au faisceau de remonter en haut de l'écran à la fin d'une trame. Il en va de même pour l'*Horizontal Blanking Interval* ou *Horizontal Front/Back Porch*. Ces réglages représentent une des parties délicates à parfaire avant de pouvoir afficher une image correcte.

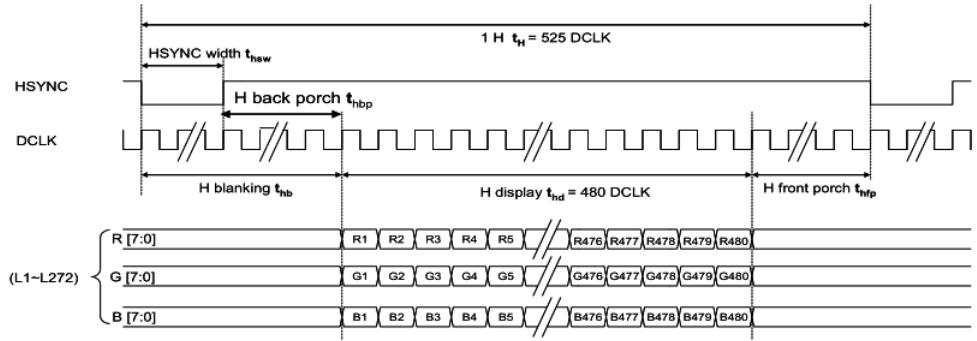


FIGURE 6 – Timings vidéos LCD

Source: Datasheet du contrôleur LCD Orise OTA5180A [18]

3.4.2 M2M

- UART : Des signaux d'UART ont été routés sur des *headers* pour pouvoir communiquer en série avec la carte. Cette communication série pourra servir au debug ainsi qu'à de la communication *Machine to machine* .
- GPIOs : 20 GPIOs ont été laissés à l'utilisateur pour être utilisées comme GPIOs ou d'autres bus tels qu'I2C.

3.5 Outils utilisés

3.5.1 STM32CubeMX

Le STM32CubeMX [26] est un outil fourni par ST pour faciliter et accélérer les développements des produits embarquant un MCU STM32. C'est un outil permettant de générer le code d'initialisation en C en utilisant l'interface graphique. Plusieurs fenêtres sont disponibles : l'assistant « brochage » facilite l'affectation des broches pour éviter les conflits grâce à un solveur de conflits embarqué ; l'assistant « arbre d'horloge » attribue les horloges et exécute la validation dynamique ; l'assistant « périphériques et middleware » pour éviter tout paramètre non utilisable/conflit ; et l'assistant « consommation d'énergie » qui permet d'estimer la consommation de MCU et les périphériques configurés. Il a grandement servi lors du développement hardware pour faire le *mapping* des signaux en sortie/entrée du MCU et à s'assurer facilement et rapidement qu'il n'existe pas de conflit entre les différentes fonctionnalités associées à chaque broche.

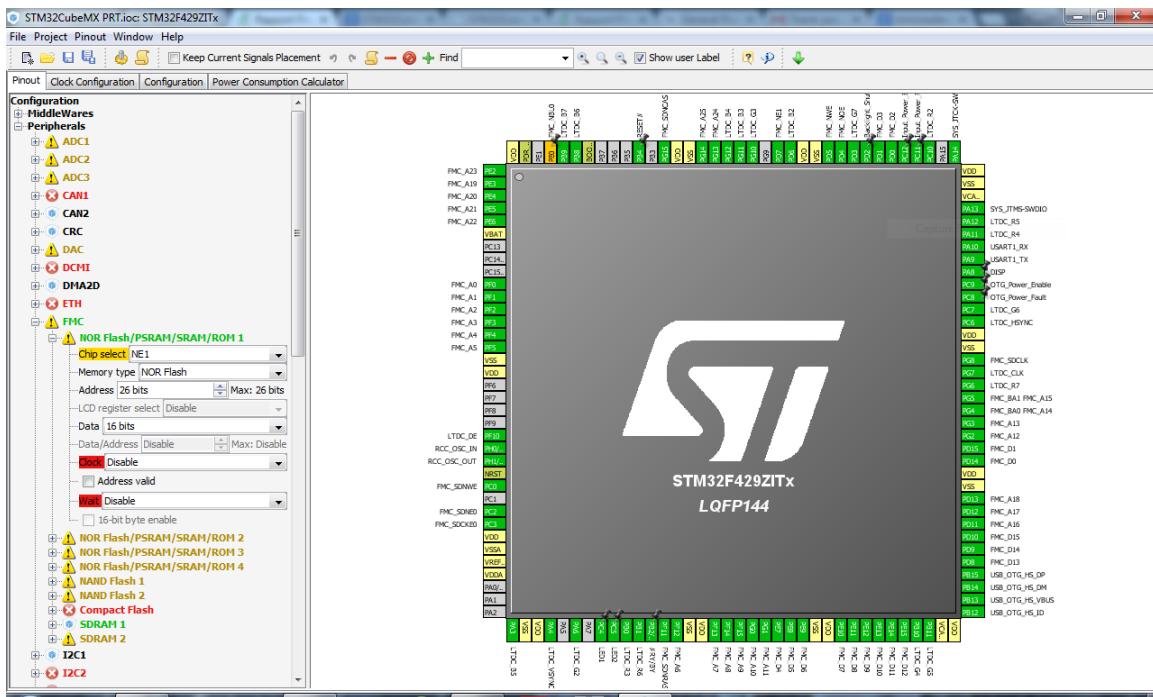


FIGURE 7 – Interface de STM32CubeMX

3.5.2 Altium

Altium Designer [2] est un logiciel de conception assistée par ordinateur pour systèmes électroniques. Altium Designer propose des fonctions de [20] :

- Saisie de schémas
 - Simulation (basée sur Xspice)
 - Conception/routage de circuits imprimés, capture des schémas, gestion des bibliothèques de composants, routage, génération des fichiers de fabrication Gerber
 - Programmation de FPGA et microprocesseurs/microcontrôleurs
 - Interfaçage avec des logiciels de CAO mécanique (SolidWorks par ex.) permettant de faciliter l'intégration de l'électronique dans la mécanique

Nous avons choisi d'utiliser ce logiciel plutôt qu'une autre (Eagle, Allegro, KiCad, etc.) parce que c'est celui que nous utilisons au ClubElek et que nous savons utiliser.

3.5.3 SVN

Afin d'héberger les différents documents concernant le projet ainsi que les schémas électriques, un dépôt SVN a été créé sur la plate-forme Assembla [4]. Un compte gratuit correspondant aux prestations normalement payantes nous a gracieusement été offert par l'équipe d'Assembla. Ce dépôt SVN permet de versionner tous les fichiers, sans risques de perdre ou d'écraser des modifications, et il permet aussi de synchroniser facilement entre nos ordinateurs et de manière déterministe (contrairement à Dropbox ou encore Google Drive qui ne sont pas vraiment adapté car on ne choisit pas l'instant de synchronisation, ce qui amène des problèmes). De plus, SVN est intégré à Altium, nous pouvons gérer le versionnement directement depuis celui-ci.

3.6 Mise en œuvre du PCB

3.6.1 Schématique

Nous avons commencé par faire les schémas de la carte. Pour garder de la lisibilité, nous n'avons pas mis tous les schémas sur une feuille A0. Nous avons découpé les pages de schémas en blocs logiques s'inspirant fortement du diagramme que nous avons initialement défini.

Altium dispose d'outils puissants permettant de faire ce genre de schématique en utilisant différents types de connecteurs pour relier les signaux entre eux [9] (*Net Label* , *Port* , *Sheet Entry* , etc.).

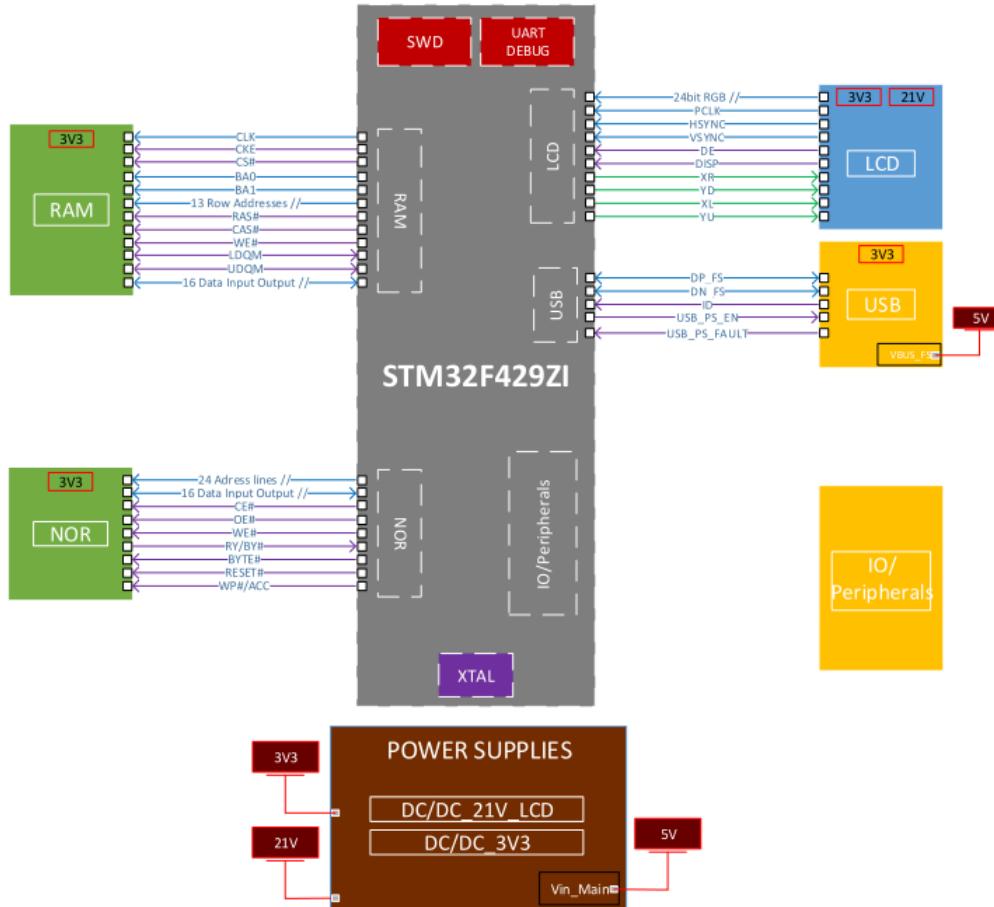


FIGURE 8 – Schéma-bloc de la carte

Revoyons page par page de la schématique (page 35) les points importants :

- Alimentations : nous avons choisi d'alimenter notre carte à partir d'une alimentation USB standard délivrant du 5V (du même genre qu'on utilise pour charger nos smartphones). Ce choix nous évite d'avoir besoin d'une alimentation stabilisée pour générer les tensions nécessaires à la carte et d'utiliser un convertisseur AC/DC pour passer de la tension secteur à un 3V3 stabilisé.

Nous avons besoin de deux tensions différentes sur notre carte : une tension de 3,3V pour alimenter tous les circuits numériques que nous utilisons et une tension de 21V pour alimenter les LEDs de rétroéclairage d'écran.

Nous avons fait la somme des courants consommés par chaque ICs dans le pire des cas à 85°C, nous arrivons à un pire cas de 580 mA sur l'alimentation 3,3V. Nous choisissons donc d'utiliser un convertisseur Buck pour faire la conversion 5V/3,3V. Nous choisissons d'utiliser un Buck de TI, le LMR10510. Il dispose de caractéristiques parfaitement adaptées à nos besoins :

- *Input voltage range of 3V to 5.5V*
- *Output voltage range of 0.6V to 4.5V*
- *Output current up to 1A*
- ***1.6-MHz Switching Frequency***
- *Low shutdown I_q , 30 nA typical*
- *Internal soft-start*
- *Internally compensated*
- ***Thermal shutdown***

De plus le boîtier de ce Buck est un SOT23 tout à fait soudable manuellement. Le LMR10510 est proposé à 1,08€ sur Farnell et TI fournit des samples gratuitement. La fréquence relativement élevée de découpage de ce Buck à 1,6 MHz permet de réduire la taille de l'inductance ce qui permet une plus grande intégration de la carte.

Le dimensionnement de l'inductance, la capacité et la diode de sortie a été fait en utilisant une feuille Excel que nous avons créée. Les formules utilisées proviennent de la datasheet et des cours que nous avons eu.

Les valeurs calculées sont les suivantes :

Rapport cyclique	Valeur d'inductance	Capacité de sortie
70%	2,19 nH	40 nF

Pour la diode, nous choisissons une diode Schottky pour sa tension directe faible et ses temps de commutation faible adaptée adaptée à notre fréquence de découpage. Nous ferons attention également à ce que sa tension de blocage inverse soit au moins de 3,3V.

Nous utilisons les valeurs proposées par la datasheet pour une tension d'entrée de 5V, une tension de sorte de 3,3V et un courant de sortie moyen de 1A :

Valeur d'inductance	Capacité de sortie
2,2 uH	22 uF

Nous remarquons que les valeurs de la capacité et de l'inductance sont largement supérieures à celles que nous avons calculées. Cela impacte positivement les performances du système puisque ça a pour effet de diminuer le courant de *ripple*. Nous estimons que celui-ci sera de quelques mV.

Pour générer la tension de 21V, nous utilisons un Boost. Nous estimons le courant de charge à 40 mA dans le pire des cas.

Nous choisissons pour pratiquement les mêmes raisons que le Buck d'utiliser le LM27313 de TI. Ses caractéristiques sont les suivantes :

- *Input Voltage Range (2.7 V to 14 V)*
- ***1.6-MHz Switching Frequency***

- Output current up to 800 mA
- Low shutdown $I_q < 1 \mu A$ typical
- Internally compensated
- Thermal shutdown

Le boîtier est un SOT23 également. Le LM27313 est proposé à 1,23€ sur Farnell et TI fournit des samples gratuitement. Pour dimensionner ce Boost, nous allons utiliser une approche différente pour gagner du temps. Les valeurs de l'inductance, et la capacité de sortie seront déterminées par simulation. Nous utiliserons l'outil de simulation en ligne fourni par TI : Webench.

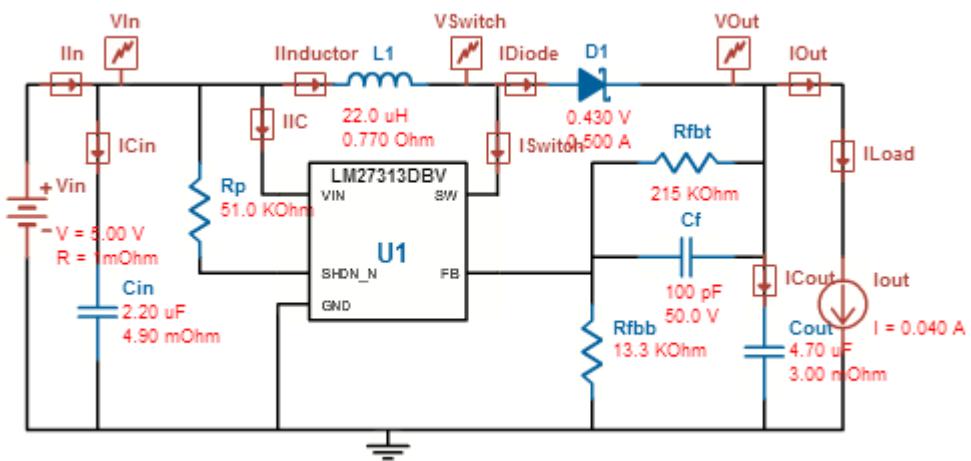


FIGURE 9 – Schéma utilisé pour la simulation

Les valeurs retenues sont les suivantes :

Valeur d'inductance	Capacité de sortie
22 uH	4,7 uF

En régime permanent, nous avons une tension de sortie qui varie très peu.



FIGURE 10 – Tension de sortie

La simulation estime le rendement à 70% ce qui n'est pas particulièrement étonnant compte tenu du faible courant de charge.

Pour conclure sur les alimentations, on notera que les Buck et Boost choisi sont protégés en court-circuit puisqu'ils disposent d'un *Thermal shutdown* arrêtant le découpage en cas de surchauffe des ICs. Nous veillerons également à utiliser des condensateurs céramiques en entrée et en sortie avec un diélectrique X5R ou X7R pour la stabilité de leur capacité en température et leur Equivalent Series Resistance (ESR)/Equivalent Series Inductance (ESL) faible. Ce dernier critère est particulièrement important, car un ESL important viendrait déstabiliser la boucle de régulation déplaçant ses pôles.

Des LEDs ont été placées pour indiquer la présence des bonnes tensions. On utilisera un transistor en saturé-bloqué pour la LED indiquant la présence du 21V pour éviter d'avoir une chute de tension trop importante au niveau de la résistance de limitation de courant de la LED. Cet étage a été dimensionné en simulation avec LTSpice.

- STM32F429ZI : le découplage du MCU est crucial pour son bon fonctionnement. Les capacités limitent la circulation de fortes variations des courants la carte. En effet, ce sont elles qui répondent aux fortes demandes de courant lors des commutations et ce sont elles qui mettent à la masse toutes les perturbations haute fréquence qui peuvent se retrouver sur le rail d'alimentation. Elles servent également de réserve d'énergie locale en cas de microcoupures de l'alimentation.

La *rule of thumb* impose 100 nF par pin d'alimentation + une plus grande capacité

servant de réservoir d'énergie local.

Nous choisissons d'utiliser un quartz pour générer l'horloge principale du MCU. C'est un Quartz Crystal de 8 MHz que nous avons choisi. Il permet d'avoir une horloge plus précise que l'horloge interne du MCU (± 50 ppm max.).

Par ailleurs il faudra faire attention à bien câbler les signaux sur les bonnes entrées. Avec 144 broches, l'erreur est vite arrivée.

- RAM : de la même manière que pour le MCU, le découplage de la RAM est crucial pour son bon fonctionnement. Nous avons appliqué les mêmes règles ici.

- NOR : de la même manière que pour le MCU, le découplage de la NOR est crucial pour son bon fonctionnement.

La principale difficulté de la NOR a déjà été mentionnée auparavant : le câblage du signal RY/BY#. En effet, ce signal indique au MCU la fin d'opérations longue telles que l'écrasement d'un secteur par exemple. Le problème est que ce signal rentrait en conflit avec un des bits de données de l'écran LCD et la question que l'on s'est posée a été de savoir si on pouvait se passer de ce signal ou pas. Après quelques recherches (l'information n'a pas été simple à trouver et à nécessité la consultation des sources des pilotes logiciels), il s'est avéré que l'interface CFI permet à un pilote générique d'interroger la mémoire NOR afin de récupérer les valeurs maximales des temporisations d'écrasement. Ainsi ce signal n'est pas indispensable. Nous l'avons quand même par précaution câblé sur une des entrées libres du MCU.

- USB : pour des raisons de sécurité et de conformité à la norme USB, nous avons choisi de rajouter un étage de protection sur les pistes VBUS de l'USB d'alimentation et de l'USB OTG. Il s'agit du TPS2552DBVT de TI qui est un limiteur de courant commandable dont les caractéristiques principales sont les suivantes :

- *Up to 1.5 A Maximum Load Current*
- *Meets USB Current-Limiting Requirements*
- *Adjustable Current Limit, 75 mA–1300 mA (typ)*
- **85-mΩ High-Side MOSFET** : donc pertes en conduction faibles. Ce critère est extrêmement important si on veut optimiser la consommation énergétique de notre montage et diminuer la chute de tension aux bornes du MOSFET.
Rappelons que c'est le premier étage à l'entrée de notre carte.
- *Reverse Input-Output Voltage Protection*
- *Fast Overcurrent Response - 2-us (typ)*
- *15 kV ESD Protection*

Ce circuit limite le courant en cas de court-circuit à une valeur maximale que l'on fixe de manière hardware et bloque le MOSFET pour couper le courant si la panne dure et que le circuit s'échauffe. Il protège également en cas de surtension ou de *reverse voltage* (tension de sortie > tension d'entrée) en bloquant le MOSFET.

Il servira donc, sur l'USB d'alimentation, à protéger l'alimentation sur laquelle la carte sera connectée en cas de court-circuit suite à une erreur de manipulation ou une panne de composant.

Et sur l'USB OTG, il protégera la carte si celle-ci est hôte (et donc délivre le 5V)

en cas de défaillance du device qui est connecté (que ce soit un court-circuit ou une surtension) ou inversement protégera l'hôte sur lequel est connectée notre carte dans le cas où celle-ci est device.

Il est capable de signaler une défaillance au MCU avec le signal FAULT#.

On peut également utiliser ces circuits comme interrupteurs commandables puisqu'ils disposent d'une broche EN#. On pourrait imaginer vouloir couper l'alimentation d'un périphérique USB par exemple ou redémarrer brutalement le MCU en envoyant un 0 sur l'EN# du circuit en entrée de l'alimentation (le circuit redémarre tout seul parce qu'il y a une pull-down sur les EN#)

- Périphériques I/O : le LCD a été relativement compliqué à câbler à cause d'une incompatibilité de notre MCU avec le protocole RGB888 qu'attend le driver de l'écran. En effet, notre MCU avec le boîtier que nous avons choisi TQFP144 est capable de générer au plus du RGB666, malgré le fait que la datasheet mentionne affirme le contraire (comme quoi...). La solution que nous avons choisi a donc été de relier ces bits de poids faibles à la masse.

3.6.2 Routage

- Fabricant du PCB : le ClubElek est sponsorisé par Cirly [8] spécialiste lyonnais de la fabrication rapide de circuits imprimés prototypes et petites séries. C'est en principe eux qui fabriqueront notre carte. Un membre du binôme (Badr) est allé visiter leur usine située à Brignais, c'était une visite très enrichissante. Il a pu découvrir le process de fabrication et discuter avec eux des éventuelles contraintes à garder en tête lors de la conception de PCB 4 couches.



FIGURE 11 – Ligne d'électrolyse à l'usine de Cirly

- Taille du PCB : en vue d'une éventuelle intégration par la suite de cette carte dans une mécanique, nous choisissons de la faire la plus compacte possible.
Nous choisissons un PCB rectangulaire de 70x43mm ($70/43 = 1,62$; tiens, c'est le chiffre d'or ?). Plus sérieusement, ce *template* s'appelle Sick of Beige [25]. C'est un format utilisé dans le monde des hobbyistes, il a été proposé par le blog Dangerous Prototypes pour uniformiser les PCB faits dans le monde de l'OpenHardware. Dangerous Prototypes propose également des protections et supports pour ce format de PCBs.

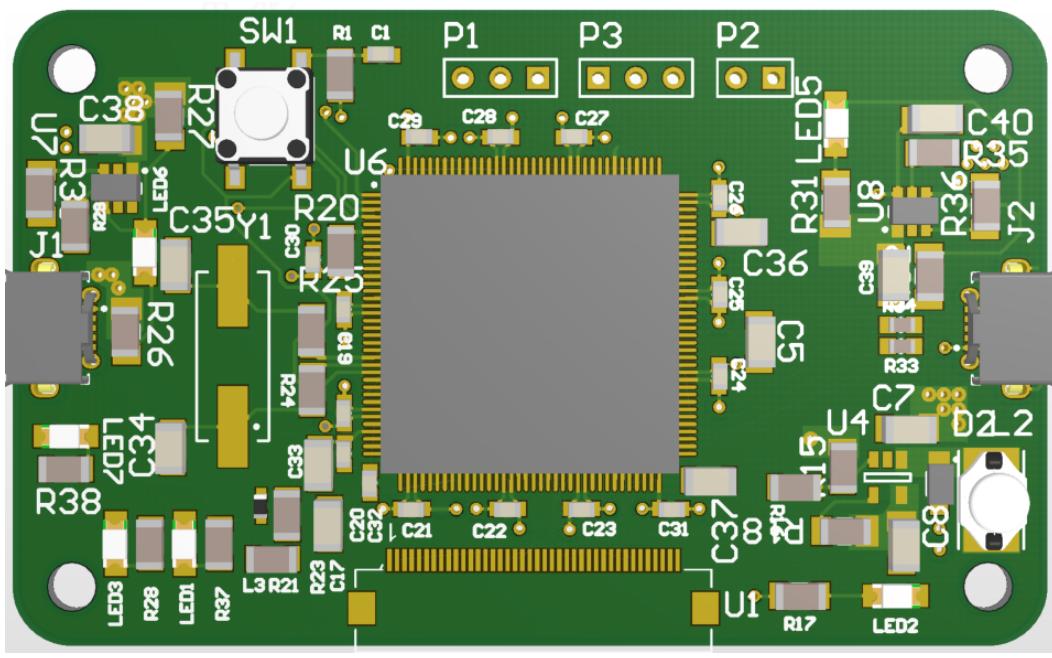


FIGURE 12 – Vue 3D du PCB

- *Stack-up* : compte tenu de la taille du PCB et du nombre de composants et signaux à router, on peut penser que la densité sera assez haute. Nous choisissons donc d'utiliser 4 couches pour ce routage. Le stack-up définit donc l'empilement de ces couches. Il est important de bien définir son stack-up en prenant en compte :
 - Le coût et les contraintes de fabrications : le coût augmente (et de manière non linéaire !) avec le nombre de couches, les fabricants de PCB proposent un certain nombre de stack-ups 'standard' même si les matériaux (épaisseurs, constantes diélectriques principalement) varient d'un fabricant à l'autre. Généralement, il est financièrement plus intéressant d'adapter son stack-up à ce que propose le fabricant.
 - Les contraintes CEM : un bon stack-up optimise la CEM et diminue la sensibilité des circuits aux parasites électromagnétiques.

N'ayant pas de contraintes CEM ni de contraintes de sensibilité électromagnétique, nous tiendrons compte seulement des deux premiers critères.

Nous avons demandé un dossier technique à Cirly et le stack-up retenu est finalement le suivant :

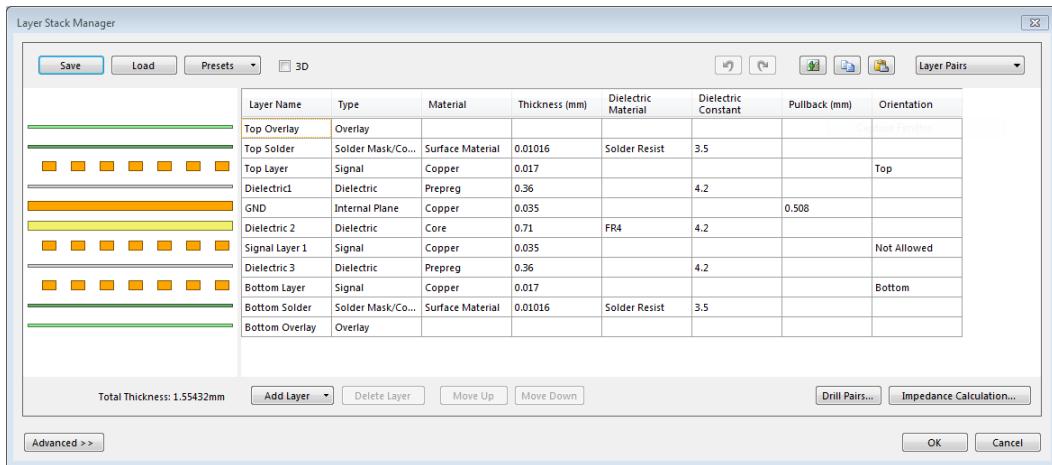


FIGURE 13 – Stack-up utilisé pour le PCB

On peut voir que nous avons renseigné l'ordre d'empilement des différentes couches, les types de matériaux utilisés, leurs épaisseurs ainsi que leurs constantes diélectriques (utilisées pour calculer les impédances des pistes).

- Placement et routage : Un bon routage réduit la majorité des problèmes (et c'est d'autant plus intéressant qu'il n'a aucune incidence sur la BOM, il n'y a pas de compromis coût/routage à faire). Quelques principes de base que nous essayons de respecter au maximum lors de ce routage (et tout le temps de manière générale) :
 - Sur les cartes mixtes (analogique/numérique) : un seul plan de masse avec zonage des parties mixtes. Cela sert à garder une relative maîtrise de la circulation des courants sur la carte.
 - Tracer des pistes formant les boucles les plus courtes possible.
 - Les capacités de découplage doivent être placées au plus près des composants. Il faut éviter les vias entre les capacités de découplage et les IC car les vias sont fortement inductives (plus inductives qu'une piste).
 - Faire attention aux diaphonies par couplage capacitif ou inductif. Une perturbation sur un conducteur va injecter un signal perturbateur sur le conducteur voisin. Ce couplage peut avoir deux origines :
 1. Lorsqu'un conducteur est soumis à une d.d.p., la capacité mutuelle (notée C_m) injecte un courant perturbateur sur le conducteur voisin. C'est la diaphonie capacititive.
 2. Lorsqu'un courant circule dans un conducteur, il génère un champ magnétique qui se couple dans la boucle formée par le conducteur voisin par rapport à la masse. C'est la diaphonie inductive.
 - L'égalisation de la longueur des pistes sur les bus de données et d'adresses des mémoires est conseillée pour avoir des temps de propagations égaux.
 - L'égalisation de la longueur des pistes et la symétrie des pistes de bus différentiels (le D+ et D- du bus USB par ex.) est conseillée.
 - L'utilisation d'un plan de masse permet de réduire de façon importante la diaphonie entre pistes lorsque la distance entre pistes est supérieure à la hauteur par rapport au plan de masse. La présence d'un plan de masse réduit la diaphonie de façon très significative. C'est pour cela qu'il vaut mieux placer, dès

que possible, des plans de masse dans le stack-up pour diminuer la diaphonie entre pistes sensibles (horloges, bus haute vitesse, pistes d'alimentations à découpage, etc.).

- Réduire les problèmes de CEM consiste avant tout à améliorer l'équipotentialité des systèmes en HF.
- Un plan de masse ne doit pas être fendu. Un plan de masse fendu voit son impédance fortement augmenter en HF. Il a une impédance très faible tant qu'il reste homogène.
- Un anneau de garde en bord de carte relié à la masse est une bonne pratique pour encaisser les ESD et augmenter l'équipotentialité des masses.

Il faut être attentif, minutieux, cohérent et logique dans son routage en respectant les règles de base citées ci-dessus et toutes autres règles que l'on s'impose. Parmi les quelques erreurs récurrentes que nous essayions d'éviter :

- Changement des largeurs de pistes en cours de route (éviter de former des goulots d'étranglement)
- Angles droits sur les pistes
- Pistes sensibles placées en dessous d'inductances
- Pistes formant des boucles très larges
- Plans de masse fendus
- Placements de capacités de découplage non optimisé

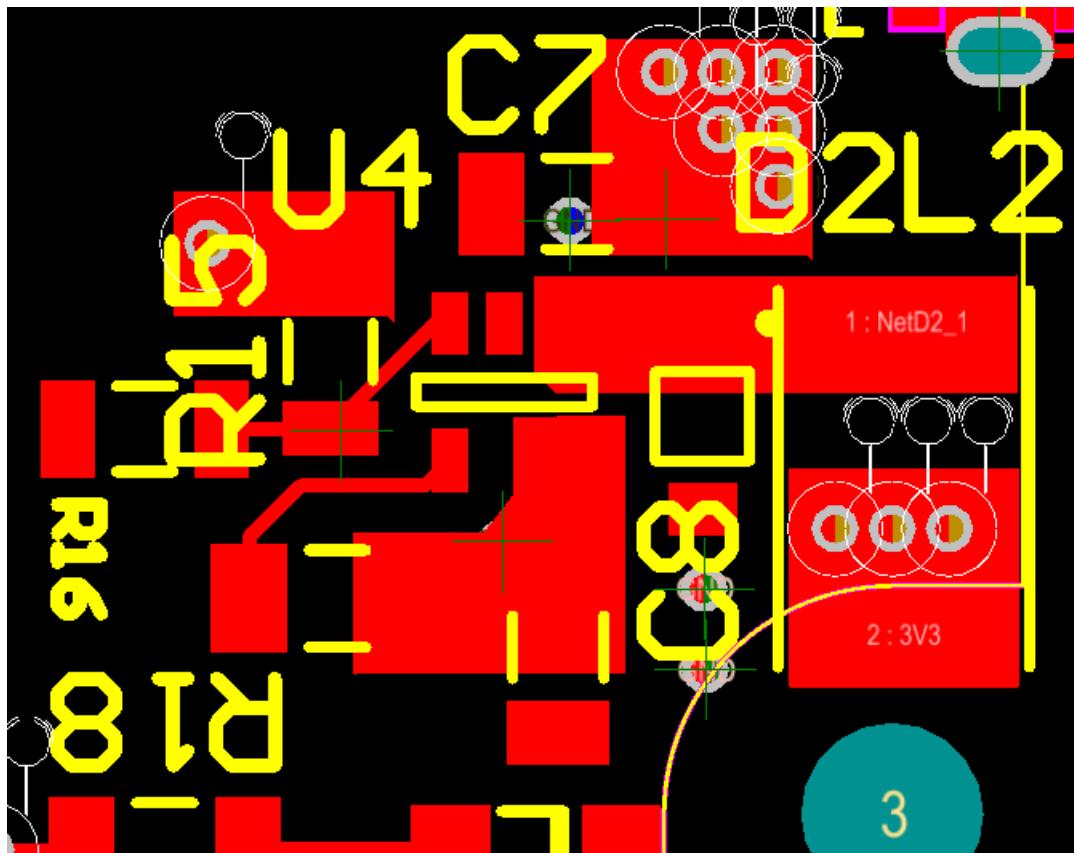


FIGURE 14 – Exemple de placement et routage du Buck

Nous avons essayé de garder la piste entre D2 et L2 la plus courte possible. Vu que ce sont des composants qui vont 'voir' le découpage, les inductances parasites des pistes risquent d'augmenter les résonances et les bruits. Nous avons également essayé de rapprocher R15, R16 et la broche FB dans le but d'éviter d'avoir du bruit qui s'ajoute à au signal de *feedback* et qui, de fait, perturberait la régulation. Nous avons également augmenté le nombre de vias connectant le plan de masse du Buck au plan de masse global pour augmenter l'équipotentialité des masses et diminuer l'impédance de celle-ci. Cela diminue les pertes, mais surtout réduit le bruit rayonné et conduit.

3.6.3 Procédure de test et board bring-up

Afin de s'assurer du bon fonctionnement de la carte, il est nécessaire de la tester une fois soudée.

Le premier test à effectuer est la validité des soudures, car des courts-circuits sont assez susceptibles d'être présents à cause du boîtier du microcontrôleur.

Une fois les soudures validée, il faut tester le bon fonctionnement des Buck et Boost en s'assurant qu'ils délivrent les bonnes tension.

Enfin, un test de programmation de la mémoire interne du microcontrôleur permettra d'envoyer un programme de test (U-Boot) que l'on exécutera et qui permettra de valider la bonne exécution du code et l'usage de l'UART.

4 Software

La partie logicielle de ce projet consiste en la création d'un firmware basé sur un système GNU/Linux. Elle nécessite donc l'utilisation d'un certain nombre de composants logiciels dont il faut récupérer les sources, les paramétrier et les compiler. Cette opération peut prendre un certain temps, car il faut s'assurer de la compatibilité de différentes versions qui n'ont pas été nécessairement testées ensemble. Il est donc indispensable d'être aux faits des types de changement que chaque brique peut apporter à son interface afin de savoir où peuvent se situer les problèmes potentiels. Un exemple évident de ce problème serait un programme utilisant un appel système qui ne serait pas implémenté dans la version du noyau utilisée, ou un programme nécessitant la glibc alors qu'une autre libc est utilisée. La figure 15 illustre la position occupée par la libc dans un système GNU/Linux.

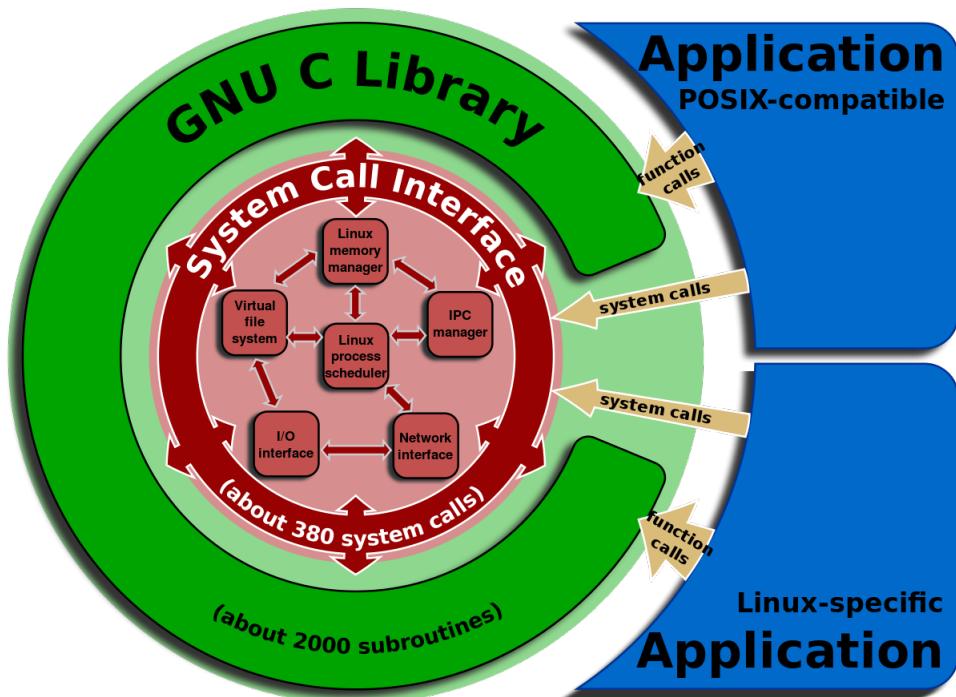


FIGURE 15 – Place de la libc dans un système GNU/Linux

Source: http://commons.wikimedia.org/wiki/File:Linux_kernel_System_Call_Interface_and_glibc.svg

On a donc les mêmes problématiques qu'un *packageur* d'une distribution dont le rôle est de compiler les différents logiciels en s'assurant que ses dépendances sont satisfaites dans les bonnes versions et en appliquant des patchs si nécessaire.

La figure 16 illustre la problématique de stabilité des interfaces telle qu'on la trouve dans le logiciel libre, du fait de l'indépendance des projets et de l'infinité de configurations et variations possibles dans les sources modifiées.

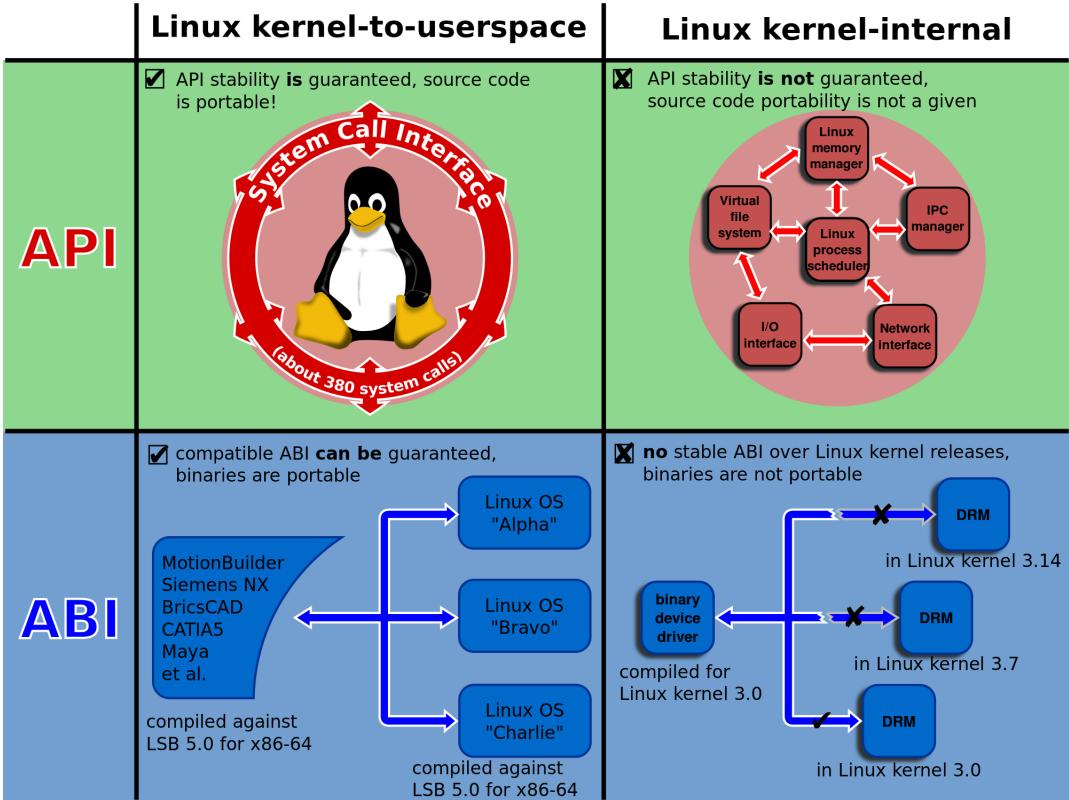


FIGURE 16 – Stabilité des interfaces du noyau Linux

Source: http://commons.wikimedia.org/wiki/File:Linux_kernel_interfaces.svg

4.1 Architecture générale

Étant données les contraintes de place dans la mémoire de la carte de développement utilisée pour le prototypage logiciel (2Mo de Flash interne), ainsi que dans la carte finale, il est nécessaire de construire le système le plus minimal possible. Ce n'est donc pas une distribution GNU/Linux tel que Debian ou même Ångström qui est utilisé mais un ensemble de composant logiciel choisi, configurés et compilés afin de constituer un système de fichier racine permettant un démarrage et une exploitation basique de la carte.

4.1.1 Noyau

Le noyau utilisé est le noyau uClinux, qui est une variante du noyau Linux pouvant s'exécuter sur des processeurs ne possédant pas de Memory Management Unit (MMU) [10]. Un certain nombre de fonctionnalités reliées à la présence de mémoire virtuelle sont absentes, tels qu'un comportement différent pour certaines options de `mmap` ou encore l'appel système `fork` (seul `vfork` est implémenté, car sans MMU, il est impossible d'implémenter le comportement copy-on-write lors de la création du nouvel espace d'adressage pour le processus). La stabilité du système est aussi plus faible, car un processus utilisateur peut corrompre la mémoire de n'importe quel programme, même le noyau, à moins que la Memory Protection Unit (MPU) ne soit utilisée.

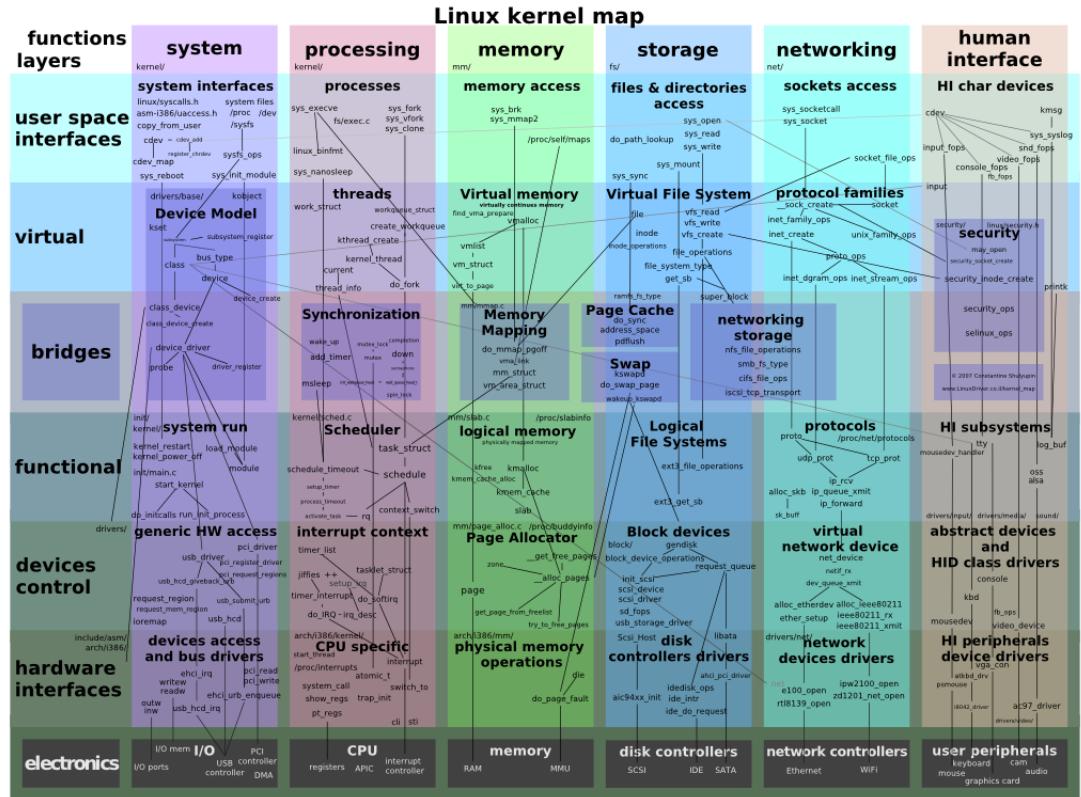


FIGURE 17 – Carte du noyau Linux

Source: http://www.makelinux.net/kernel_map/

4.1.2 Bootloader

Le bootloader U-Boot est le logiciel permettant d'amorcer l'exécution du noyau uLinux. Il permet également de reprogrammer la mémoire flash et propose une interface basique en ligne de commande accessible sur une UART. Il réalise différentes tâches d'initialisation telles que celles des mémoires externes (initialisation du FMC).

4.1.3 Espace utilisateur

La racine est presque exclusivement constituée du binaire de BusyBox [7], qui est un programme implémentant les commandes les plus basiques ainsi qu'un shell ash [1]. Quelques scripts d'initialisations sont également présents dans /etc, ainsi que les fichiers de périphériques de /dev (le système de fichier ROMFS [22] étant en lecture seule, on ne peut pas utiliser directement d'outils tels que mdev ou udev pour peupler dynamiquement /dev/).

4.1.4 Système de fichier ROMFS

Ce système de fichier basique en lecture seule permet d'utiliser au mieux la mémoire Flash disponible. Il n'utilise presque pas d'espace pour la gestion des fichiers, mais il est uniquement disponible en lecture seule. Il faut utiliser l'outil genromfs [22] afin de générer une image ROMFS. Cet outil transforme une arborescence en fichier prêt à être écrit en

mémoire. Un avantage de ce système de fichier outre sa compacité est son support de l’XIP. Il est ainsi possible d’exécuter le code directement depuis la Flash, sans avoir à le copier en RAM, ce qui fait économiser autant de mémoire vive.

4.2 Solutions existantes

Des systèmes utilisables sur la carte d’évaluation STM32F429 Discovery sont mis à disposition sur des dépôts Git sur GitHub. Le principal est celui de la société Emcraft [12] : <https://github.com/EmcraftSystems/linux-emcraft.git>, l’autre étant celui de l’utilisateur Robutest : <https://github.com/robustest/uclinux.git>. Le dépôt de Robutest contient un système totalement prêt à l’emploi avec des Makefile pour le construire (bootloader, noyau, espace utilisateur), mais le noyau n’est pas le plus récent et n’inclut pas le pilote pour l’IP USB Synopsys utilisé dans le STM32F429. Le dépôt d’Emcraft contient uniquement un noyau un peu plus récent, avec le pilote, mais le noyau n’est pas forcément configuré correctement pour la carte de développement.

Une fusion des deux a été effectuée, en utilisant le noyau d’Emcraft plus récent et la configuration ainsi que le reste du système proposé par Robutest. Cette phase de portage de la configuration n’est pas aussi aisée qu’elle paraît, car des divergences existent entre les deux noyaux, et certaines options ont été ajoutées ou retirées. Ainsi, il a également été nécessaire d’importer les modifications apportées à certains fichiers sources du noyau de Robutest qui n’ont pas été contribuées dans le noyau d’Emcraft et de résoudre divers problèmes de compilation. Un *build system* basé sur CMake a aussi été réécrit en reproduisant les fonctionnalités proposées par le dépôt de Robutest, pour améliorer la gestion des dépendances entre les composants et la facilité de développement d’une application indépendante. On peut maintenant inclure un programme et l’installer dans l’arborescence générée pour la racine avec un simple fichier CMakeLists.txt classique de quelques lignes.

4.3 Outils de développement

Les outils de développement utilisés pour mener ce projet sont le compilateur GCC porté pour ARM Cortex-M, version 2010q1. Cette version de GCC semble correctement compiler le noyau uLinux, ce qui est cohérent avec leurs dates de sortie proche. Une archive du compilateur déjà compilé est disponible à cette adresse : https://sourcery.mentor.com/public/gnu_toolchain/arm-none-linux-gnueabi/\arm-2010q1-202-arm-none-linux-gnueabi-bin.tar.bz2. Ce portage de GCC génère des fichiers ELF [11], qui sont ensuite convertis au format BFLT [6] qui est plus simple et autorise le XIP permettant d’exécuter un programme directement depuis la mémoire flash NOR sans le copier en RAM. Il est également fourni avec une version précompilée de la uClibc [28] avec laquelle il link les programmes par défaut.

Un débogueur GDB est aussi fourni. Celui-ci est un client se connectant à un serveur GDB, qui est en l’occurrence OpenOCD [17]. Il gère la communication avec la sonde SWD ST-LINK/V2 embarquée sur la carte de développement STM32F429 Discovery.

Make est utilisé dans les *build system* de Linux, BusyBox et U-Boot. Il est invoqué par la CMake pour compiler ces programmes qui n’utilisent pas CMake pour leur *build system*.

CMake est utilisé pour orchestrer la compilation des différents composants.

Enfin, la carte de développement STMicroelectronics STM32F429 Discovery a été utilisée afin de travailler sur la partie logicielle avant que notre carte soit prête physiquement.

Afin de gérer les sources de la partie logicielle du projet, un dépôt Git a été créé grâce au compte offert par Assembla [4].

Les documents L^AT_EX des rapports ont été rédigés sur la plate-forme collaborative Overleaf [19] qui dispose d'une interface web collaborative proche de Google Drive (édition simultanée par plusieurs utilisateurs), en éliminant le besoin d'une installation locale de L^AT_EX (Overleaf est basé sur Texlive). Une fonctionnalité intéressante d'Overleaf est qu'un document peut être accédé comme un dépôt Git, permettant une synchronisation dans les deux sens. Ceci permet de travailler hors ligne, et également de faire une sauvegarde à jour et de voir facilement les modifications apportées au cours du temps par les différents utilisateurs.

4.4 Construction du firmware

L'image du firmware prête à être écrite en RAM est donc produite en plusieurs étapes orchestrées avec CMake.

D'abord, les *build system* basés sur des Makefile de uClinux, U-Boot et BusyBox sont appelés par CMake. Ensuite, les programmes de l'utilisateur sont compilés avec CMake. Une phase d'installation permet de peupler un dossier qui sera la racine du système uClinux. Enfin, cette racine est transformée en image ROMFS grâce à genromfs, et on peut utiliser OpenOCD pour flasher le bootloader U-Boot, puis le noyau uClinux, puis l'image ROMFS dans la mémoire flash.

Pour que ces étapes produisent un système fonctionnel, il est impératif que U-Boot soit flashé à une adresse pour laquelle il a été construit, de même que le noyau uClinux. Il faut également que l'image ROMFS soit écrite à l'adresse à laquelle le pilote MTD [15] de Linux s'attend à le trouver.

4.4.1 Paramétrage du noyau uClinux

Le noyau uClinux utilisé étant la fusion du noyau de Robustest (GitHub) et d'Emcraft (*upstream* de Robustest, noyau plus récent), il a fallu adapter les options de configuration et porter les modifications de Robustest qui n'ont pas été remontées à Emcraft. De plus, Robustest a configuré son noyau pour qu'il s'exécute correctement sur la carte d'évaluation STM32F429 Discovery, ce qui n'est pas le cas du noyau d'Emcraft.

Quelques options de configurations permettent de choisir l'adresse à laquelle le noyau s'exécutera pour que l'édition des liens se fasse correctement (le *linker script* tient compte de cette adresse). Cette adresse doit également être connue de U-Boot afin qu'il puisse trouver le noyau et le lancer. D'autres options permettent d'activer les MTD [15] qui sont l'abstraction fournie par le noyau Linux pour les mémoires flash accessibles directement (et non pas via une Flash Translation Layer (FTL)).

4.4.2 Paramétrage du bootloader U-Boot

Le paramétrage de U-Boot est assez direct. L'activation de trop d'options augmente sa taille, mais elles peuvent se révéler pratiques en phase de développement (shell permettant de reprogrammer la mémoire par exemple). Il faut indiquer les options à passer au noyau lors de l'amorçage, ainsi que son adresse en mémoire.

4.4.3 Paramétrage de BusyBox

La dernière version stable de BusyBox (1.23.2) est utilisable. Il faut choisir les options à activer, plus il y en a, plus le binaire final est gros, ce qui peut poser problème sur la carte de développement STM32F429 Discovery qui ne possède seulement que 2Mo de Flash interne.

4.4.4 Utilisation des différentes mémoires

L'utilisation des mémoires suivantes permet d'exploiter au mieux les ressources de la carte :

- Flash interne (2Mo) : Bootloader U-Boot ($\tilde{70}$ Ko) + Noyau uClinux ($\tilde{700}$ Ko)
- RAM interne (256Ko) : Pile du noyau (4Ko)
- NOR externe (32Mo) : Userspace (XIP)
- SDRAM externe (16Mo) : Données des programmes externes et du noyau

La raison est la suivante : la mémoire flash interne est accessible en un cycle, alors que la NOR externe est beaucoup plus longue à être lue. Le contrôleur des mémoires externes FMC du STM32F429 ne peut en outre servir qu'un accès à la fois. Il est donc impossible d'accéder à la SDRAM en même temps qu'à la NOR, contrairement aux RAM et Flash interne d'après l'agencement de la matrice du bus AHB interne du microcontrôleur [27]. Le composant le plus critique en terme de performance étant le noyau, c'est celui-ci qui est mis en flash interne, et sa pile en RAM interne (sa faible taille est idéale, et son accès sera rapide). Ce changement est encore à valider, car il n'est pas l'objet d'une option de configuration, et a peut-être des répercussions non triviales sur d'autres parties du code. Le bootloader réside forcément en Flash interne, car il doit s'exécuter avant l'initialisation du FMC. Enfin, les programmes non critiques sont placés en mémoire NOR externe, et leurs données en SDRAM externe pour des raisons d'espace mémoire.

4.5 Validation du hardware

La validation de la carte peut être réalisée à partir du bootloader U-Boot ainsi que d'OpenOCD. Une connexion d'OpenOCD et une programmation de la flash interne indiquent un bon fonctionnement du microcontrôleur, et la bonne exécution de U-Boot permet de valider une application *bare-metal* raisonnablement simple.

Par ailleurs, il est nécessaire de tester la mémoire SDRAM et NOR afin de vérifier que les informations de timing sont correctement renseignées et que la SDRAM fonctionne de manière prédictible et sans erreur. U-Boot dispose d'une commande `mtest` permettant de tester des plages de mémoires. De la même manière, OpenOCD peut être scripté en Jim TCL afin de réaliser un test de lecture écriture de la mémoire.

4.6 Prévision des performances

Les performances de calcul de cette carte seront selon toute vraisemblance relativement faibles par rapport à une carte intégrant un processeur avec des caches. On a en effet un goulet d'étranglement majeur au niveau du FMC qui est un point de passage des données de la NOR et de la SDRAM externes. De plus, la fréquence maximale à laquelle le FMC peut faire fonctionner la SDRAM externe est $HCLK/2$, avec $HCLK$ l'horloge du bus AHB interne auquel est lié le FMC et qui est aussi l'horloge du cœur [27]. Avec une

fréquence maximale de 180 MHz pour *HCLK*, le transfert du code et des données vers le cœur diviseront probablement par 7 ou 8 la vitesse d'exécution par rapport à l'usage de la RAM et Flash interne.

Cependant, l'expérience de la carte de développement a prouvé que malgré ces chiffres relativement bas, il est tout à fait possible d'avoir un système réactif et sans lenteurs pour un usage de base.

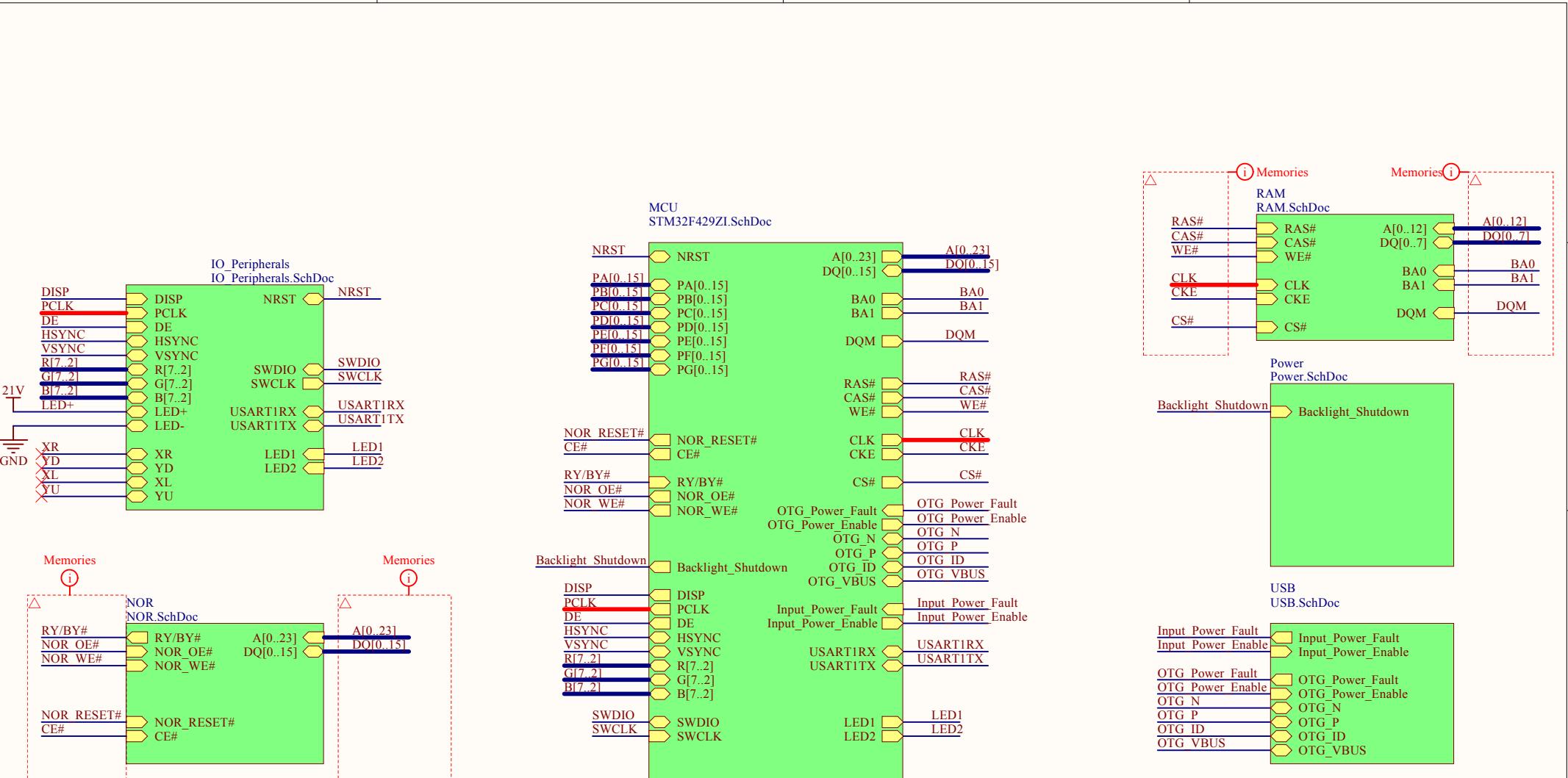
Une amélioration possible serait d'utiliser les nouveaux ARM Cortex M7 qui implantent des mémoires cache dans le but de mieux gérer ce type d'architecture logicielle.

Conclusion

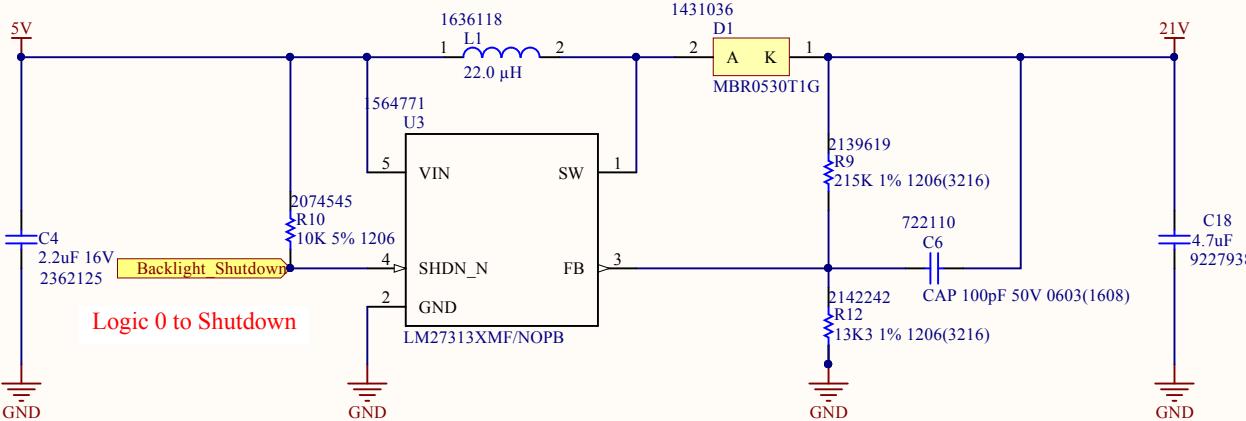
Ce projet nous a permis de nous familiariser avec un certain nombre de composants tels que les mémoires NOR et SDRAM ainsi que les critères de choix qui leur sont associés. Sur le plan matériel, l'intégration de tous les composants nécessaires sur une carte de petite dimension a été un défi et a été l'occasion d'apprendre et de se perfectionner dans l'utilisation d'Altium Designer. Sur le plan logiciel, il a été l'occasion de se pencher sur les composants fondamentaux d'une distribution GNU/Linux ainsi que leur rôle exact et sur la mise en place des outils de développement associés. Il a de plus mis en lumière l'interdépendance entre le logiciel et le matériel, en travaillant en équipe dont un membre est plus orienté sur l'aspect matériel et l'autre sur l'aspect logiciel. Cette diversité de profil s'est révélée efficace et a permis d'avancer ce projet aux multiples facettes.

Nous n'avons pas atteint nos objectifs dans les temps impartis (arriver à démarrer Linux sur notre PCB) parce que nous nous sommes heurtés à des problématiques auxquelles nous ne pensions pas, mais également parce que nous avons voulu faire les choses proprement et approfondir notre compréhension des concepts et phénomènes en jeu. Nous regrettons que ce semestre de 4GE soit aussi dense et que le temps dédié au PRT soit limité par tout le reste. Ce projet sera néanmoins poursuivi par la suite (juste après les partiels) dans le cadre d'un projet personnel.

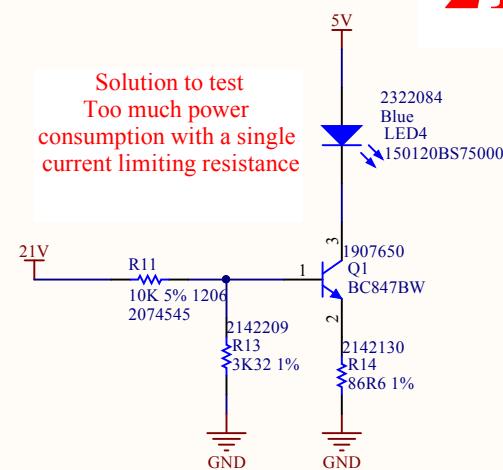
Annexe 1 : Schémas électriques de la carte



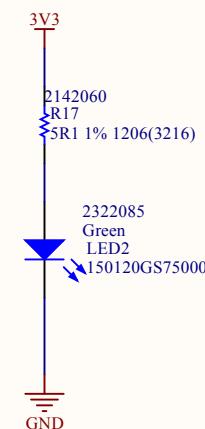
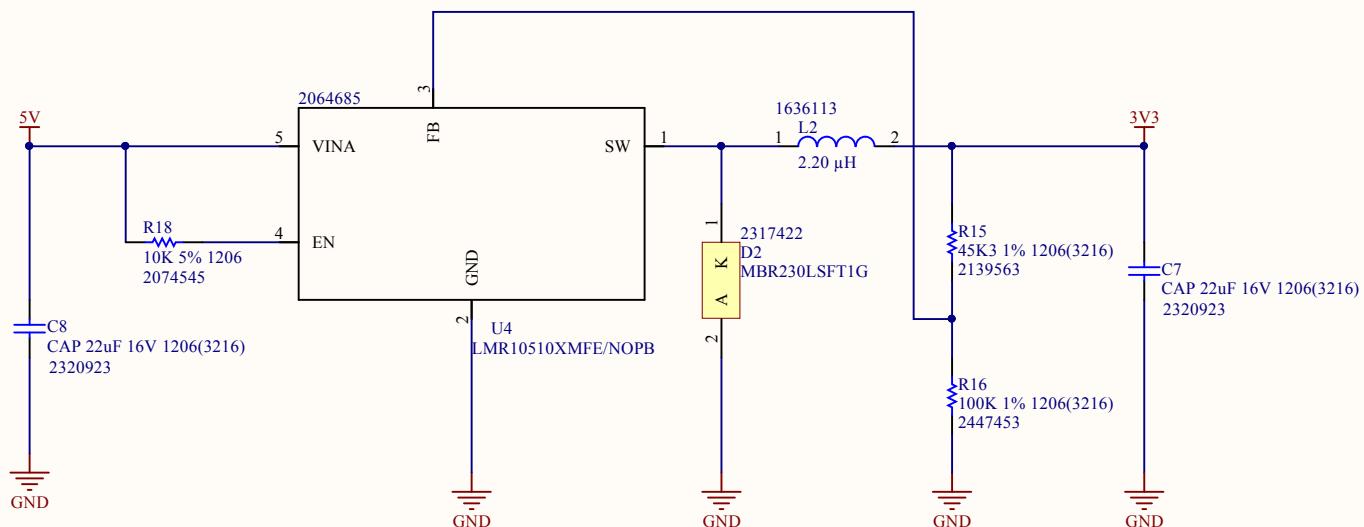
21V



Solution to test
Too much power
consumption with a single
current limiting resistance



3V3



Title Power	Drawn By:Badr BOUSLIKHN	
Size: A4	Number:*	Revision:1.0
Date: 06/06/2015	Time: 17:46:51	Sheet2 of 7
File: C:\Users\Badr\Documents\prt-ge-s2-2015-svn\trunk\Board	02 Schema	BoM Layout\prt STM32F429ZI Eval-Board\Power.SchDoc

INSA Lyon
Departement GE

INSA

INSTITUT NATIONAL
DES SCIENCES
APPLIQUÉES
LYON

1

2

3

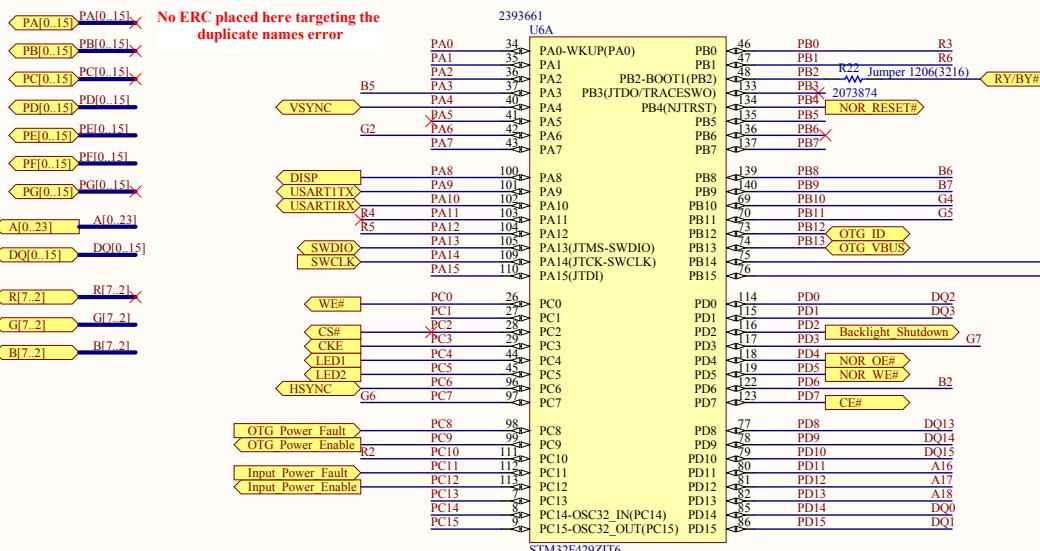
4

1

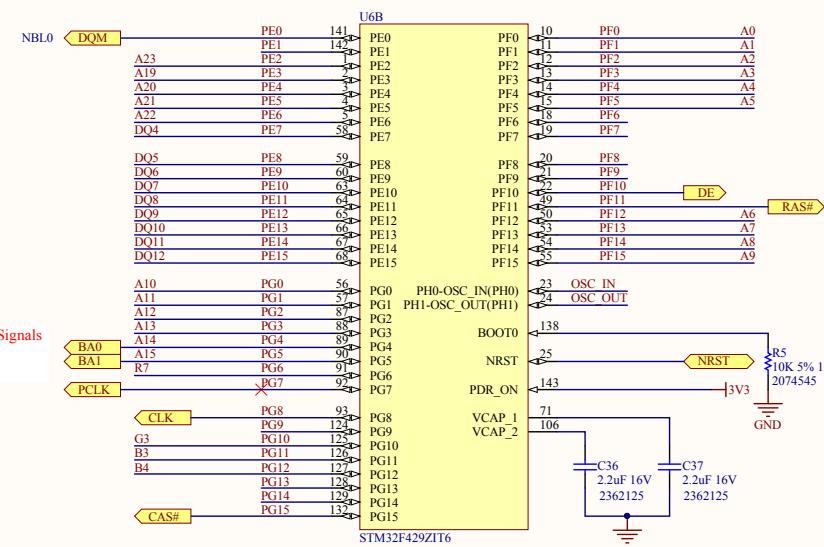
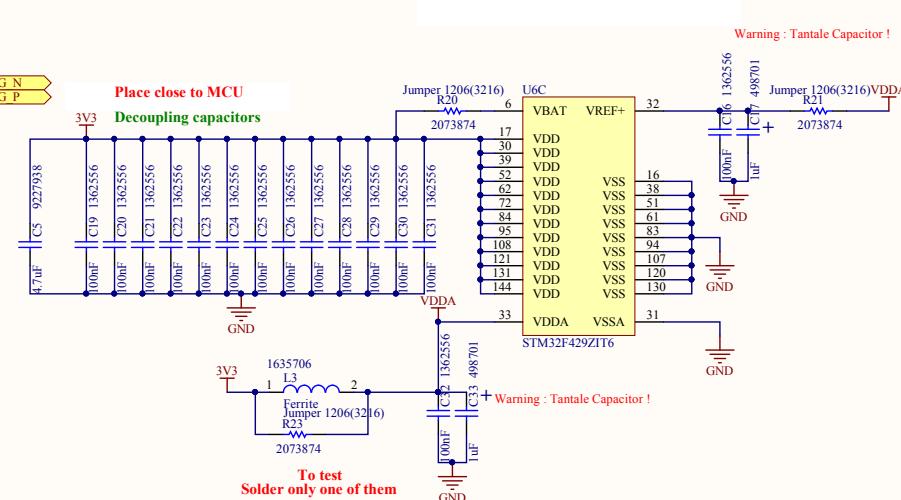
2

3

4

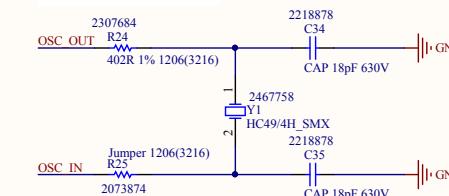


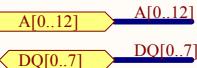
MCU Power



HSE XTAL

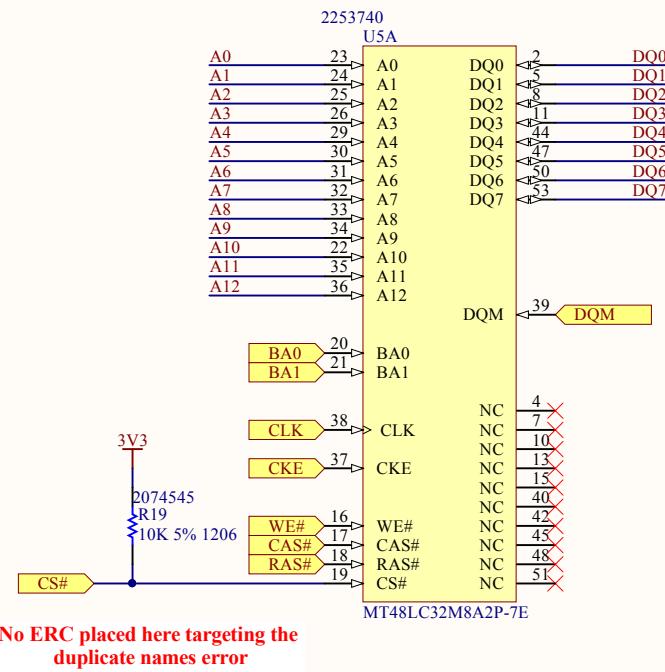
402 Ohm value to tune
5 to 6 times ESR value of
the Quartz





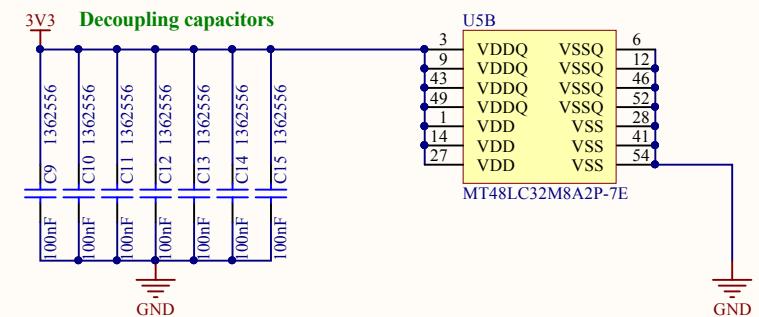
A

A



B

B



C

C

Place close to SDRAM

D

D

Title RAM		Drawn By: Badr BOUSLIKHN	
Size: A4	Number:*	Revision: 1.0	
Date: 06/06/2015	Time: 17:46:52	Sheet 4 of 7	
File: C:\Users\Badr\Documents\prt-ge-s2-2015-svn\trunk\Board	02 Schema	BoM Layout	PRT STM32F429ZI Eval-Board\RAM

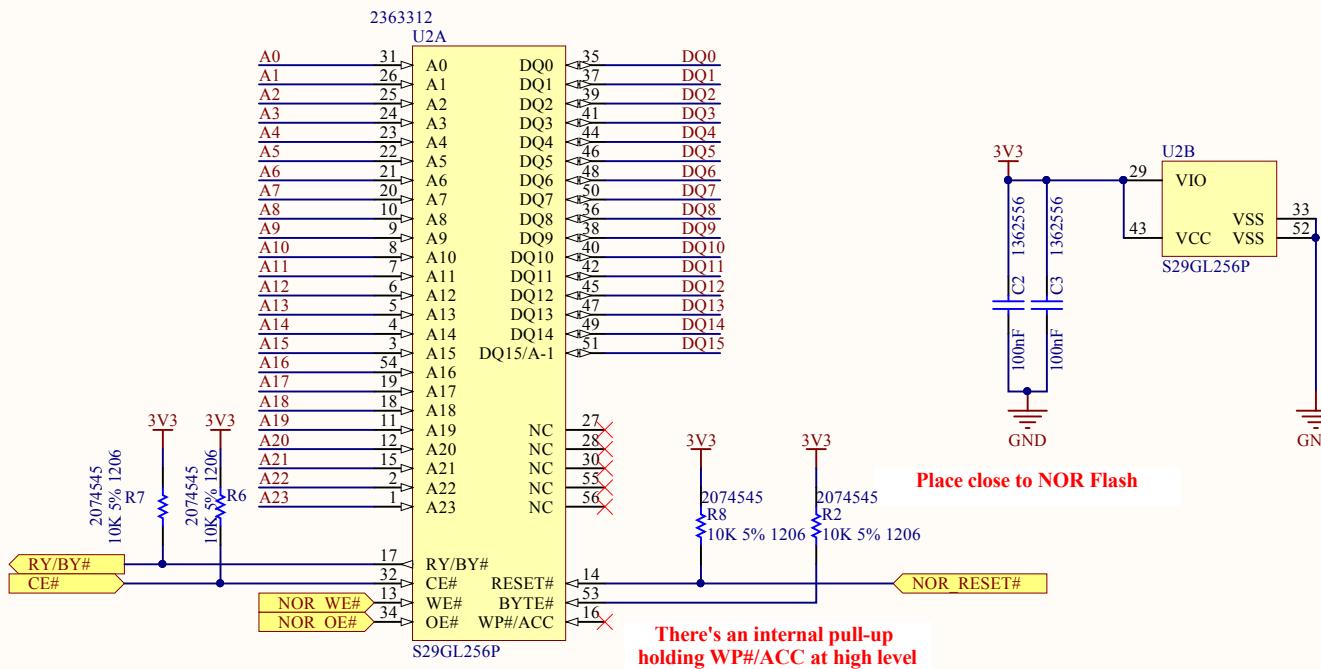
INSA Lyon
Departement GE

INSA

INSTITUT NATIONAL
DES SCIENCES
APPLIQUÉES
LYON

A[0..23] → A[0..23]
DQ[0..15] → DQ[0..15]

A



B

A

B

C

C

D

D

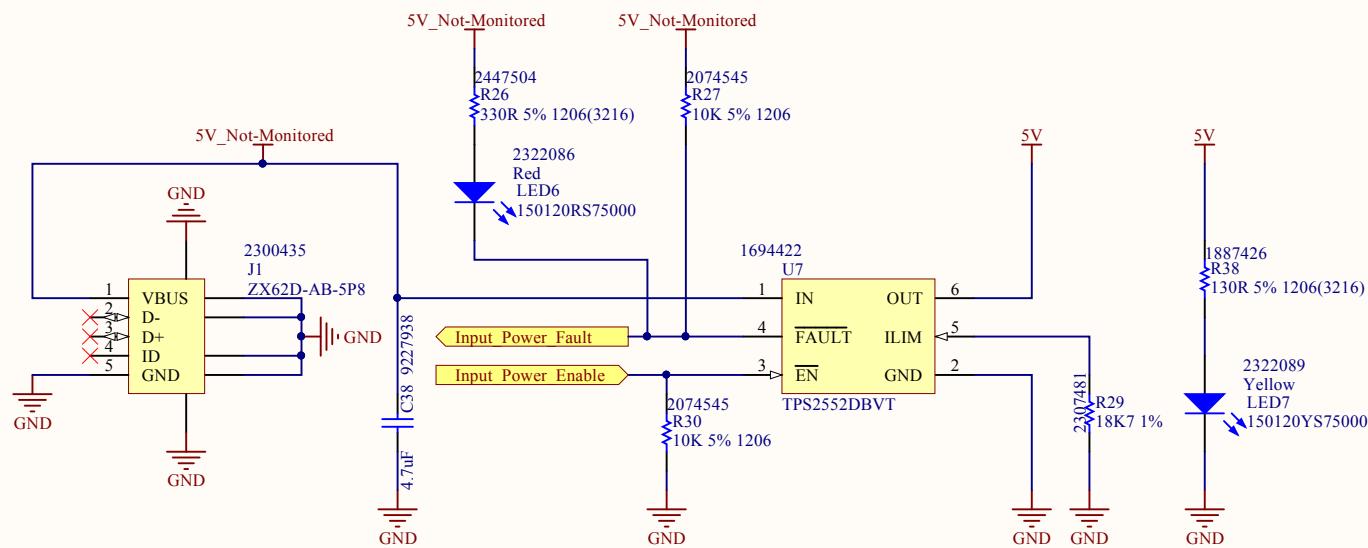
Title NOR		Drawn By:Badr BOUSLIKHN	
Size: A4	Number:*	Revision:1.0	
Date: 06/06/2015	Time: 17:46:52	Sheet5	of 7
File: C:\Users\Badr\Documents\prt-ge-s2-2015-svn\trunk\Board	02 Schema	BoM	Layout\PRT STM32F429ZI Eval-Board\NOR.SchDoc

INSA Lyon
Departement GE

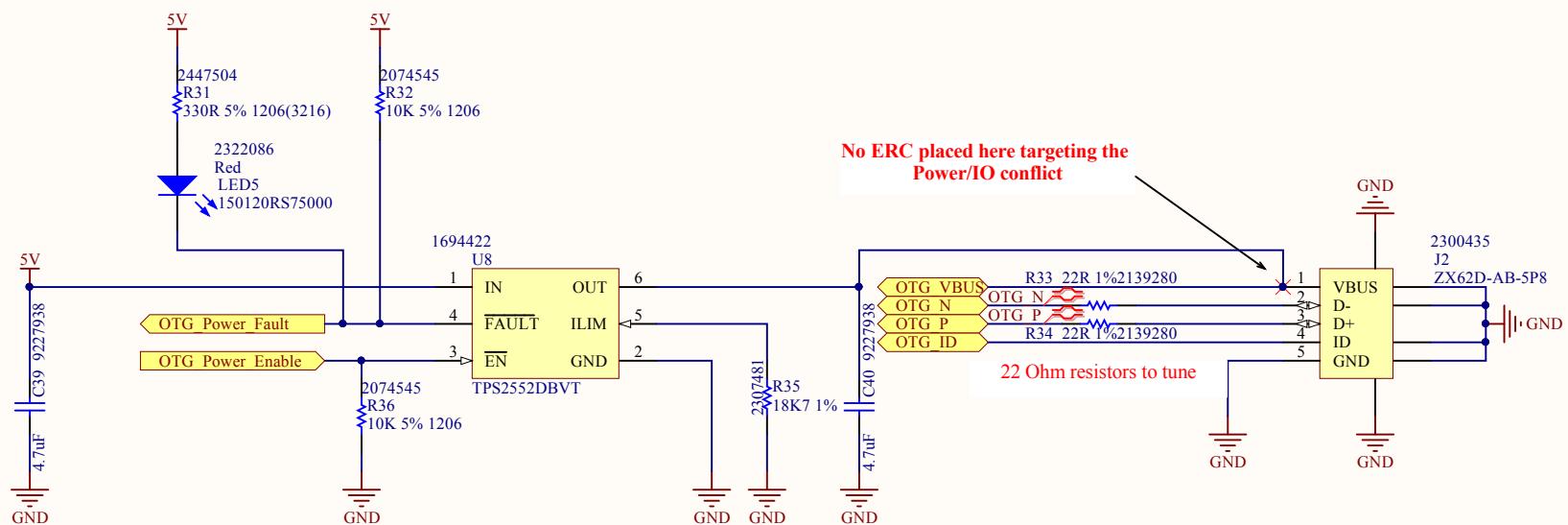
INSA

INSTITUT NATIONAL
DES SCIENCES
APPLIQUÉES
LYON

5V Power Input



USB OTG FS



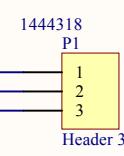
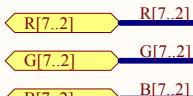
1

2

3

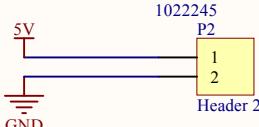
4

A

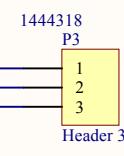


SW Debug port

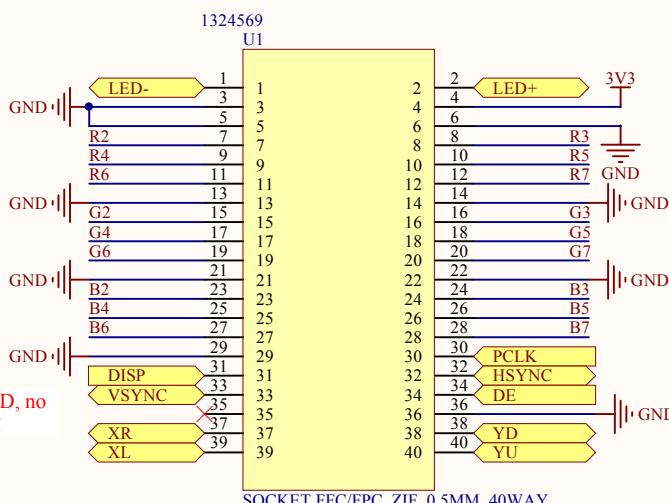
B



Alternate Power Input

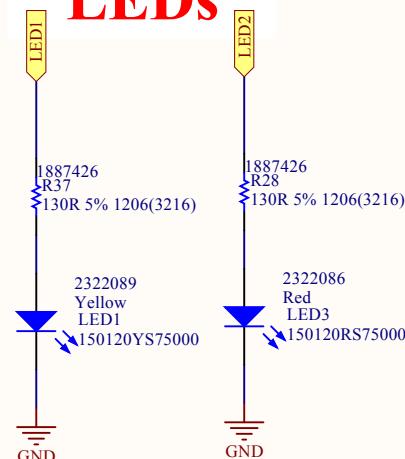


USART1 interface

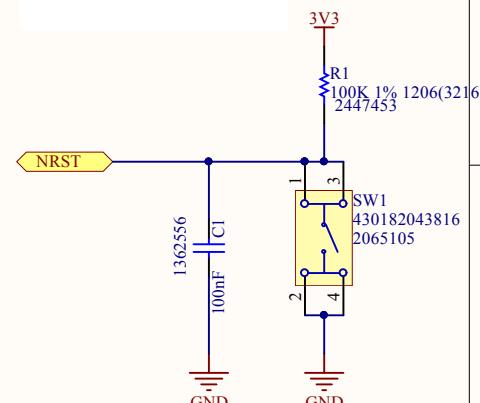


LCD Interface

User LEDs



Reset Switch



Boot mode selection pins		Boot mode	Aliasing
BOOT1	BOOT0		
x	0	Main Flash memory	Main Flash memory is selected as the boot space
0	1	System memory	System memory is selected as the boot space
1	1	Embedded SRAM	Embedded SRAM is selected as the boot space

Annexe 2 : Bill Of Material de la carte

Comment	Description	Designator	Footprint	Quantity	Manufacturer 1	Supplier 1	Supplier Order Qty 1	Supplier Part Number 1	Supplier Unit Price 1 (€)	Total Price (€)
STM32F429ZIT6	ARM Cortex-M4 32-bit MCU+GPU, 225 DMIPS, 2048 kB Flash, 256 kB Internal RAM, 11U6	STM-LQFP144_L		1	STMICROELECTRONICS	Farnell	1 2393661		13,79	
MT48LC32M8A2P-7E	256Mb PC133 SDR SDRAM, 32Mb x 8, 7.5ns, 3.3V, 0 to 70 degC, 54-Pin TSOP, Pb-Free	U5	MIC-T-P-54_N	1	MICRON	Farnell	1 2253740		9,49	
S29GL256P	256Mb NOR	U2	56-Pin_Standard_TSOP	1	SPANSION	Farnell	1 2363312		7,1	
TPS2552DBVT	Adjustable, Active Low, Constant-Current, Current-Limited Power-Distribution Switch, U7, U8	DBV6_L		2	TEXAS INSTRUMENTS	Farnell	2 1694422		3,29	
ZX62D-AB-5P8	USB 2.0 Standard Micro-USB Connector, Micro AB-Standard (Bottom Mount), 30 V A/J1, J2		HIRO-ZX62D-AB-5P8_V	2	HIROSE(HRS)	Farnell	2 2300435		1,6	
SOCKET FFC/FPC, ZIF, 0.5MM, 40WAY	LCD Connector	U1	HIROSE_FH125-40S-0.5SH(55)	1	HIROSE(HRS)	Farnell	1 1324569		2,31	
22.0 µH	SMD-Power Inductor WE-PD4, L = 22.0 µH	L1	WE-PD4-S	1	WURTH ELEKTRONIK	Farnell	1 1636118		2,16	
2,20 µH	SMD-Power Inductor WE-PD4, L = 2,20 µH	L2	WE-PD4-S	1	WURTH ELEKTRONIK	Farnell	1 1636113		2,16	
10K 5% 1206	10K 0.25W 5% 1206 (3216 Metric) SMD	R2, R5, R6, R7, R8	RESC1206(3216)_M	13	MULTICOMP	Farnell	100 2074545		0,0196	
CAP 22uF 16V 1206(3216)	CAP 22uF 16V ±10% 1206 (3216 Metric) Thickness 1.9mm SMD	C7, C8	CAPC1206(3216)190_N	2	MULTICOMP	Farnell	5 2320923		0,312	
100nF	CAP 100nF 50V ±10% 0603 (1608 Metric) Thickness 1mm SMD	C1, C2, C3, C9, C10	CAPC0603(1608)100_N	25	YAGEO (PHYCOMP)	Farnell	25 1362556		0,0519	
Jumper 1206(3216)	Jumper 1206 (3216 Metric)	R20, R21, R22, R23	RESC1206(3216)_L	5	MULTICOMP	Farnell	100 2073874		0,0128	
4.7uF	CAP 4.7uF 50V ±10% 1206 (3216 Metric) Thickness 1.9mm SMD	C5, C18, C38, C39	CAPC1206(3216)190_N	5	KEMET	Farnell	5 9227938		0,254	
LM27313XMF/NOPB	Boost converter	U3	DBV0005A	1	TEXAS INSTRUMENTS	Farnell	1 1564771		1,23	
MBR0530T1G	Schottky Power Rectifier, 2-Pin SOD-123, Pb-Free, Tape and Reel	D1	ONSC-SOD-123-2-425-04_V	1	ON SEMICONDUCTOR	Farnell	5 1431036		0,244	
MBR230LSFT1G	Schottky Power Rectifier, 2-Pin SOD-123, Pb-Free, Tape and Reel	D2	ONSC-SOD-123FL-2-498-01_V	1	ON SEMICONDUCTOR	Farnell	5 2317422		0,223	
LMR10510XMF/NOPB	Buck converter	U4	DBV0005A	1	TEXAS INSTRUMENTS	Farnell	1 2064685		1,11	
1uF	CAP 1uF 16V ±10% 1206 (3216 Metric) Thickness 1.7mm SMD	C17, C33	CAPC1206(3216)170_L	2	AVX	Farnell	5 498701		0,221	
Header 2	Header, 2-Pin	P2	HDR1X2	1	HARWIN	Farnell	10 1022245		0,09	
150120RS75000	SMD mono-color Chip LED, WL-SMCW, Red	LED3, LED5, LED6	1206_A	3	WURTH ELEKTRONIK	Farnell	3 2322086		0,27	
215K 1% 1206(3216)	215K 0.25W 1% 1206 (3216 Metric) SMD	R9	RESC1206(3216)_M	1	VISHAY DRALORIC	Farnell	50 2139619		0,016	
2,2uF 16V	CAP 2,2uF 16V ±20% 1206 (3216 Metric) Thickness 1.9mm SMD	C4, C36, C37	CAPC1206(3216)190_L	3	MURATA	Farnell	10 2362125		0,0798	
45K3 1% 1206(3216)	45K3 0.25W 1% 1206 (3216 Metric) SMD	R15	RESC1206(3216)_L	1	VISHAY DRALORIC	Farnell	50 2139563		0,014	
HC49/4H_SMX	Surface Mount Quartz Crystal	Y1	HC49/4H_SMX	1	ABRACON	Farnell	1 2467758		0,649	
150120RS75000	SMD mono-color Chip LED, WL-SMCW, Yellow	LED1, LED7	1206_A	2	WURTH ELEKTRONIK	Farnell	2 2322089		0,312	
430182043816	WS-TASV SMD Tact Switch 6x6 mm	SW1	430182043816	1	WURTH ELEKTRONIK	Farnell	1 2065105		0,624	
130R 5% 1206(3216)	130R 0.25W 5% 1206 (3216 Metric) SMD	R28, R37, R38	RESC1206(3216)_L	3	MULTICOMP	Farnell	25 1887426		0,0243	
CAP 100pF 50V 0603(1608)	CAP 100pF 50V ±10% 0603 (1608 Metric) Thickness 1mm SMD	C6	CAPC0603(1608)100_L	1	YAGEO (PHYCOMP)	Farnell	10 722110		0,06	
CAP 18pF 630V	CAP 18pF 500V ±5% 1206 (3216 Metric) Thickness 1.7mm SMD	C34, C35	CAPC1206(3216)170_L	2	MURATA	Farnell	2 2218878		0,254	
5R1 1% 1206(3216)	5R1 0.25W 1% 1206 (3216 Metric) SMD	R17	RESC1206(3216)_N	1	MULTICOMP	Farnell	10 2142060		0,036	
150120SG75000	SMD mono-color Chip LED, WL-SMCW, Green	LED2	1206_A	1	WURTH ELEKTRONIK	Farnell	1 2322085		0,312	
150120BS75000	SMD mono-color Chip LED, WL-SMCW, Blue	LED4	1206_A	1	WURTH ELEKTRONIK	Farnell	1 2322084		0,312	
402R 1% 1206(3216)	402R 0.25W 1% 1206 (3216 Metric) SMD	R24	RESC1206(3216)_L	1	PANASONIC ELECTRONIC COMPONENTS	Farnell	10 2307684		0,03	
18K7 1%	18K7 0.25W 1% 1206 (3216 Metric) SMD	R29, R35	RESC1206(3216)_L	2	PANASONIC ELECTRONIC COMPONENTS	Farnell	10 2307481		0,03	
BC847BW	NPN Silicon AF Transistor, 45 V VCEO, 100 mA IC, SOT323, Reel, Green	Q1	INF-SOT323_N	1	NXP	Farnell	5 1907650		0,0508	
13K3 1% 1206(3216)	13K3 0.25W 1% 1206 (3216 Metric) SMD	R12	RESC1206(3216)_L	1	MULTICOMP	Farnell	10 2142242		0,024	
3K32 1%	3K32 0.25W 1% 1206 (3216 Metric) SMD	R13	RESC1206(3216)_L	1	MULTICOMP	Farnell	10 2142209		0,024	
86R6 1%	86R6 0.25W 1% 1206 (3216 Metric) SMD	R14	RESC1206(3216)_N	1	MULTICOMP	Farnell	10 2142130		0,024	
22R1 1%	22R1 0.1W 1% 0603 (1608 Metric) SMD	R33, R34	RESC0603(1608)_M	2	VISHAY DRALORIC	Farnell	10 2139280		0,0173	
Ferrite	SMD EMI Suppression Ferrite Bead WE-CBF, Z = 600 Ohm	L3	SMD-0603	1	WURTH ELEKTRONIK	Farnell	1 1635706		0,166	
Header 3	Header, 3-Pin	P1, P3	HDR1X3	2	MOLEX	Farnell	2 1444318		0,075	
100K 1% 1206(3216)	100K 0.25W 1% 1206 (3216 Metric) SMD	R1, R16	RESC1206(3216)_L	2	MULTICOMP	Farnell	10 2447453		0,005	
330R 5% 1206(3216)	330R 0.25W 5% 1206 (3216 Metric) SMD	R26, R31	RESC1206(3216)_M	2	MULTICOMP	Farnell	10 2447504		0,0045	

Total 70,7 €

Glossaire

(Reduced) Media-independent interface est une interface standard pour le protocole Ethernet permettant la communication entre le transceiver, contrôleur gérant la stack Ethernet et le PHY drivant les lignes physiques. 7, 46

Advanced High-performance Bus est le bus principal interne de nombreux micro-contrôleurs ARM (Cortex 3/4). 13, 45

AHB Advanced High-performance Bus. 13, 33, 45, *Glossaire* : Advanced High-performance Bus

CFI Common Flash Interface. 14, 22, 45, *Glossaire* : Common Flash Interface

Common Flash Interface est une interface standardisée par le JEDEC pour interroger les mémoires NOR à propos des valeurs de temporisations critiques telles que la durée nécessaire à l'écrasement d'un secteur. 14, 45

Digital Signal Processor en anglais Digital Signal Processor est une microprocesseur à part entière ou une unité dans un microprocesseur/microcontrôleur optimisé pour exécuter des applications de traitement numérique du signal (filtrage, extraction de signaux, etc.) le plus rapidement possible. Des instructions particulière comme le MAC (Multiply and Accumulate) sont au cœur des DSP. 6, 45

DSP Digital Signal Processor. 6, 7, 45, *Glossaire* : Digital Signal Processor

Equivalent Series Inductance est une valeur une inductance série d'un condensateur. 21, 45

Equivalent Series Resistance est une grandeur modélisant toutes les pertes présentes dans le condensateur exprimées sous la forme d'une résistance série. 21, 45

ESL Equivalent Series Inductance. 21, 45, *Glossaire* : Equivalent Series Inductance

ESR Equivalent Series Resistance. 21, 45, *Glossaire* : Equivalent Series Resistance

eXecute In Place permet d'exécuter un programme directement depuis une mémoire flash NOR (mode de lecture mot par mot) sans avoir besoin de le copier en RAM d'abord comme il serait nécessaire de le faire si la mémoire était de la NAND. 13, 47

Flash Translation Layer est une couche d'abstraction entre les blocs physiques d'une mémoire Flash et les blocs logiques présentés à l'utilisateur. Elle est couramment employée afin d'implémenter le wear leveling permettant de répartir l'usure des secteurs sur la totalité de la mémoire pour prolonger sa durée de vie. 32, 46

Flexible Memory Controller permet au microcontrôleur STM32F429 d'accéder à des mémoires externes entre autre NOR et SDRAM. Il ne peut servir qu'une à la fois, ce qui empêche d'accéder simultanément à la RAM externe et à la Flash externe, ce qui est un goulet d'étranglement. 7, 45

Floating Point Unit est une unité de calcul en virgule flottante. C'est une partie d'un processeur, spécialement conçue pour effectuer des opérations sur des nombres à virgule flottante. Elle est très utile pour accélérer certains algorithmes. 6, 45

FMC Flexible Memory Controller. 7, 11, 13, 30, 33, 45, *Glossaire* : Flexible Memory Controller

FPU Floating Point Unit. 6, 7, 45, *Glossaire* : Floating Point Unit

FTL Flash Translation Layer. 32, 46, *Glossaire* : Flash Translation Layer

General Purpose Input/Output permettent à une carte électronique la possibilité de communiquer avec d'autres circuits électroniques pour des usages divers et variés. 5, 46

Gerber est le standard le plus utilisé pour transmettre des informations concernant la fabrication des circuits imprimés. Il contient la description des diverses couches de connexions électriques (pistes, pastilles, *footprint* des CMS, les vias...). 17

GPIO General Purpose Input/Output. 5, 16, 46, *Glossaire* : General Purpose Input/Output

IC Integrated Circuit. 1, 19

Intellectual Property est un bloc implémentant une fonction au sein d'un SoC. Un microcontrôleur moderne possède donc un grand nombre d'IP pour ses périphériques. 6, 46

IP Intellectual Property. 6, 31, 46, *Glossaire* : Intellectual Property

Joint Test Action Group Joint Test Action Group est le nom de la norme IEEE 1149.1 intitulée « Standard Test Access Port and Boundary-Scan Architecture ». La technique de Boundary-Scan (littéralement, scrutation des frontières) est conçue pour faciliter et automatiser le test des cartes électroniques numériques. Elle consiste à donner un accès auxiliaire aux broches d'entrée-sortie des composants numériques fortement intégrés. 7, 46

JTAG Joint Test Action Group. 7, 46, *Glossaire* : Joint Test Action Group

M2M Machine to machine. 46, *Glossaire* : Machine to machine

Machine to machine désigne la communication entre machines. 46

Memory Management Unit permet à un processeur d'implémenter la mémoire virtuelle. La MMU permet ainsi de mapper des pages de mémoire virtuelle sur des pages de mémoire physique. 29, 46

Memory Protection Unit permet à un processeur de protéger en écriture, lecture ou exécution certaines zones de la mémoire. 29, 46

MII/RMII (Reduced) Media-independent interface. 7, 46, *Glossaire* : (Reduced) Media-independent interface

MMU Memory Management Unit. 29, 46, *Glossaire* : Memory Management Unit

MPU Memory Protection Unit. 29, 46, *Glossaire* : Memory Protection Unit

PCB Printed Circuit Board. 1, 23, 24, 34

PHY est une abréviation couramment utilisée pour désigner la couche physique du modèle OSI (couche de plus bas niveau). 7

PMU Power Management Unit. 1, 46, *Glossaire* : Power Management Unit

Power Management Unit est un circuit intégré générant les tensions nécessaires au bon fonctionnement des autres circuits intégrés sur une carte, et séquence notamment les différentes alimentations du processeur lors de sa phase de démarrage. 1, 46

Single Wire Debug est un standard de debug des processeurs ARM avec des fonctionnalités de debug proches du JTAG, mais en utilisant très peu de signaux (une horloge, un signal de données et une masse). 7, 47

SoC System on Chip. 6, 46, *Glossaire* : System on Chip

SWD Single Wire Debug. 7, 31, 47, *Glossaire* : Single Wire Debug

System on Chip est un circuit intégré implémentant un cœur de processeur ainsi que tous les périphériques nécessaires à son fonctionnement tel que de la mémoire vive et ROM, ainsi que des fonctions plus avancées comme des contrôleurs de bus (USB, I2C, etc.). 6, 46

THUMB est un encodage des instructions ARM compact de taille variable. La plupart des instructions sont codées sur 16bits et d'autres plus rares sur 32bits, ce qui permet de gagner de la place et d'augmenter la vitesse d'exécution en rapportant la taille d'une instruction à la largeur habituelle du bus des mémoires NOR. 13

ULPI UTMI+ low pin interface. 7, 47, *Glossaire* : UTMI+ low pin interface

UTMI+ low pin interface est une interface standard pour le protocole économique en signaux permettant la communication entre le transceiver, contrôleur gérant la stack USB et le PHY drivant les lignes physiques D+/D-. 7, 47

XIP eXecute In Place. 13, 31, 47, *Glossaire* : eXecute In Place

Références

- [1] *Almquist Shell*. URL : <http://www.in-ulm.de/~mascheck/various/ash/#busybox>.
- [2] *Altium Designer*. URL : <http://www.altium.com/>.
- [3] *ARM Holdings eager for PC and server expansion*. URL : http://www.theregister.co.uk/2011/02/01/arm_holdings_q4_2010_numbers/.
- [4] *Assembla*. URL : <https://www.assembla.com/>.
- [5] *BeagleBone Black*. URL : <http://beagleboard.org/black>.
- [6] *Binary FLaT format*. URL : <http://retired.beyondlogic.org/uLinux/bflt.htm>.
- [7] *BusyBox*. URL : <http://www.busybox.net>.
- [8] *Cirly*. URL : <http://www.cirly.com/>.
- [9] *Connectivity and Multi-Sheet Design*. URL : <http://techdocs.altium.com/display/ADOH/Connectivity+and+Multi-Sheet+Design>.
- [10] *Differences between uLinux and Linux*. URL : http://blackfin.ulinux.org/doku.php?id=ulinux-dist:difference_from_linux.
- [11] *ELF generic ABI in System V ABI*. URL : <http://www.sco.com/developers/gabi/>.
- [12] *Emcraft*. URL : <http://www.emcraft.com/>.
- [13] *Euroserver project*. URL : <http://www.euroserver-project.eu/>.
- [14] *Joint Electron Device Engineering Council*. URL : <https://www.jedec.org/>.
- [15] *Memory Technology Device*. URL : http://www.linux-mtd.infradead.org/doc/general.html#L_overview.
- [16] *NHD-4.3-480272MF-ATXI#-T-1 Datasheet*. Rev. 7. Newhaven Display. Mar. 2012. URL : <http://www.newhavendisplay.com/specs/NHD-4.3-480272MF-ATXI-T-1.pdf>.
- [17] *OpenOCD*. URL : <http://openocd.org/>.
- [18] *OTA5180A LCD driver Datasheet*. Version 0.2. Orise. Avr. 2008. URL : http://www.newhavendisplay.com/app_notes/OTA5180A.pdf.
- [19] *Overleaf*. URL : <https://www.overleaf.com/>.
- [20] *Page Wikipédia d'Altium Designer*. URL : https://www.wikiwand.com/fr/Altium_Designer.
- [21] *Raspberry Pi*. URL : <https://www.raspberrypi.org/products/raspberry-pi-2-model-b/>.
- [22] *ROM File System*. URL : <http://romfs.sourceforge.net/>.
- [23] *S29GL256P Datasheet*. S29GL-P_00_A14. Spansion. Oct. 2012. URL : <http://www.farnell.com/datasheets/1756769.pdf>.
- [24] *SDR SDRAM Datasheet*. Rev. W 09/14 EN. Micron. Sept. 2014. URL : <http://www.farnell.com/datasheets/1674459.pdf>.

- [25] *Sick of Beige standard PCB sizes v1.0.* URL : http://dangerousprototypes.com/docs/Sick_of_Beige_standard_PCB_sizes_v1.0.
- [26] *STM32CubeMX.* URL : http://www.st.com/web/catalog/tools/FM147/CL1794/SC961/SS1743/PF259242?icmp=stm32cubemxpron_pr-stm32cubef2_apr2014&sc=stm32cube-pr2.
- [27] *STM32F429XX Reference manual.* DocID018909 Rev 8. ST Microelectronics. Oct. 2014. URL : http://www.st.com/web/en/resource/technical/document/reference_manual/DM00031020.pdf.
- [28] *uClibc.* URL : <http://www.uclibc.org/>.