Architecture des systèmes à processeur

Prof. Géraldine Conti

Basé sur les cours des Profs. Sanchez, Starkier, Mosqueron et Dassatti

1

Le processeur embarqué

2

Le processeur embarqué

- Aussi nommé microcontrôleur
- Constitué de :
  - un cœur de processeur (ou plusieurs)
  - des périphériques (entrées /sorties ou IOs)
    - Contrôleur écran
    - Contrôleur USB
  - de la mémoire

- Contrainte de performance, notion de temps réel
- Problème de la densité de code : taille du code applicatif (mémoire), la mémoire tend à occuper la majeure partie d'une puce, impact de la taille du coeur dans le choix du processeur

3

OMAP/DM3730

- **OMAP** = Open Multimedia Applications Platform
- C'est un **SoC**

4

Environnement

- Mémoires MMC
- Interfaces utilisateur
- Image
- Wireless
- Audio
- …

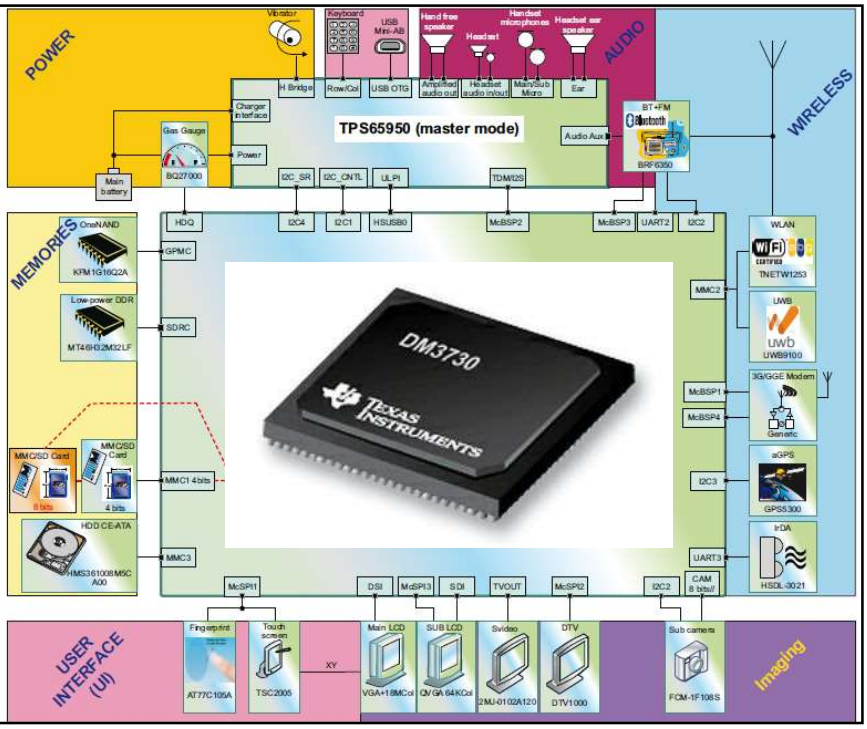
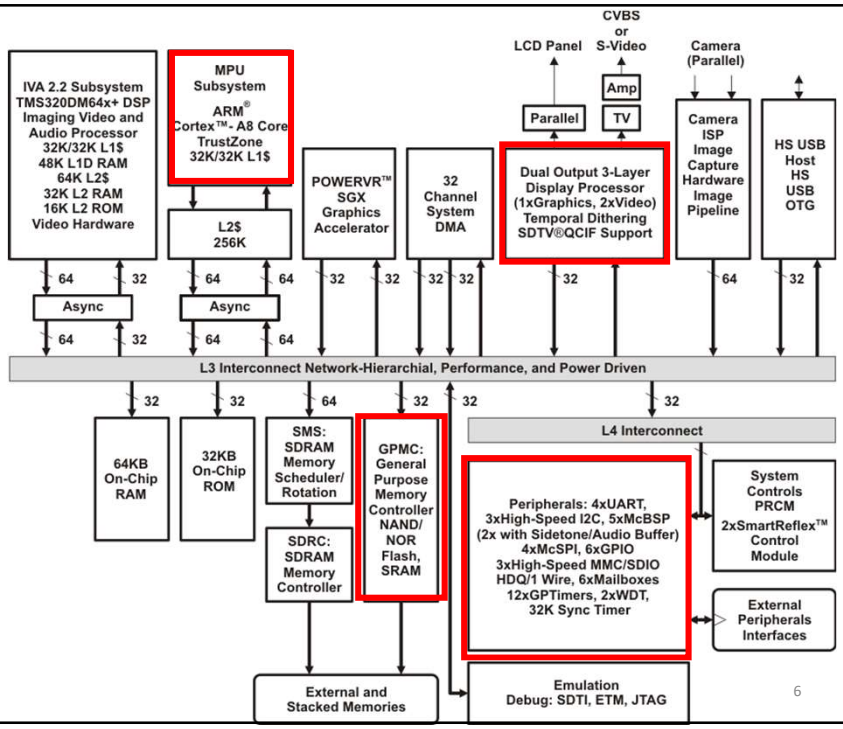


Schéma Bloc



Accès aux IOs

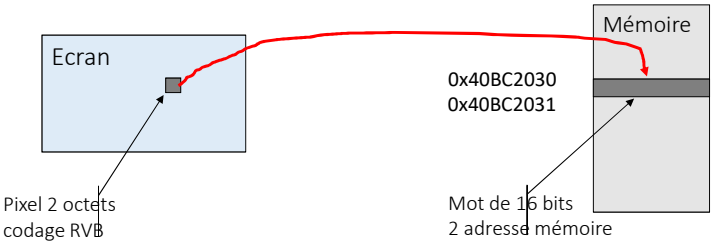
- Accessibles comme la mémoire en lecture et écriture (**plan d'adressage**)
- Un périphérique d'entrées /sorties est vu comme un ensemble de registres avec une adresse mémoire pour chaque registre => **memory mapped Ios**

Adresse		DM3730	REPTAR	Description
0x0000 0000	0x000F FFFF	Boot space		Boot space 1MB
0x0010 0000	0x17FF FFFF	GPMC		non utilisé
0x1800 0000		CS3 128 MB	FPGA	Local bus mode asynchrone
	0x19FF FFFF		SP6, 32MB	
0x1A00 0000	0x1BFF FFFF		FPGA SP3, 32MB	
0x1C00 0000	0x1FFF FFFF		Inaccessible 64 MB	
0x2000 0000	0x23FF FFFF	CS4 128 MB	FPGA SP6 64MB	Local bus mode synchrone
0x2400 0000	0x27FF FFFF		Inaccessible 64 MB	
0x2800 0000	0x2BFF FFFF			non utilisé
0x2C00 0000	0x2FFF FFFF	CS5 64MB	Ethernet	
0x3000 0000	0x3FFF FFFF	CS0 256 MB	Nand Flash	
0x4000 0000	0x400F FFFF	On-chip		Boot ROM interne 1MB
0x4010 0000		memory		
	0x47FF FFFF	128MB		
0x4800 0000	0x48FF FFFF	L4-Core		
0x4900 0000	0x49DF FFFF	L4-Per		
0x49E0 0000	0x67FF FFFF			
0x6800 0000		L3		
	0x6FFF FFFF	interconnect		
0x7000 0000	0x7FFF FFFF			
0x8000 0000		SDRAM (1GB)	DDR ?	
	0xBFFF FFFF	controller		
0xC000 0000	0xFFFF FFFF			

7

Labo Etape 1 : Display Subsystem

- Contrôleur de **l'affichage**
- Frame Buffer
  - mémoire d'affichage utilisé par le processeur (écriture)
  - un emplacement de la mémoire correspond un **pixel**
  - Un pixel utilise 1, 2 ou 4 octets de la mémoire par exemple 5 bits rouge, 6 bits vert, et 5 bits bleu
- Exemple :

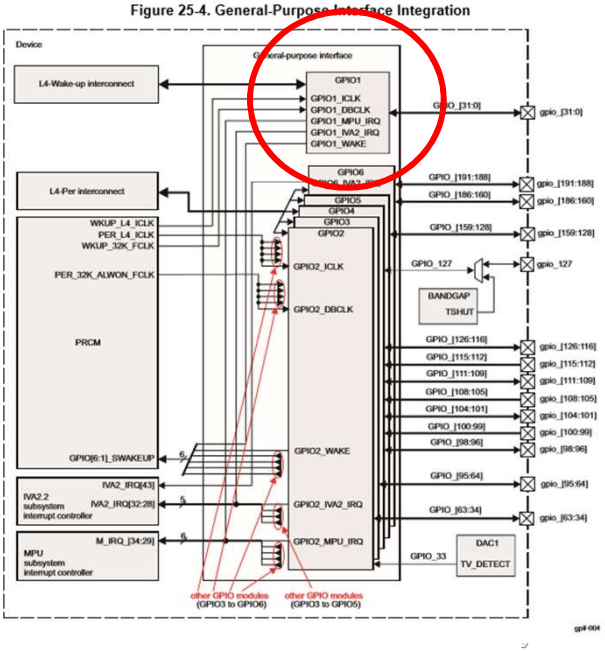


8

Labo Etape 2 : GPIOs

- Périphérique le plus courant: le **GPIO** (General Purpose IO)
- 6 **banques** nommées : GPIO1-6
  - Le GPIO1 est utilisé pour le Wake-Up
- 32 pins par banque, un total de **192 pins** disponibles

Figure 25-4. General-Purpose Interface Integration



Les banques de GPIOs

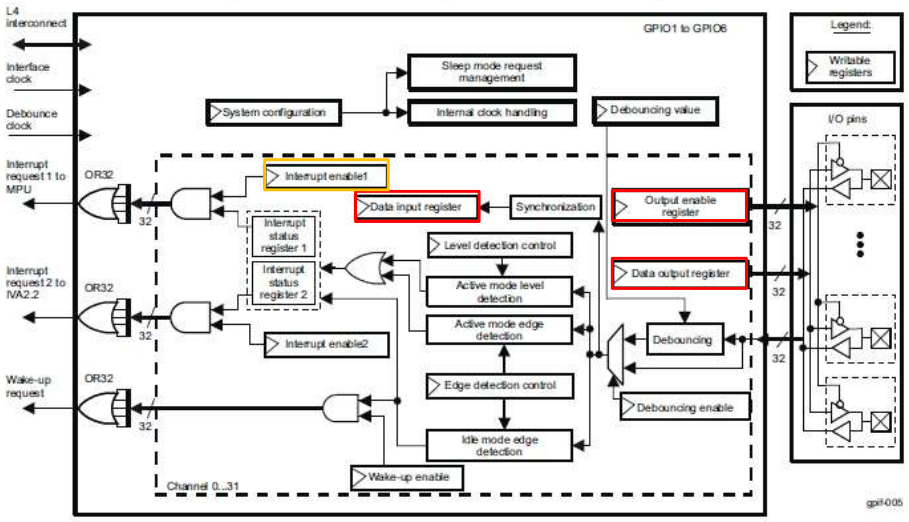
- Dans chaque banque, il y a les mêmes **26 registres de config**
  - Exemple pour GPIO4
- **Adresse** = Adresse_GPIO4(0x49054000) + Offset
  - **L'offset** reste le même entre les différents banques
- GPIO_CLEARDATAOUT permet de faire directement un clear de bit à bit

Table 25-10. GPIO4 Register Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address
GPIO_REVISION	R	32	0x000	0x4905 4000
GPIO_SYSCONFIG	RW	32	0x010	0x4905 4010
GPIO_SYSSTATUS	R	32	0x014	0x4905 4014
GPIO_IRQSTATUS1	RW	32	0x018	0x4905 4018
GPIO_IRQENABLE1	RW	32	0x01C	0x4905 401C
GPIO_WAKEUPENABLE	RW	32	0x020	0x4905 4020
GPIO_IRQSTATUS2	RW	32	0x028	0x4905 4028
GPIO_IRQENABLE2	RW	32	0x02C	0x4905 402C
GPIO_CTRL	RW	32	0x030	0x4905 4030
GPIO_OE	RW	32	0x034	0x4905 4034
GPIO_DATAIN	R	32	0x038	0x4905 4038
GPIO_DATAOUT	RW	32	0x03C	0x4905 403C
GPIO_LEVELDETECT0	RW	32	0x040	0x4905 4040
GPIO_LEVELDETECT1	RW	32	0x044	0x4905 4044
GPIO_RISINGDETECT	RW	32	0x048	0x4905 4048
GPIO_FALLINGDETECT	RW	32	0x04C	0x4905 404C
GPIO_DEBOUNCENABLE	RW	32	0x050	0x4905 4050
GPIO_DEBOUNCINGTIME	RW	32	0x054	0x4905 4054
GPIO_CLEARIRQENABLE1	RW	32	0x060	0x4905 4060
GPIO_SETIRQENABLE1	RW	32	0x064	0x4905 4064
GPIO_CLEARIRQENABLE2	RW	32	0x070	0x4905 4070
GPIO_SETIRQENABLE2	RW	32	0x074	0x4905 4074
GPIO_CLEARWKUENA	RW	32	0x080	0x4905 4080
GPIO_SETWKUENA	RW	32	0x084	0x4905 4084
GPIO_CLEARDATAOUT	RW	32	0x090	0x4905 4090
GPIO_SETDATAOUT	RW	32	0x094	0x4905 4094

10

Détail des GPIOs



Configuration des pads

- Usage multiple des **pins (pads)** du processeur
  - GPIOs ou entrées sorties de périphériques
- Une configuration « **padconf** » pour chaque pin
  - sélection IO par le paramètre « **muxmode** »(3 bits)
  - configuration physique de la pin : pull-up, pull-down, wake-up

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WAKEUPEVENT0	WAKEUPENABLE0	OFFPULLTYPESELECT0	OFFPULLUDENABLE0	OFFOUTVALUE0	OFFOUTENABLE0	OFFENABLE0	INPUTENABLE0	RESERVED			PULLTYPESELECT0	PULLUDENABLE0	MUXMODE0		

Exercice (Labo Etape 2)

On veut allumer la LED0 → Routage sur la carte → #pin =

1. Configuration des GPIOs

GPIO ?
Bit à changer ?

2. Configuration des pads (tab. 13.4)

VAR SOM pin	DM3730 GPIO_	DM3730 module/bit	Schematic reference	Board reference	Component description
PUSH-BUTTONS					
65	140		SW1	SW0	Push button
69	142		SW2	SW1	Push button
57	167		SW3	SW2	Push button
59	97		SW4	SW3	Push button
62	126		SW5	SW4	Push button
LEDs					
71	143		D2	LED0	Green LED
67	141		D3	LED1	Green LED
56	101		D11	LED2	Green LED
55	102		D14	LED3	Green LED

Nom du registre	Adr. physique	Mode0	Mode1	Mode2	Mode3	Mode4	Mode5	Mode6	Mode7
CONTROL_PADCONF_MCBSP3_DX[31:16]	0x4800 216C	mcbsp3_dr	uart2_rts			gpio_141	hsusb3_tll_data5		safe_mode
CONTROL_PADCONF_MCBSP3_CLKX[15:0]	0x4800 2170	mcbsp3_clkx	uart2_tx			gpio_142	hsusb3_tll_data6		safe_mode
CONTROL_PADCONF_MCBSP3_CLKX[31:16]	0x4800 2170	mcbsp3_fsx	uart2_rx			gpio_143	hsusb3_tll_data7		safe_mode
CONTROL_PADCONF_UART2_CTS[15:0]	0x4800 2174	uart2_cts	mcbsp3_dx	gpt_9_pwm_evt		gpio_144			safe_mode

Labo Etape 2 : Driver

- **Initialise** un périphérique
  - écriture dans les registres de configuration
- **Communique** avec le périphérique
  - écrit ou lit des données (par registres ou zone mémoire)
  - contrôle le périphérique (par registre)
- **Masque la complexité et facilite l'interfaçage**
  - pas d'accès direct aux registres pour l'utilisateur
  - bibliothèque de fonction simples(API – Application Programming Interface)

15

Interruptions et Modes

16

Interruptions

- **Périphériques :**
  - *timer, clavier, entrées /sorties (port série, parallèle, USB, disques, ….)*
- **Erreurs systèmes :**
  - *erreurs mémoires, division par zéro, …*
- **Interruption processeur (IPI) :**
  - **interruption par un autre processeur (système multi-processeurs)**

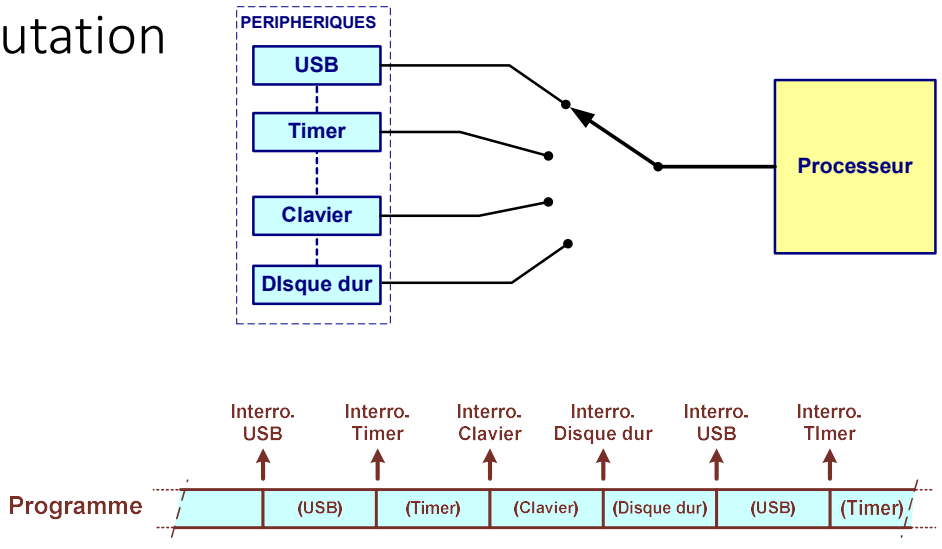
17

Gestion des interruptions

- **5 modes :**
  - Sans interruption (scrutation)
  - Interruptions multi-sources sans identification de la source
  - Interruptions multi-sources avec identification de la source
  - Interruptions chaînées
  - Interruptions imbriquées

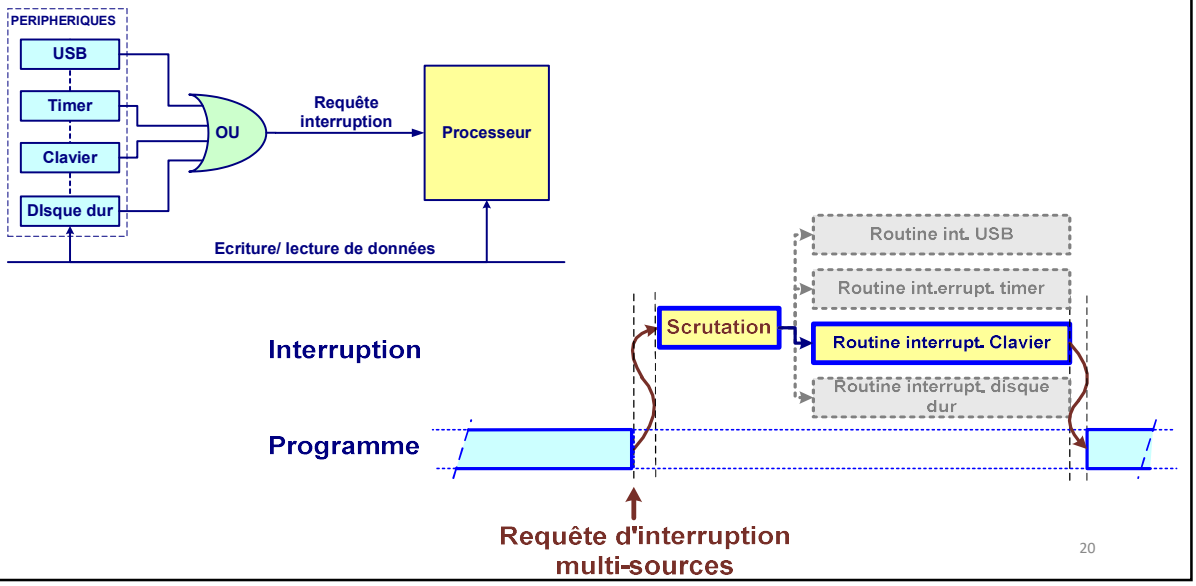
18

Scrutation



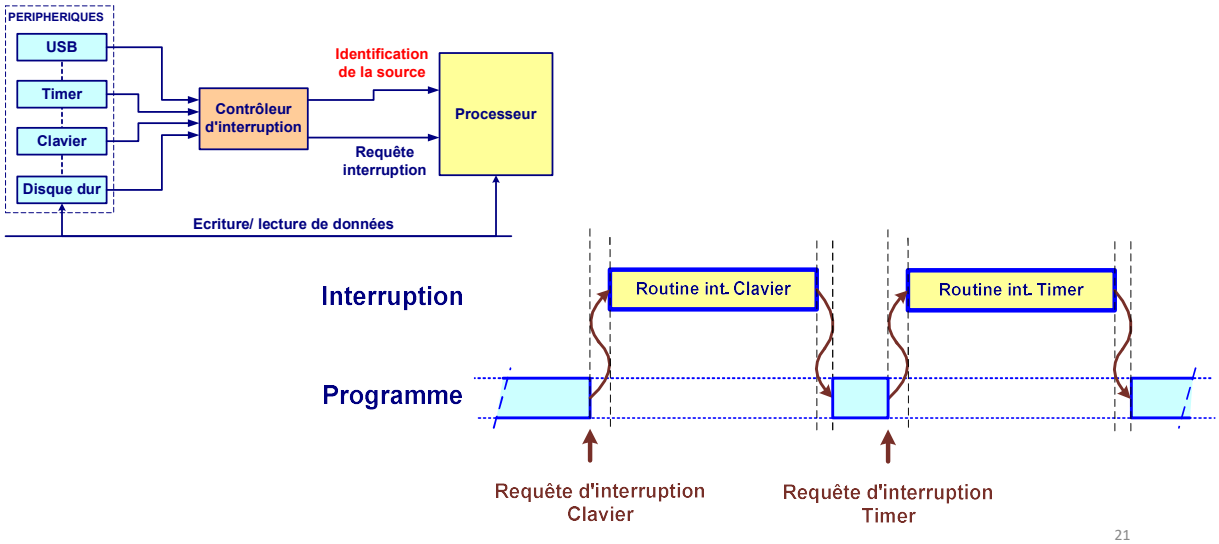
19

Interruptions multi-sources (SANS ID)



20

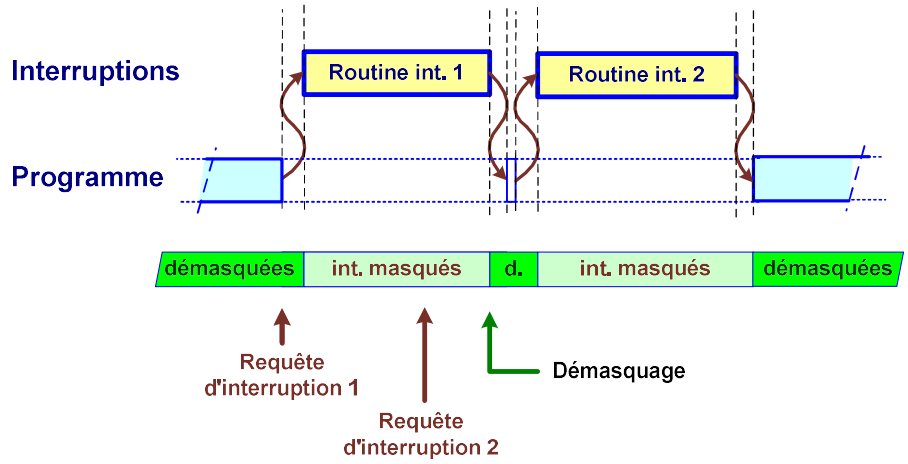
Interruptions multi-sources (avec ID)



21

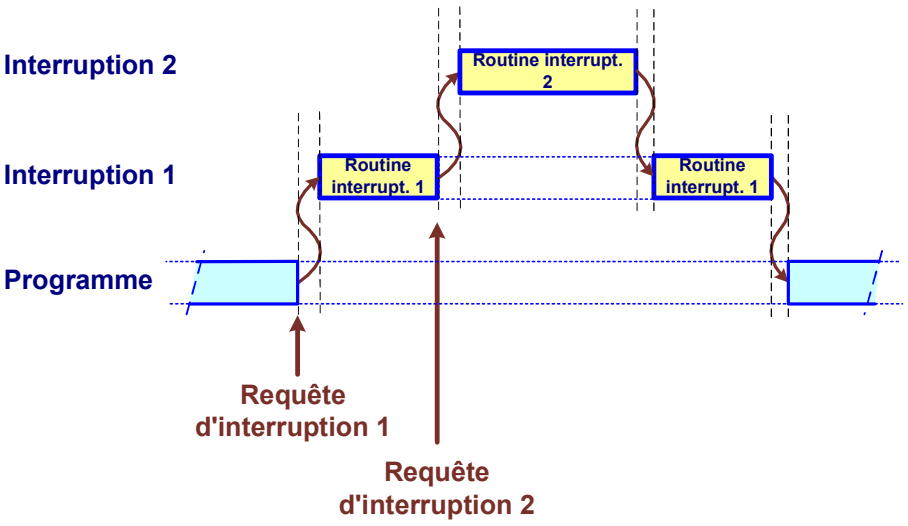
Interruptions chaînées

Les adresses des interruptions sont stockées dans **une pile de stockage (stack)**.



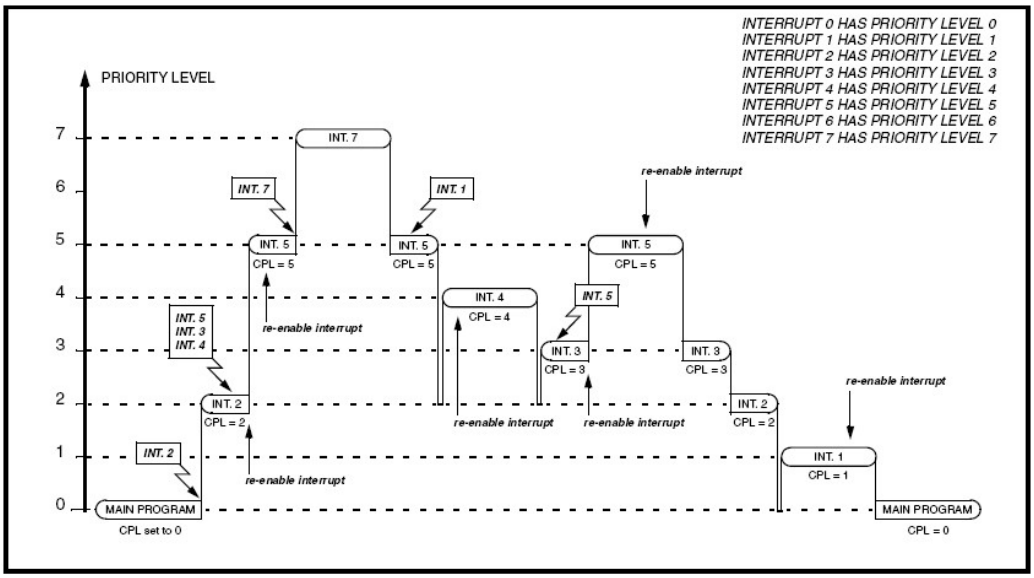
22

Interruptions imbriquées



23

Exemple : interruptions imbriquées



24

Exercice

Un processeur supportant les interruptions imbriquées reçoit les interruptions suivantes depuis l'instant t = 0

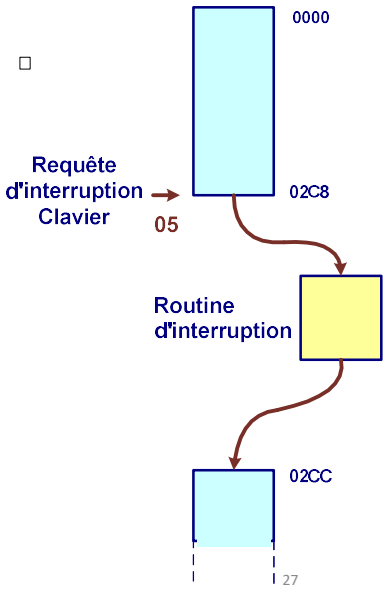
- Priorité 2 t = 1ms durée de la routine 3ms
- Priorité 4 t = 3ms durée de la routine 4ms
- Priorité 3 t = 4ms durée de la routine 2ms
- Priorité 7 t = 5ms durée de la routine 3ms
- Priorité 6 t = 6ms durée de la routine 1ms

Dessinez le chronogramme des interruptions successives.

25

Vecteur d'interruption

- **Moyen technique** pour attacher une routine d'interruption (ISR) à une requête d'interruption (IRQ)
- On stocke l'adresse mémoire de l'ISR qui devra être exécutée si l'IRQ est acceptée



27

Vecteur d'interruption

- La **table des vecteurs d'interruptions** (zone dans la mémoire) contient toutes les adresses vers les ISR associées à chaque source d'interruption.
- Le clavier est la source d'interruption No 5.

Horloge Système	E100h
Port série	E200h
USB	F100h
Timer	E5D0h
Afficheur	E750h
Clavier	EAD0h
Joystick	EB00h
Disque dur	F000h

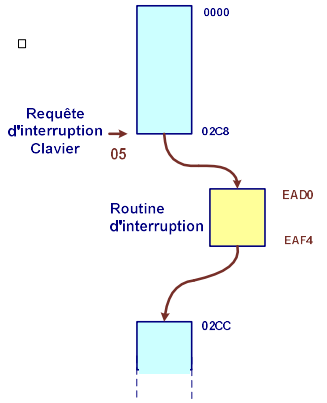
FC40 = Adresse Base = **adresse du vecteur** de la source 0

28

Vecteur d'interruption

- La **table des vecteurs d'interruptions** (zone dans la mémoire) contient toutes les adresses vers les ISR associées à chaque source d'interruption.
- Le clavier est la source d'interruption No 5.

Horloge Système	E100h	FC40 = Adresse_Base
Port série	E200h	FC44
USB	F100h	FC48
Timer	E5D0h	FC4C
Afficheur	E750h	FC50
Clavier	EAD0h	FC54 =Adresse_Base + (05 x 4)
Joystick	EB00h	FC58
Disque dur	F000h	FC5C



Adresse vecteur = Adresse base + 2^{Nbytes} * No de l'interruption

29

Exemple de vecteurs d'interruption ARM

Exception (Interruption)	Mode	Reg.	Priorité	Adresse du vecteur
Reset	Supervisor	_svc	1	0x00000000
Undefined instruction	Undefined	_und	6	0x00000004
Software interrupt	Supervisor	_swc	6	0x00000008
Prefetch Abort	Abort	_abt	5	0x0000000C
Data Abort	Abort	_abt	2	0x00000010
(not assigned)				0x00000014
Interrupt	IRQ	_irq	4	0x00000018
Fast interrupt	FIQ	_fiq	3	0x0000001C

30

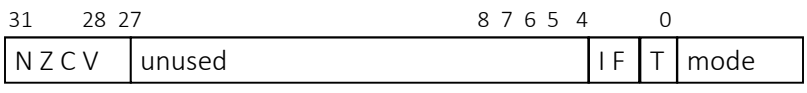
Modes de fonctionnement

- Le processeur ARM peut fonctionner dans 7 modes différents:
  - Le **mode utilisateur**
    - C'est le mode dans lequel s'exécute la plupart des applications.
  - + 6 modes système ou *privilégiés*
    - **Interrupt** (standard) _irq
    - **Fast interrupt** _fiq
    - **Abort** _abt
    - **Indéfini** (*undefined*) _und
    - **Supervisor** (software interrupt) _svc
      - Réservé pour les appels système
    - **Système**

31

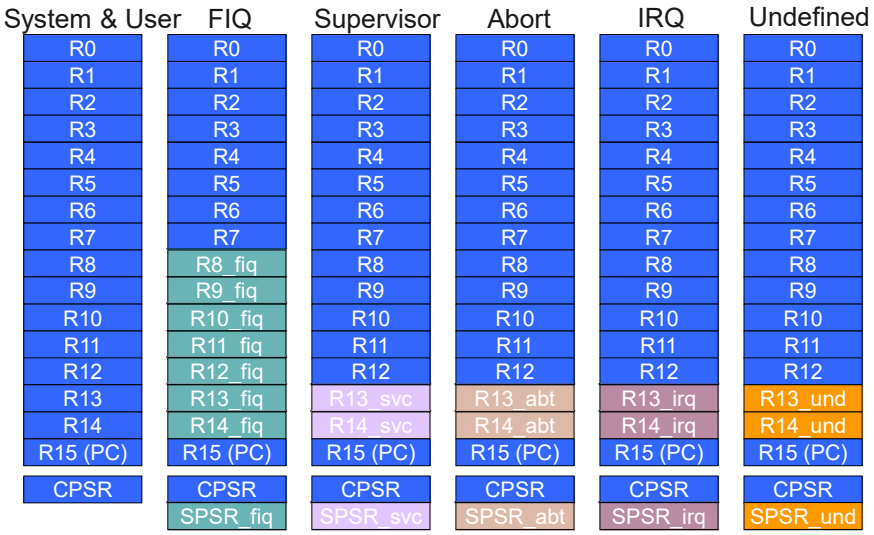
Current Program Status Register (CPSR)

- Permet de **changer le mode** du processeur
  - **Bits de condition (N Z C V)**
  - **Bits de masquage des interruptions (I F)**
  - **Jeu d'instructions (T)**
  - **Mode du processeur (mode)**



32

Registres du CPU



33

Micro-programme d'interruption (ARM)

Interruption

- Si I/F = 1, la requête d'interruption est ignorée.

1. Termine l'instruction courante
2. Sauvegarde de **l'adresse de retour dans le Link Register (R14)**
   1. R14_irq (ou R14_fiq) = R15 (PC) => Note : R15 contient l'adresse suivante + 4
3. Sauvegarde du **registre de statut**
   1. SPSR_irq/SPSR_fiq = CPSR
4. Activation du **mode** d'exécution
   1. CPSR[4:0] = 0x0b10010 (ou 0x0b10001 pour FIQ)
5. Désactivation de **l'entrée d'interruption**:
   1. Pour IRQ: activation du masque IRQ (I=1) => CPSR[7] = 1 (FIQ peut encore interrompre)
   2. Pour FIQ: activation du masque IRQ+FIQ (I=1, F=1) => CPSR[6,7] = 1
6. Chargement du **vecteur d'interruption** à l'adresse respective:
   1. Pour IRQ: R15 (PC) = 0x00000018
   2. Pour FIQ: PC = 0x0000001C
7. **Routine** .....

34

Micro-programme d'interruption (ARM)

Retour d'interruption

1. Chargement du PC par **adresse de retour** (instruction spéciale)
    1. R15 = R14_irq(fiq) – 4
2. Récupération du statut, donc du **mode d'exécution**
    1. CPSR = SPSR_irq(fiq)

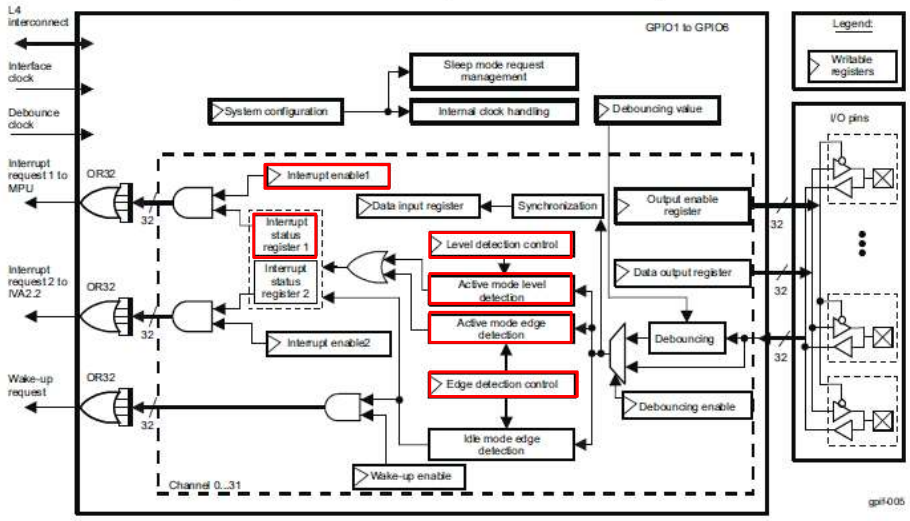
35

Traitement des interruptions (DM3730)

- **Contrôleur d'interruption** (ch 12)
    - partie INTCPS (Int. Controller Processor Subsystem)
- Supporte **96 interruptions**
- Traite les **interruptions normales** (IRQ) et **rapides** (FIQ) avec un mécanisame de priorité
- **6 interruptions pour les GPIOs** (une par banque de 32 GPIOs)

36

Interruptions des GPIOs



Labo Etape 3: Initialisation des interruptions

1. Placer dans la table des vecteurs **l'adresse de la routine d'interruption**
2. Choisir le **type d'interruption** (normale ou fast) et le niveau de priorité: _ILRn
3. **Démasquer** la ligne d'interruption (par exemple, la banque de GPIO choisie): _MIRm
4. **Accepter** une interruption IRQ ou FIQ: _CONTROL
5. Activation des interruptions par le **CPSR**

```
MRS r1, cpsr
BIC  r1, r1, #0x80   @IRQ clear
MSR cpsr_c, r1
```

Labo Etape 3 : Routine d'interruption

1. Sauver les registres R0 à R12, et LR dans la pile
2. Relever le n° de l'interruption servie en lisant le registre INTCPS_SIR_IRQ
3. Sauver les registres R0 à R12, et LR dans la pile
4. Appeler une routine (C ou assembleur) , le handler d'interruption et effectuer le traitement souhaité
5. Acquitter l'interruption à la source (stopper la requête) dans le handler ou dans la routine
6. Autoriser une interruption IRQ ou FIQ: _CONTROL
7. Restaurer les registres depuis la pile
8. Terminer la routine avec une instruction spéciale restaurant le CPSR (par exemple SUBS PC, LR, #4)

39