Emulation STM32L Discovery avec Qemu

Clavelin Aurélien, Eid Timothée, Mercier Michaël

Plan

Plan

Clavelin Aurélien, Eid Timothée, Mercier Michaël

Présentation

- Découverte du STM32L
- Très basse consommation
- Carte embarquée
- Beaucoups de périphériques
- Interface de debugage

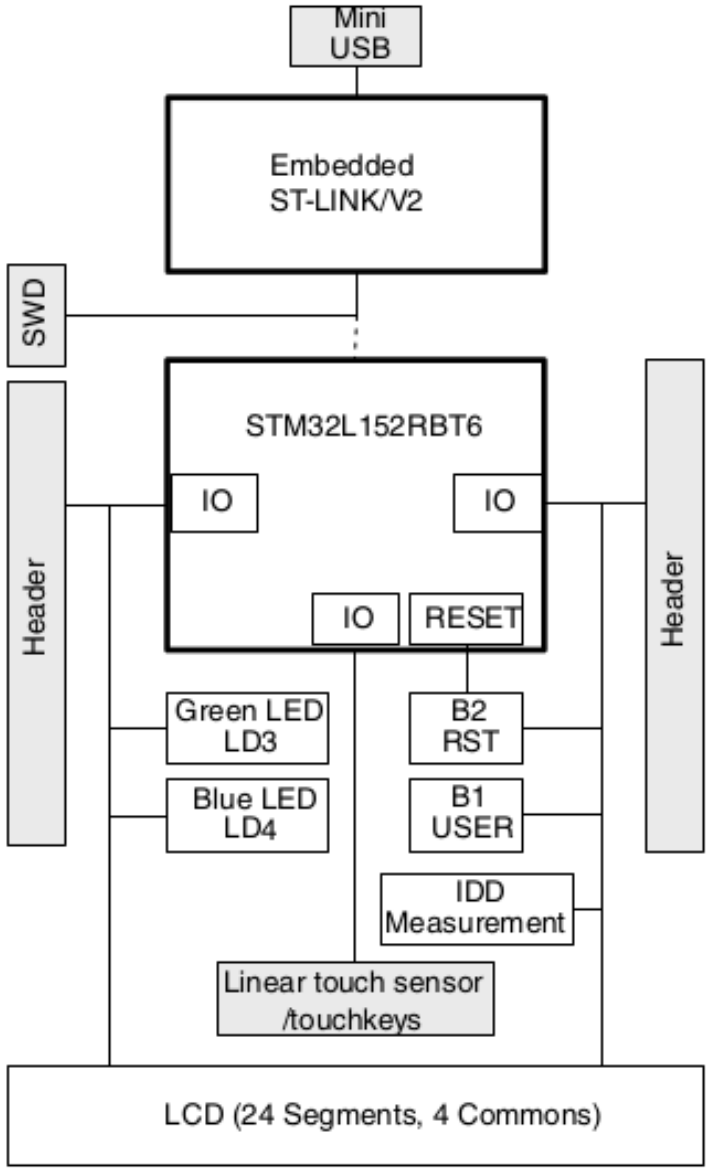
Caractéristiques

Microcontroleur

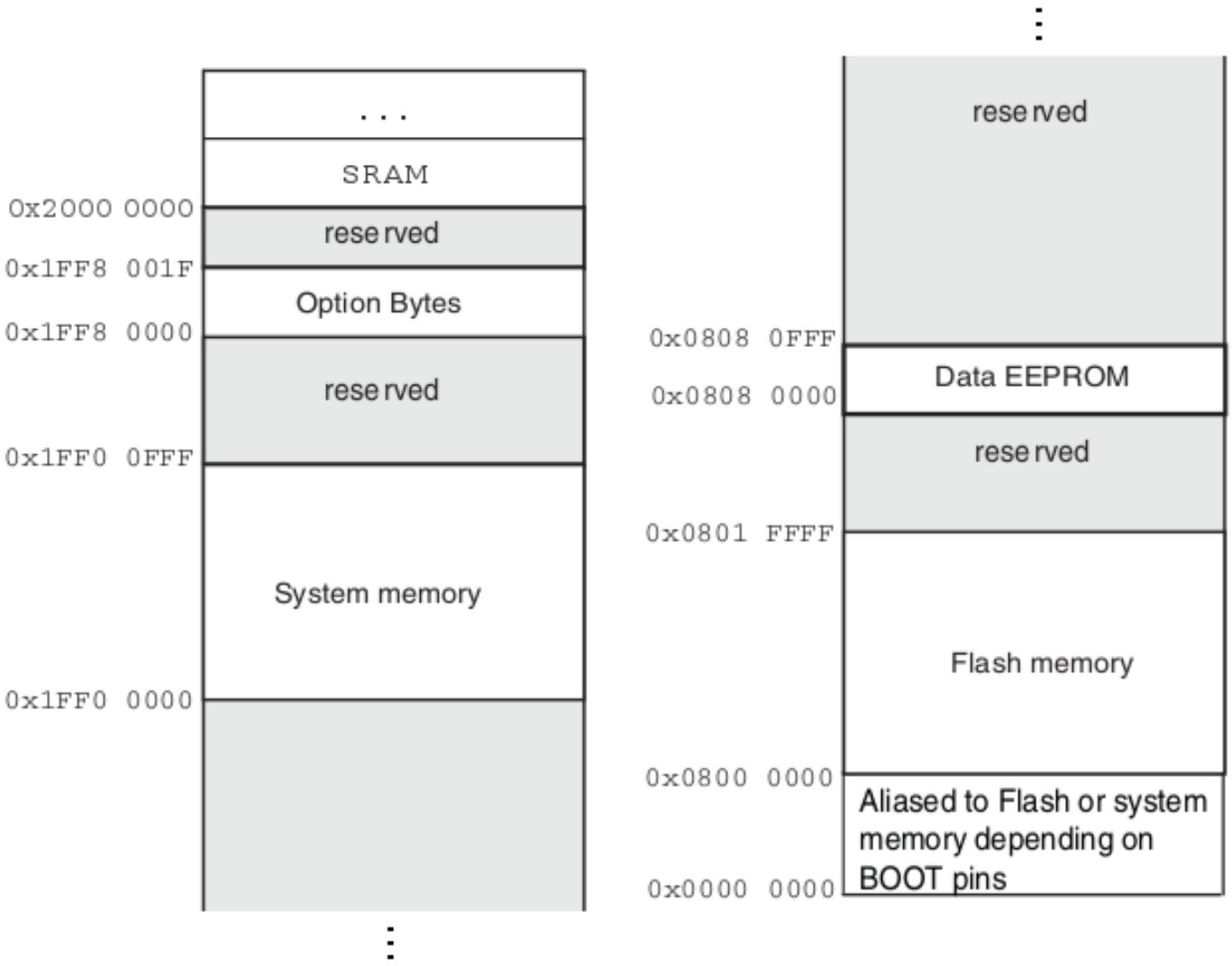
- Adresse 32 bits
- Mémoire
  - Flash : 128 Ko
  - RAM : 16 Ko
- Others
  - RTC (Real Time Clock)
  - USART (Universal Asynchrone/Synchrone Receiver/Transmitter)
  - I2C (Inter Integrated Circuit)
  - SPI (Serial peripheral interface)
  - ADC (Analog-Digital Convertor)
  - DAC (Digital-Analog Convertor)
  - Comparateurs
  - ...

Clavelin Aurélien, Eid Timothée, Mercier Michaël

Caractéristiques



Structure de la mémoire

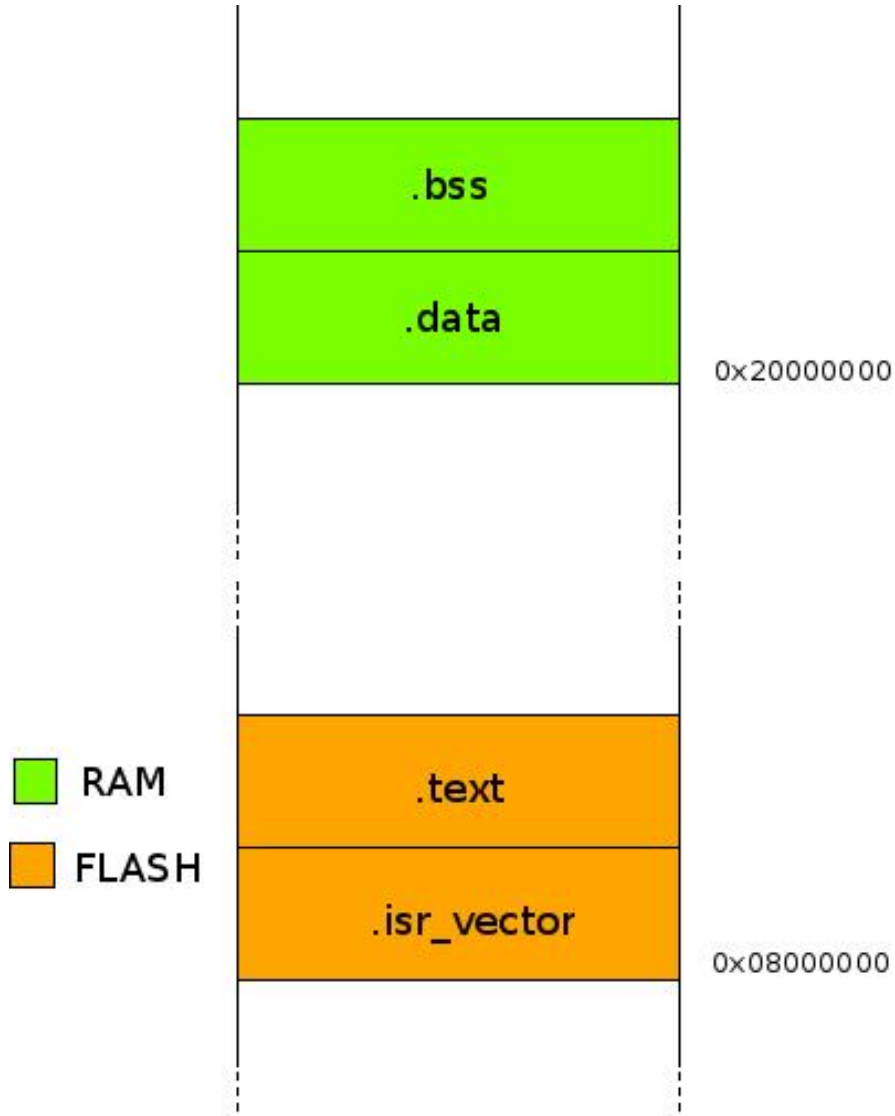


Structure d'un programme

Pour créer un executable ELF

- Code source
- Startup program
- Linker

Structure d'un programme



Chargement sur la carte

Prérequis

- Suite de cross compilation (arm-none-eabi)
- Utilitaire texane-stlink

Procédure de chargement

- Lancement de l'utilitaire texane-stlink
  - # ./st-util -p 4242
- Chargement du programme avec GDB
  - # arm-none-eabi-gdb
  - (gdb) target extended-remote :4242
  - (gdb) load ./programme.elf

Plan

Qemu

- Emulation complète
  - x86/x64
  - ARM
  - PowerPC
  - Sparc
  - ...
- VMWare, Virtualbox, ...

Emulation d'un programme

Procédure de chargement

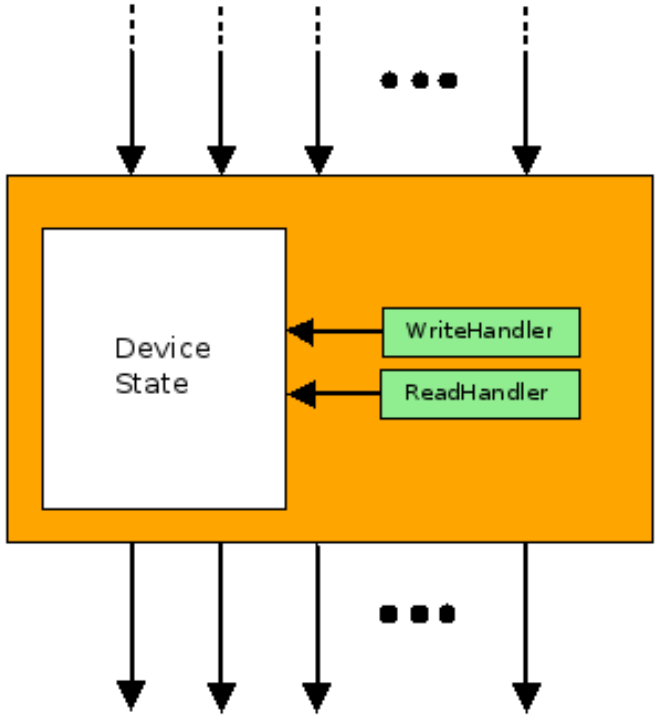
- Compilation du programme utilisateur
- Lancement de Qemu avec le programme émulé

q

emu_systeme_arm

- -M stm32l152rbt6
- -chardev xxx
- -kernel /path/to/program

Composant



- Interfaces d'entrées (fils)
- Interfaces de sorties (fils)
- Etat du composant (Device State)
- Gestion d'une zone mémoire déléguée
- Enregistré dans Qemu : Accessible pour les machines

Machine

- Qemu met à disposition des machines

  Syborg Platforme virtuelle Symbian

  n800 Nokia N800

  lm3s6965evb Stellaris LM3S6965EVB

  stm32l152rbt6 STM32L Discovery (Travail réalisé)

- Assemblage de composants
- Communication avec des CharDev

CharDev

- Interface de communication
  - Socket
  - Clavier
  - Port série
  - ...
- Bidirectionnelle
- Fonctionnement
  - Fonction d'envoi
  - Handler de réception
  - Initialisation au lancement de Qemu
  - Connexion à l'initialisation de la machine

Plan

Clavelin Aurélien, Eid Timothée, Mercier Michaël

GPIO - Description

- General Purpose Input/Output
- Entrée/Sortie pour usage général
- Paramètrable grâce à des registres de configuration
- Modes de fonctionnement
  - Implémentés
    - Output
    - Input
  - Non implémentés
    - Alternate function

Clavelin Aurélien, Eid Timothée, Mercier Michaël

GPIO - Implémentation

- Registres représentés par une structure

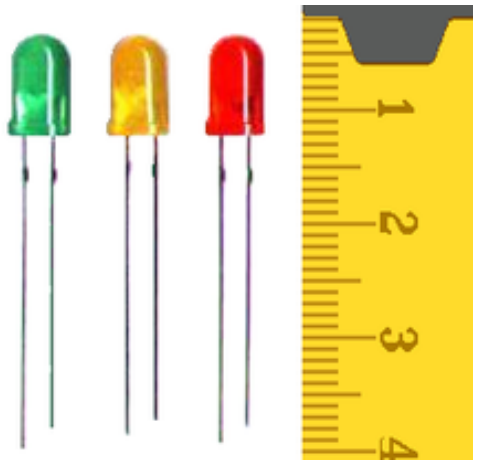
```
 4
 5 typedef struct {
 6     SysBusDevice busdev;
 7
 8     /* Registres GPIO (Reference Manual p119 */
 9     uint32_t mode; /* Mode */
10     uint16_t otype; /* Output type */
```

- R/W en mémoire grâce aux fonctions handlers

```
74 static uint32_t stm32_gpio_read(void *opaque, target_phys_addr_t offset) {
75     stm32_gpio_state *s = (stm32_gpio_state *) opaque;
76
77     switch (offset) {
78         case 0x00: /* Mode */
79             return s->mode;
80         case 0x04: /* oType */
81             return s->otype;
82         case 0x08: /* oSpeed */
```

Clavelin Aurélien, Eid Timothée, Mercier Michaël

LED



- Connexion à une sortie GPIO
- Envoi de l'état au CharDev quand la sortie GPIO change d'état

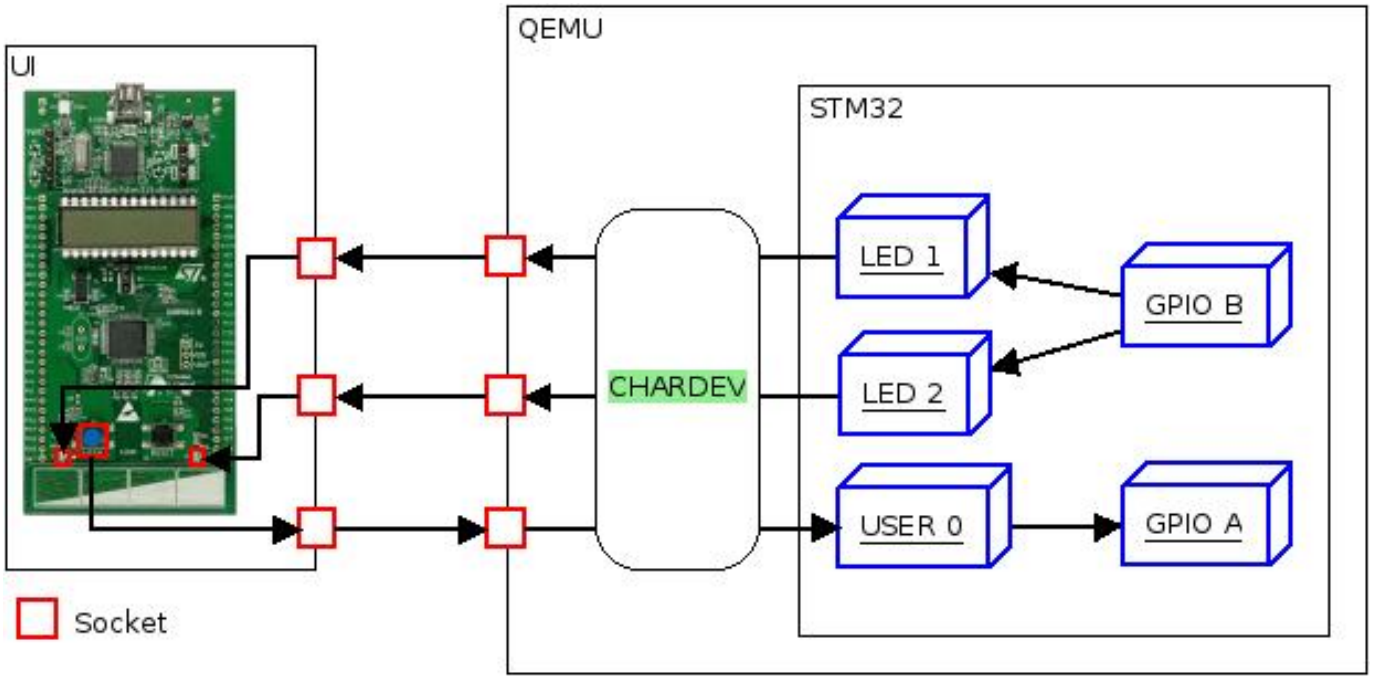
Clavelin Aurélien, Eid Timothée, Mercier Michaël

Bouton

- Connexion à une entrée GPIO
- Envoi d'un signal au GPIO quand il reçoit un signal du chardev

Clavelin Aurélien, Eid Timothée, Mercier Michaël

Machine



RCC

Reset and Clock Control (Contrôle les horloges)

Procédure d'écriture sur une pin GPIO

- Ecriture dans le registre "Output data"
- La valeur sera recopiée sur la pin au prochain tick d'horloge

Problème

Les composants RCC et timers n'ont pas été implémentés dans Qemu

Solution adoptée

Notre implémentation des GPIO recopie la valeur sur la pin sans attendre le tick de l'horloge

Zones d'adresse

Mapping d'adresses

- Zone de boot pouvant être remappée en fonction des cavaliers sur
  - SRAM : Pour les tests
  - Flash : Pour ecrire un programme persistant

Problème

Ce remappage n'existe pas sur Qemu

Solution adoptée

Création de 2 version de programme

- Qemu : le programme est "nativement" logé aux adresses de boot (0x0000 0000)
- Hardware : le programme est logé en flash (0x0800 0000) puis mappé materiellement sur les adresses de boot (0x0000 0000)

Plan

GPIO Output

Déroulement

- Initialisation des GPIO
  - Le registre Mode
  - Activation de la RCC pour le GPIO_B (LED)
- While(1)
  - Switch_LED_ON()
    - Ecrire dans le registre de sortie
  - Delay()
  - Switch_LED_OFF()
    - Ecrire dans le registre de sortie
  - Delay()

GPIO Input + Attente active

Limite

Nous n'avons pas implémenté les interruptions processeur pour pouvoir utiliser des handlers d'interruption. Ce programme réalise donc une attente active

Déroulement

- Initialisation
  - GPIO_B en sortie, GPIO_A (Bouton) en entrée
  - Activation de la RCC pour les GPIO
- While(1)
  - Si l'état a changé
    - Switch_LED()

Plan

Prochaines évolutions

- Gestion des interruptions materielles
- Implémentation des protocoles (Alternate function)
- Implémentation des composants (Ecran, ADC, ...)
- Implémentation du composant RCC et des Timers

Rétrospective

Points négatifs

- Temps pour appréhender le système Qemu
- Temps pour appréhender le hardware

Points positifs

- Utilisation des CharDev
- Compréhension des mécanisme de Qemu
- Documentation du Wiki Qemu

Questions