GPIO

Table des matières

1 Introduction

La carte de développement met à notre dispositon un ensemble de bits dits GPIO rendu accessibles au câblage via deux pseudo bus parallèles. Ces bits sont analogues à ceux du port parallèle. J'ai étudié les GPIO dans le contexte d'une donnée inscrite en sortie (pas en entrée). Ces bits sont alimentés à 3,3 volts lorsqu'ils sont positionnés à 1. Nous verrons dans cette partie, comment commander ces bits par bloc de 16.

2 Localisation des GpIo

Les pseudo ports parallèle **J10** et **J7** nous donnent accès à :

- La masse, 5v et 3,3V
- Certaines broche du module AT91RM9200 hors GPIO
- Des GPIO utilisées par le module
- Des GPIO déconnectables du module via des jumpers
- Des GPIO libres

2.1 Affectation *théorique* des GpIo

La localisation des GPIO est donnée par ces 2 schémas :

- le schéma de la carte
- le schéma de plus haut niveau

GPIO directement à des composants :

- **I2C**
  - a25, a26
- **USB**
  - c14, c15
- **SDCARD/MMC**
  - a27, a28, a29

  - b3, b4, b5
  - c2
- nWAIT
  - c6

GPIO à des composants via jumpers :

- **Led**
  - c0
  - Jmp23
- **UART 0**
  - a17, a18, a20, a21
  - Jmp14, à Jmp17
- **UART 3**
  - a5, a6
  - b0, b1
  - JM18 à Jmp21
- **UART Debug & 2**
  - a22, a23, a30, a31
  - Jmp9, Jmp10, Jmp12 et Jmp13
- **UART 1**
  - b18, b19, b20, b21, b23, b24, b25, b26
  - Jmp1 à Jmp8
- **Reset** (difficilement exploitable puisque réservé au flashage)
  - a31
  - Jmp11

2.2 En pratique

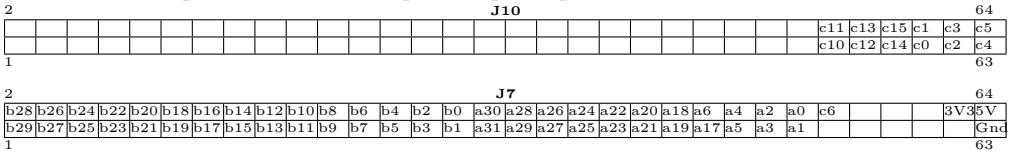
- Famille C (c[0-6] et c[10-15]) répartie sur les 12 bits de poids forts des 2 bus.
  - c0 commande la led (non utilisable)
  - c6 utilisé par *nWait* (utilisation déconseillé) ?
  - c2 utilisé par SDCARD/SSC ?
  - c14, c15 utilisés par USB ?
  - c1, c3, c4, c5, c[10-13] : 8 non affectés ?
- Famille B (b[0-29]) répartie sur les 30 bits de poids faibles du bus J7.
  - b0, b1 utilisés par UART 3 (utilisables)
  - b[3-5], utilisés par SDCARD/SSC (utilisation non réussie : dé-sélectionner **driver/MMC** lors de la compile du noyau)

  - b[12-19], utilisés par *MAC+PHY Ethernet* (cf *build/linux-2.6.22.1/arch/arm/mach-at91/at91rm9200_devices.c*)
  - b[18-21] et b[23-26] utilisé par UART 1 (utilisable sauf b21 : 2v pour 1 logique)
  - b2, b[6-11], b22 et b[27-29] : 8 bits non affectés ?

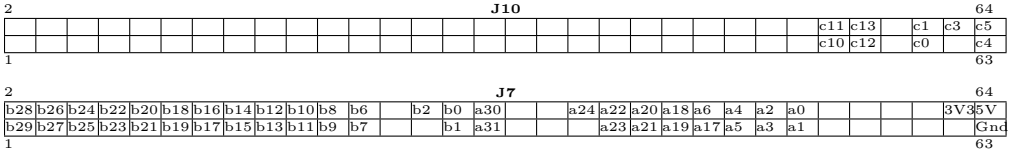
- Famille A (a[0-6] et a[17-31]) répartie sur les 22 derniers bits au milieu du bus J7.
  - a5 et a6 utilisés par UART 3 (utilisables)
  - a17, a18, a20 et a21 utilisés par UART 0 (utilisables)
  - a22, a23 utilisés par UART 2 (utilisables)
  - a25, a26 utilisés par I2C (utilisables : déselection du driver à la compil du noyau ?)
  - a[27-29] utilisés par SDCARD/SSC (utilisables ? !)
  - a30 et a31 utilisés par UART DEBUG (non utilisables : font planter la carte)
  - a[0-4], a19, a24 : 7 bits non affectés (utilisables directement)

FIGURE 1 : Disposition des GPIO sur les deux pseudo ports parallèles

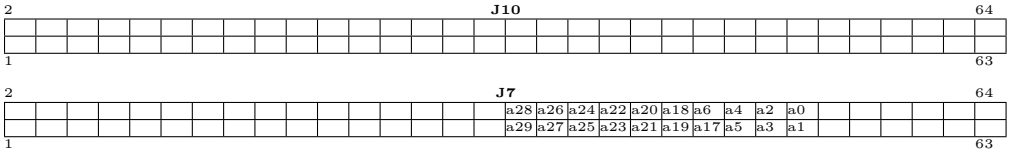
Voici les GPIO présentes sur les 2 pseudo ports parallèle **J10** et **J7**.



Les GPIO testées libres avec la distribution de base (à réaliser) :



Les GPIO testées comme rendues libres (à compléter) :



3 Test via ssh

On peut éteindre puis rallumer la led du PC Client par exemple :

```
$ ssh-keygen -t dsa
$ cat ~/.ssh/id_dsa.pub | ssh root@192.168.1.2 -p 2222 'cat > ~/.ssh/authorized_keys'

$ ssh root@192.168.1.2 -p 2222 gpio -PC0
$ ssh root@192.168.1.2 -p 2222 gpio +PC0
```

Ce petit programme d'exemple permet de positionner les d'un coup toutes les GPIO modifiables. Attention toucher à **a30** ou à **a31** freeze la carte. Les bits doivent être testé au voltmètre ou à l'osciloscope.

```
#!/bin/sh

echo "Ce script bascule l'ensemble des GPIO A, B et C."
Usage()
{
    echo -e "Usage: \n\t$1 {+|-|?}"
    exit 1
}

[ $# -eq 1 ] || Usage $0
[ $1 == "+" ] || [ $1 == "-" ] || [ $1 == "?" ] || Usage $0

listA=" a0  a1  a2  a3  a4  a5  a6              \
                                a17 a18 a19 \
        a20 a21 a22 a23 a24 a25 a26 a27 a28 a29 "
listB=" b0  b1  b2  b3  b4  b5  b6  b7  b8  b9  \
        b10 b11 b12 b13 b14 b15 b16 b17 b18 b19 \
        b20 b21 b22 b23 b24 b25 b26 b27 b28 b29 "
listC=" c0  c1  c2  c3  c4  c5  c6              \
        c10 c11 c12 c13 c14 c15                 "


# $1 : bus name
# $2 : list
LoopOnList()
{
        echo -ne "$1:\t"
        for io in $2
        do
                [ $1 != "?" ] && gpio ${1}P${io}
                gpio ?P${io}
                echo -n "$? "
        done
}

[ $1 == "?" ] || echo "mise a $1 ..."
LoopOnList "A"  "$listA"
LoopOnList "B"  "$listC"
LoopOnList "C"  "$listC"
echo
```

4 Programme C

On peut donc utiliser les GPIO **a[0-6]** et **a[17-25]** afin d'avoir 16 bits consécutifs. Les procédures suivantes permettent d'utiliser 16 bits avec :

- pour bit de poids fort (utilisé comme bit *strobe*) la GPIO **a25**.
- pour bit de poids faible la GPIO **a0**.

```
#include <memory.h>
#include <sys/mman.h>
#include <sys/ioctl.h>
#include <fcntl.h>
...

int mapPage(int *fd, void **map_base)
{
  int rc = 1;

  if((*fd = open("/dev/mem", O_RDWR | O_SYNC)) == -1){
    exit(-1);
  }

  /* Map one page */
  *map_base = mmap(0, MAP_SIZE, PROT_READ | PROT_WRITE, MAP_SHARED, *fd, AT91_SYS & ~MAP_MASK);
  if(*map_base == (void *) -1){
    close(fd);
    exit(-1);
  }

  return rc;
}

int unmapPage(int *fd, void **map_base)
{
  int rc = 1;

  /* Unmap previously mapped page */
  if(munmap(*map_base, MAP_SIZE) == -1){
    close(fd);
    exit(-1);
  }

  close(*fd);
  *fd = 0;
  *map_base = (void *) 0;
  return rc;
}

/* return GPIO from a0-a6 and a17-a25 */
unsigned int getLedGpio()
{
  int fd, pin,i;
  void *map_base;
  unsigned long readval, writeval, lowVal, hightVal;
  unsigned long lowMask = 0x0000007F;
```

```
  unsigned long hightMask = 0x03FE0000;
  unsigned int  combinedVal;

  int dest_addr = PIOA_OFFSET;
  int id = PIOA_ID;

  mapPage(&fd, &map_base);

  /* First, enable PIO clock */
  writeval = 1 << id;
  *((unsigned long*)(map_base + ((PMC_OFFSET + PMC_PCER) & MAP_MASK))) = writeval;
  /* Then read port value */
  readval = *((unsigned long*)(map_base + ((dest_addr + PIO_PDSR) & MAP_MASK)));

  lowVal = readval & lowMask;
  hightVal = (readval & hightMask) >> 10;
  combinedVal = hightVal | lowVal;

  unmapPage(&fd, &map_base);
  return combinedVal;
}

/* set GPIO from a0-a6 and a17-a25 */
int setLedGpio(unsigned int val)
{
  int rc = 1, fd, pin;
  void *map_base;
  unsigned long writeval, lowVal, hightVal, combinedVal;
  unsigned int mask = 0xFFFF;
  unsigned int lowMask = 0x007F;
  unsigned int hightMask = 0xFF80;

  int dest_addr = PIOA_OFFSET;
  int id = PIOA_ID;

  if ((val | mask) != mask) rc = -1;
  lowVal = val & lowMask;
  hightVal = ((val & hightMask) >> 7) << 17;
  combinedVal = lowVal | hightVal;

  mapPage(&fd, &map_base);

  /* set GPIO from a0 to a6 */
  for (pin=0; pin<=6; ++pin)
    {
      /* Setup PIO registers */
      writeval = 1<< pin;
      *((unsigned long*)(map_base + ((dest_addr + PIO_PER) & MAP_MASK))) = writeval;
      *((unsigned long*)(map_base + ((dest_addr + PIO_OER) & MAP_MASK))) = writeval;

      if ((combinedVal >> pin) & 0x00000001 == 1)
{
  /* set */
  *((unsigned long*)(map_base + ((dest_addr + PIO_SODR) & MAP_MASK))) = writeval;
}
      else
{
  /* unset */
  *((unsigned long*)(map_base + ((dest_addr + PIO_CODR) & MAP_MASK))) = writeval;
}
    }
 /* set GPIO from a17 to a25 */
  for (pin=17; pin<=25; ++pin)
    {
      /* Setup PIO registers */
      writeval = 1<< pin;
      *((unsigned long*)(map_base + ((dest_addr + PIO_PER) & MAP_MASK))) = writeval;
      *((unsigned long*)(map_base + ((dest_addr + PIO_OER) & MAP_MASK))) = writeval;

      if ((combinedVal >> pin) & 0x00000001 == 1)
{
  /* set */
  *((unsigned long*)(map_base + ((dest_addr + PIO_SODR) & MAP_MASK))) = writeval;
}
      else
{
  /* unset */
  *((unsigned long*)(map_base + ((dest_addr + PIO_CODR) & MAP_MASK))) = writeval;
}
    }

  unmapPage(&fd, &map_base);
  return rc;
}
...
int strobeIt(unsigned int paio)
{
  int rc = 1;
  rc &= setLedGpio(paio & 0x7FFF);
  logEmit(DefaultLog, "LED GPIO: 0x%4x\n", getLedGpio());

  rc &= setLedGpio(paio | 0x8000);
  logEmit(DefaultLog, "LED GPIO: 0x%4x\n", getLedGpio());

  rc &= setLedGpio(paio & 0x7FFF);
  logEmit(DefaultLog, "LED GPIO: 0x%4x\n", getLedGpio());
  return rc;
}
es
```