MI11
TP Linux Temps Réel

Jeanneau Louis, Schulster Alex

Printemps 2022

Table des matières

Préambule

Afin d'avoir toutes les fonctionnalités du noyau nécessaires à la réalisation de ce TP, nous copions l'image mise à disposition sur le *moodle* dans le directoire `/tftpboot`.

Ainsi, le `JoyPiNote` bootera automatiquement sur cette image avec les fonctionnalités supplémentaires nécessaires.

1 Communication Xenomai / Linux

Nous devons effectuer une communication entre une tâche temps réel et une tâche non-temps réel. Pour cela, on utilise les *pipes* fournies par l'API `Alchemy`. On lit la phrase suivante sur la documentation :
Xenomai threads open their side of the pipe using the ***`rt_pipe_create()`*** *service ; regular Linux threads do the same by opening one of the* ***`/dev/rtpN`*** *special devices, where* ***`N`*** *is the minor number agreed upon between both ends of each pipe.*

Nous choisissons de manière arbitraire un *minor* de 1 pour le pipe. Ainsi, notre programme main devra ouvrir le fichier `/dev/trp1` et le lire pour pouvoir accéder aux messages envoyés par notre tâche temps réel.

Lors de l'envoi des messages, nous devons choisir un mode entre `P_URGENT` et `P_NORMAL` (message en tête ou bout de queue). Nous décidons de le mettre en mode normal pour adopter un comportement FIFO.

1.1 Premier programme

Voici le code *C* de notre programme de communication unidirectionnel :

```c
// Including necessary libraries
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <alchemy/task.h>
#include <alchemy/pipe.h>

// Defining constants
#define TASK_PRIO 99
#define TASK_MODE T_JOINABLE
#define TASK_STKSZ 0
#define MSG_SIZE 50

// Global pipe descriptor
RT_PIPE pipe_desc;

// Real-time task that continuously sends 'Hello, World' message
// to the pipe
void say_hello_task() {
    // Counter
    int i = 0;
    // Message buffer
    char msg[MSG_SIZE];
    for (;;)
    {
        // Generate message with appropriate counter
        sprintf(&msg, "Hello, World %.03d!\n", i);
        // Send message to pipe
        rt_pipe_write(&pipe_desc, &msg, sizeof("Hello, World! 000\n"), P_NORMAL);
        // Sleep for 1 second
        rt_task_sleep(1000000000);
        // Increment counter
        i++;
    }
}


int main() {
    // Return code for task creation
    int err;
    // Task descriptor
    RT_TASK task_desc;
    // Pipe minor (arbitrary choice)
    int minor = 1;
    // Message buffer
    char msg[MSG_SIZE];
```

```
47
48    // Create task and check that it went well
49    err = rt_task_create(&task_desc, "hello",
          TASK_STKSZ, TASK_PRIO, TASK_MODE);
50    if (err != 0) {
51       printf("error rt_task_create\n");
52       return 1;
53    }
54
55    // Create pipe and check that it went well
56    err = rt_pipe_create(&pipe_desc, "tuyau",
          minor, 0) ;
57    if (err != minor) {
58       printf("error rt_pipe_create\n");
59       rt_task_delete(&task_desc);
60       return 1;
61    }
62
63    // Start task
64    rt_task_start(&task_desc, &say_hello_task,
          NULL);
65
66    // Open pipe file and check that it went
          well
67    int fd = open("/dev/rtp1", O_RDONLY);
68    if (fd ==-1)
69    {
70       rt_task_delete(&task_desc);
71       rt_pipe_delete(&pipe_desc);
72    }
73    // Infinite for loop
74    for (;;)
75    {
76       // Read the message
77       read(fd, &msg, MSG_SIZE);
78       // Print it
79       printf(msg);
80    }
81
82    // Wait for task to finish and cleanly stop
          the program
83    rt_task_join(&task_desc);
84    rt_task_delete(&task_desc);
85    rt_pipe_delete(&pipe_desc);
86    close(fd);
87    return 0;
88 }
```

Et le résultat obtenu dans le terminal sur la cible :

```
root@joypinote-xenomai:~# cd /usr/
root@joypinote-xenomai:/usr# ./hello
Hello, World 000!
Hello, World 001!
Hello, World 002!
Hello, World 003!
Hello, World 004!
Hello, World 005!
Hello, World 006!
Hello, World 007!
Hello, World 008!
```

1.2 Communication bidirectionnelle

Pour la communication bidirectionnelle, la première tâche temps réel envoi le message `Hello, World` sur le pipe, sans compteur. Le `main` lit ce message, l'imprime puis écrit la valeur du compteur sur le pipe. Enfin, la deuxième tâche temps réel lit le pipe et imprime la valeur du compteur reçue. Voici le code :

```
1  // Including necessary libraries
2  #include <stdio.h>
3  #include <unistd.h>
4  #include <fcntl.h>
5  #include <alchemy/task.h>
6  #include <alchemy/pipe.h>
7
8  // Defining constants
9  #define TASK_PRIO 99
10 #define TASK_MODE T_JOINABLE
11 #define TASK_STKSZ 0
12 #define MSG_SIZE 50
13
14 // Global pipe descriptor
15 RT_PIPE pipe_desc;
16
17 // Real-time task that continuously sends
       'Hello, World' message
18 // to the pipe
19 void say_hello_task() {
```

```
20    for (;;)
21    {
22       // Send message to pipe
23       rt_pipe_write(&pipe_desc, "Hello, World
             !\n", MSG_SIZE, P_NORMAL);
24       // Sleep
25       rt_task_sleep(1000000000);
26    }
27 }
28
29 // Real-time task that reads the pipe and
       prints its content
30 void read_hello_task() {
31    // Message buffer
32    char msg[MSG_SIZE];
33    for (;;)
34    {
35       // Read the pipe
36       rt_pipe_read(&pipe_desc, &msg, MSG_SIZE,
             TM_INFINITE);
37       // Print the received message
38       rt_printf("Reader RT task : ");
39       rt_printf(msg);
40    }
41 }
42
43
44 int main() {
45    int err;
46    RT_TASK task_write_desc;
47    RT_TASK task_read_desc;
48    int minor = 1;
49    char msg[MSG_SIZE];
50
51    // Create task
52    err = rt_task_create(&task_write_desc,
          "hello", TASK_STKSZ, TASK_PRIO,
          TASK_MODE);
53    if (err != 0) {
54       printf("error rt_task_create\n");
55       return 1;
56    }
57
58    // Create second task
59    err = rt_task_create(&task_read_desc,
          "reader", TASK_STKSZ, TASK_PRIO,
          TASK_MODE);
60    if (err != 0) {
61       printf("error rt_task_create reader\n");
62       rt_task_delete(&task_write_desc);
63       return 1;
64    }
65
66    // Create pipe
67    err = rt_pipe_create(&pipe_desc, "tuyau",
          minor, 0) ;
68    if (err != minor) {
69       printf("error rt_pipe_create\n");
70       rt_task_delete(&task_write_desc);
71       rt_task_delete(&task_read_desc);
72       return 1;
73    }
74
75    // Start the tasks
76    rt_task_start(&task_write_desc,
          &say_hello_task, NULL);
77    rt_task_start(&task_read_desc,
          &read_hello_task, NULL);
78
79    // Open pipe file
80    int fd = open("/dev/rtp1", O_RDWR);
81    if (fd ==-1)
82    {
83       rt_task_delete(&task_write_desc);
84       rt_task_delete(&task_read_desc);
85       rt_pipe_delete(&pipe_desc);
86    }
87
88    // Counter
89    int i = 0;
90    // Infinite loop
91    for (;;)
92    {
93       // Read the pipe
94       read(fd, &msg, MSG_SIZE);
95       // Print the message
96       printf("Main : ");
97       printf(msg);
98       // Create and send new message to pipe
99       sprintf(&msg, "%.03d\n", i);
100      write(fd, &msg, MSG_SIZE);
101      // Increment counter
102      i++;
103   }
104
105   // Exit the program
106   rt_task_join(&task_write_desc);
107   rt_task_delete(&task_write_desc);
108   rt_task_delete(&task_read_desc);
109   rt_pipe_delete(&pipe_desc);
110   close(fd);
111   return 0;
112 }
```

Et le résultat obtenu dans le terminal sur la cible :

```
root@joypinote-xenomai:~# cd /usr/
root@joypinote-xenomai:/usr# ./double_hello
Main : Hello, World !
Reader RT task : 000
Main : Hello, World !
```

```
Reader RT task : 001
Main : Hello, World !
Reader RT task : 002
Main : Hello, World !
Reader RT task : 003
Main : Hello, World !
Reader RT task : 004
Main : Hello, World !
Reader RT task : 005
Main : Hello, World !
Reader RT task : 006
Main : Hello, World !
Reader RT task : 007
Main : Hello, World !
Reader RT task : 008
Main : Hello, World !
Reader RT task : 009
Main : Hello, World !
Reader RT task : 010
```

2 GPIOs RTDM

2.1 Préparation

2.1.1 Chargement de module

Logs noyau :

```
root@joypinote-xenomai:/usr# dmesg
[    0.000000] Booting Linux on physical CPU 0x0
[    0.000000] Linux version 4.19.86 (mi11p22@mi11p22-VirtualBox) (gcc version 9.3.0 (GCC)) #1 Wed Jun 8 22:26:20 CEST 2022
[    0.000000] CPU: ARMv7 Processor [410fd083] revision 3 (ARMv7), cr=30c5383d
[    0.000000] CPU: div instructions available: patching division code
[    0.000000] CPU: PIPT / VIPT nonaliasing data cache, PIPT instruction cache
[    0.000000] OF: fdt: Machine model: Raspberry Pi 4 Model B Rev 1.5
[    0.000000] Memory policy: Data cache writeback
[    0.000000] On node 0 totalpages: 294912
[    0.000000]   DMA zone: 256 pages used for memmap
[    0.000000]   DMA zone: 0 pages reserved
[    0.000000]   DMA zone: 32768 pages, LIFO batch:7
[    0.000000]   HighMem zone: 262144 pages, LIFO batch:63
[    0.000000] ------------[ cut here ]------------
[    0.000000] WARNING: CPU: 0 PID: 0 at arch/arm/kernel/devtree.c:155 arm_dt_init_cpu_maps+0x15c/0x1c0
[    0.000000] DT /cpu 2 nodes greater than max cores 1, capping them
[    0.000000] Modules linked in:
[    0.000000] CPU: 0 PID: 0 Comm: swapper Not tainted 4.19.86 #1
[    0.000000] Hardware name: BCM2835
[    0.000000] I-pipe domain: (null)
[    0.000000] [<c0211074>] (unwind_backtrace) from [<c020d46c>] (show_stack+0x20/0x24)
[    0.000000] [<c020d46c>] (show_stack) from [<c08558a0>] (dump_stack+0x24/0x28)
[    0.000000] [<c08558a0>] (dump_stack) from [<c02200c4>] (__warn.part.0+0xcc/0xe4)
[    0.000000] [<c02200c4>] (__warn.part.0) from [<c0220158>] (warn_slowpath_fmt+0x7c/0xa0)
[    0.000000] [<c0220158>] (warn_slowpath_fmt) from [<c0e0552c>] (arm_dt_init_cpu_maps+0x15c/0x1c0)
[    0.000000] [<c0e0552c>] (arm_dt_init_cpu_maps) from [<c0e04ff4>] (setup_arch+0x9b8/0xa20)
[    0.000000] [<c0e04ff4>] (setup_arch) from [<c0e00bc4>] (start_kernel+0x7c/0x4b8)
[    0.000000] [<c0e00bc4>] (start_kernel) from [<00000000>] ( (null))
[    0.000000] random: get_random_bytes called from print_oops_end_marker+0x34/0x5c with crng_init=0
[    0.000000] ---[ end trace 0000000000000000 ]---
[    0.000000] CPU: All CPU(s) started in HYP mode.
[    0.000000] CPU: Virtualization extensions available.
[    0.000000] pcpu-alloc: s0 r0 d32768 u32768 alloc=1*32768
[    0.000000] pcpu-alloc: [0] 0
[    0.000000] Built 1 zonelists, mobility grouping on.  Total pages: 294656
[    0.000000] Kernel command line: coherent_pool=1M 8250.nr_uarts=1 cma=64M bcm2708_fb.fbwidth=1920 bcm2708_fb.fbheight=1080 bcm2708_fb.fbswap=1 smsc95xx.macaddr=E4:5F:01:93:AC:5D vc_mem.mem_base=0xec00000 vc_mem.mem_size=0x10000000 console=ttyS0,115200 console=tty1 root=/dev/nfs nfsroot=192.168.0.1:/tftpboot/rootfs,vers=3 rw ip=dhcp rootwait elevator=deadline splash plymouth.ignore-serial-consoles dwc_otg.fiq_enable=0 dwc_otg.fiq_fsm_enable=0 dwc_otg.nak_holdoff=0 isolcpus=0,1
[    0.000000] Dentry cache hash table entries: 16384 (order: 4, 65536 bytes)
[    0.000000] Inode-cache hash table entries: 8192 (order: 3, 32768 bytes)
[    0.000000] Memory: 1154056K/1179648K available (8192K kernel code, 621K rwdata, 2112K rodata, 2048K init, 1072K bss, 25592K reserved, 0K cma-reserved, 1048576K highmem)
[    0.000000] Virtual kernel memory layout:
[    0.000000]     vector  : 0xffff0000 - 0xffff1000   (   4 kB)
[    0.000000]     fixmap  : 0xffc00000 - 0xfff00000   (3072 kB)
[    0.000000]     vmalloc : 0xc8800000 - 0xff800000   ( 880 MB)
[    0.000000]     lowmem  : 0xc0000000 - 0xc8000000   ( 128 MB)
[    0.000000]     pkmap   : 0xbfe00000 - 0xc0000000   (   2 MB)
[    0.000000]     modules : 0xbf000000 - 0xbfe00000   (  14 MB)
[    0.000000]       .text : 0x(ptrval) - 0x(ptrval)   (10208 kB)
[    0.000000]       .init : 0x(ptrval) - 0x(ptrval)   (2048 kB)
[    0.000000]       .data : 0x(ptrval) - 0x(ptrval)   ( 622 kB)
[    0.000000]        .bss : 0x(ptrval) - 0x(ptrval)   (1073 kB)
[    0.000000] SLUB: HWalign=64, Order=0-3, MinObjects=0, CPUs=1, Nodes=1
[    0.000000] NR_IRQS: 16, nr_irqs: 16, preallocated irqs: 16
[    0.000000] GIC: Using split EOI/Deactivate mode
[    0.000000] arch_timer: cp15 timer(s) running at 54.00MHz (phys).
[    0.000000] I-pipe, 54.000 MHz clocksource, wrap in 341606371735362 ms
[    0.000000] clocksource: ipipe_tsc: mask: 0   xffffffffffffffff max_cycles: 0xc743ce346, max_idle_ns: 440795203123 ns
[    0.000000] clocksource: arch_sys_counter: mask: 0 xffffffffffffff max_cycles: 0xc743ce346, max_idle_ns: 440795203123 ns
[    0.000005] sched_clock: 56 bits at 54MHz, resolution 18ns, wraps every 4398046511102ns
[    0.000022] Switching to timer-based delay loop, resolution 18ns
[    0.000525] Interrupt pipeline (release #6)
[    0.000654] Console: colour dummy device 80x30
[    0.001932] console [tty1] enabled
[    0.001997] Calibrating delay loop (skipped), value calculated using timer frequency.. 108.00 BogoMIPS (lpj=540000)
[    0.002054] pid_max: default: 32768 minimum: 301
[    0.002398] Mount-cache hash table entries: 1024 (order: 0, 4096 bytes)
[    0.002439] Mountpoint-cache hash table entries: 1024 (order: 0, 4096 bytes)
[    0.003055] CPU: Testing write buffer coherency: ok
[    0.003709] Setting up static identity map for 0x200000 - 0x20003c
[    0.005551] devtmpfs: initialized
[    0.019769] VFP support v0.3: implementor 41 architecture 3 part 40 variant 8 rev 0
[    0.020054] clocksource: jiffies : mask: 0 xffffffff max_cycles: 0 xffffffff , max_idle_ns: 19112604462750000 ns
[    0.020110] futex hash table entries: 256 (order: -1, 3072 bytes)
[    0.020262] pinctrl core: initialized pinctrl subsystem
[    0.021088] NET: Registered protocol family 16
[    0.022549] DMA: preallocated 1024 KiB pool for atomic coherent allocations
[    0.024286] hw-breakpoint: found 5 (+1 reserved) breakpoint and 4 watchpoint registers.
[    0.024330] hw-breakpoint: maximum watchpoint size is 8 bytes.
[    0.024551] Serial: AMBA PL011 UART driver
[    0.028099] bcm2835-mbox fe00b880.mailbox: mailbox enabled
[    0.070221] bcm2835-dma fe007000.dma: DMA legacy API manager at (ptrval), dmachans=0x1
[    0.072116] vgaarb: loaded
[    0.072500] SCSI subsystem initialized
[    0.072761] usbcore: registered new interface driver usbfs
[    0.072848] usbcore: registered new interface driver hub
[    0.072977] usbcore: registered new device driver usb
[    0.089947] raspberrypi-firmware soc:firmware: Attached to firmware from 2021-05-27 14:02, variant start
[    0.099960] raspberrypi-firmware soc:firmware: Firmware hash is 7d9a298cda813f747b51fe17e1e417e7bf5ca94d
[    0.111679] clocksource: Switched to clocksource ipipe_tsc
[    0.165358] VFS: Disk quotas dquot_6.6.0
[    0.165450] VFS: Dquot-cache hash table entries: 1024 (order 0, 4096 bytes)
[    0.165663] FS-Cache: Loaded
[    0.165863] CacheFiles: Loaded
[    0.175844] NET: Registered protocol family 2
[    0.176492] tcp_listen_portaddr_hash hash table entries: 512 (order: 0, 4096 bytes)
[    0.176546] TCP established hash table entries: 1024 (order: 0, 4096 bytes)
[    0.176589] TCP bind hash table entries: 1024 (order: 0, 4096 bytes)
[    0.176627] TCP: Hash tables configured (established 1024 bind 1024)
[    0.176759] UDP hash table entries: 256 (order: 0, 4096 bytes)
[    0.176799] UDP-Lite hash table entries: 256 (order: 0, 4096 bytes)
[    0.176996] NET: Registered protocol family 1
[    0.177537] RPC: Registered named UNIX socket transport module.
[    0.177577] RPC: Registered udp transport module.
[    0.177605] RPC: Registered tcp transport module.
[    0.177633] RPC: Registered tcp NFSv4.1 backchannel transport module.
[    0.181000] [Xenomai] scheduling class idle registered.
[    0.181040] [Xenomai] scheduling class rt registered.
[    0.181170] I-pipe: head domain Xenomai registered.
[    0.189174] [Xenomai] Cobalt v3.1
[    0.189207] Initialise system trusted keyrings
[    0.189355] workingset: timestamp_bits=30 max_order=19 bucket_order=0
[    0.199818] FS-Cache: Netfs 'nfs' registered for caching
[    0.200534] NFS: Registering the id_resolver key type
[    0.200586] Key type id_resolver registered
[    0.200615] Key type id_legacy registered
[    0.200653] nfs4filelayout_init: NFSv4 File Layout Driver Registering...
[    0.202936] Key type asymmetric registered
[    0.202970] Asymmetric key parser 'x509' registered
[    0.203115] bounce: pool size: 64 pages
[    0.203187] Block layer SCSI generic (bsg) driver version 0.4 loaded (major 248)
[    0.203229] io scheduler noop registered
[    0.203256] io scheduler deadline registered (default)
[    0.203445] io scheduler cfq registered
[    0.203475] io scheduler mq-deadline registered (default)
[    0.203506] io scheduler kyber registered
[    0.206617] brcm-pcie fd500000.pcie: could not get clock
[    0.206704] brcm-pcie fd500000.pcie: host bridge /scb/pcie@7d500000 ranges:
[    0.206769] brcm-pcie fd500000.pcie:   MEM 0x600000000..0x603fffffff -> 0xf8000000
[    0.261722] brcm-pcie fd500000.pcie: link up, 5.0 Gbps x1 (!SSC)
[    0.262028] brcm-pcie fd500000.pcie: PCI host bridge to bus 0000:00
[    0.262069] pci_bus 0000:00: root bus resource [bus 00-01]
[    0.262106] pci_bus 0000:00: root bus resource [mem 0x600000000-0x603fffffff] (bus address [0xf8000000-0xfbfffffff])
[    0.262190] pci 0000:00:00.0: [14e4:2711] type 01 class 0x060400
[    0.262331] pci 0000:00:00.0: PME# supported from D0 D3hot
[    0.267025] PCI: bus0: Fast back to back transfers disabled
[    0.267067] pci 0000:00:00.0: bridge configuration invalid ([bus
```

```
00-00]), reconfiguring
[    0.267377] pci 0000:01:00.0: [1106:3483] type 00 class 0x0c0330
[    0.267475] pci 0000:01:00.0: reg 0x10: [mem 0x00000000-0x00000fff 64bit]
[    0.267759] pci 0000:01:00.0: PME# supported from D0 D3hot
[    0.272471] PCI: bus1: Fast back to back transfers disabled
[    0.272511] pci_bus 0000:01: busn_res: [bus 01] end is updated to 01
[    0.272561] pci 0000:00:00.0: BAR 8: assigned [mem 0x600000000-0x6000fffff]
[    0.272608] pci 0000:01:00.0: BAR 0: assigned [mem 0x600000000-0x600000fff 64bit]
[    0.272698] pci 0000:00:00.0: PCI bridge to [bus 01]
[    0.272735] pci 0000:00:00.0:   bridge window [mem 0x600000000-0x6000fffff]
[    0.272984] pcieport 0000:00:00.0: enabling device (0140 -> 0142)
[    0.273185] pcieport 0000:00:00.0: Signaling PME with IRQ 54
[    0.273416] pcieport 0000:00:00.0: AER enabled with IRQ 54
[    0.273951] bcm2708_fb soc:fb: FB found 1 display(s)
[    0.289094] Console: switching to colour frame buffer device 240x67
[    0.299700] bcm2708_fb soc:fb: Registered framebuffer for display 0, size 1920x1080
[    0.302152] Serial: 8250/16550 driver, 1 ports, IRQ sharing enabled
[    0.304574] iproc-rng200 fe104000.rng: hwrng registered
[    0.304896] vc-mem: phys_addr:0x00000000 mem_base=0x0ec00000 mem_size:0x10000000(256 MiB)
[    0.305588] vc-sm: Videocore shared memory driver
[    0.306094] gpiomem-bcm2835 fe200000.gpiomem: Initialised: Registers at 0xfe200000
[    0.316536] brd: module loaded
[    0.325085] loop: module loaded
[    0.326404] spi-bcm2835 fe204000.spi: could not get clk: -517
[    0.328196] libphy: Fixed MDIO Bus: probed
[    0.328730] bcmgenet fd580000.genet: failed to get enet clock
[    0.328818] bcmgenet fd580000.genet: GENET 5.0 EPHY: 0x0000
[    0.328902] bcmgenet fd580000.genet: failed to get enet-wol clock
[    0.328988] bcmgenet fd580000.genet: failed to get enet-eee clock
[    0.329082] bcmgenet: Skipping UMAC reset
[    0.329402] unimac-mdio unimac-mdio.-19: DMA mask not set
[    0.341742] libphy: bcmgenet MII bus: probed
[    0.382287] unimac-mdio unimac-mdio.-19: Broadcom UniMAC MDIO bus at 0x(ptrval)
[    0.383613] dwc_otg: version 3.00a 10-AUG-2012 (platform bus)
[    0.383889] dwc_otg: FIQ disabled
[    0.383898] dwc_otg: NAK holdoff disabled
[    0.383907] dwc_otg: FIQ split-transaction FSM disabled
[    0.383921] Module dwc_common_port init
[    0.384114] usbcore: registered new interface driver uas
[    0.384249] usbcore: registered new interface driver usb-storage
[    0.384549] mousedev: PS/2 mouse device common for all mice
[    0.386504] bcm2835-wdt bcm2835-wdt: Broadcom BCM2835 watchdog timer
[    0.386662] ledtrig-cpu: registered to indicate activity on CPUs
[    0.387468] vchiq: vchiq_init_state: slot_zero = (ptrval), is_master = 0
[    0.388837] [vc_sm_connected_init]: start
[    0.395174] [vc_sm_connected_init]: end - returning 0
[    0.397105] Initializing XFRM netlink socket
[    0.397197] NET: Registered protocol family 17
[    0.397334] Key type dns_resolver registered
[    0.398641] registered taskstats version 1
[    0.398713] Loading compiled-in X.509 certificates
[    0.410256] uart-pl011 fe201000.serial: cts_event_workaround enabled
[    0.414133] fe201000.serial: ttyAMA0 at MMIO 0xfe201000 (irq = 34, base_baud = 0) is a PL011 rev2
[    0.421477] console [ttyS0] disabled
[    0.425325] fe215040.serial: ttyS0 at MMIO 0x0 (irq = 37, base_baud = 62500000) is a 16550
[    2.192731] console [ttyS0] enabled
[    2.200570] bcm2835-power bcm2835-power: Broadcom BCM2835 power domains driver
[    2.213246] mcp320x spi0.1: spi0.1 supply vref not found, using dummy regulator
[    2.224657] mcp320x spi0.1: Linked as a consumer to regulator.0
[    2.235868] input: joypinote_keypad as /devices/platform/joypinote_keypad/input/input0
[    2.250457] brcmstb_thermal fd5d2200.thermal: registered AVS TMON of-sensor driver
[    2.480202] random: fast init done
[    2.580744] of_cfs_init
[    2.587332] of_cfs_init: OK
[    2.594589] bcmgenet: Skipping UMAC reset
[    2.702244] bcmgenet fd580000.genet: configuring instance for external RGMII (no delay)
[    3.761880] bcmgenet fd580000.genet eth0: Link is Down
[    7.921904] bcmgenet fd580000.genet eth0: Link is Up - 1Gbps/Full - flow control rx/tx
[    7.961716] Sending DHCP requests ., OK
[    8.029500] IP-Config: Got DHCP answer from 192.168.0.1, my address is 192.168.0.53
[    8.041221] IP-Config: Complete:
[    8.048406]      device=eth0, hwaddr=e4:5f:01:93:ac:5d, ipaddr=192.168.0.53, mask=255.255.255.0, gw=255.255.255.255
[    8.063019]      host=192.168.0.53, domain=, nis-domain=(none)
[    8.072947]      bootserver=0.0.0.0, rootserver=192.168.0.1, rootpath=
[    8.072958]      nameserver0=192.168.0.1
[    8.106919] VFS: Mounted root (nfs filesystem) on device 0:15.
[    8.117785] devtmpfs: mounted
[    8.127873] Freeing unused kernel memory: 2048K
[    8.136717] Run /sbin/init as init process
[    9.306746] udevd[91]: starting version 3.2.9
[    9.333237] random: udevd: uninitialized urandom read (16 bytes read)
[    9.341632] random: udevd: uninitialized urandom read (16 bytes read)
[    9.351516] random: udevd: uninitialized urandom read (16 bytes read)
[    9.436824] udevd[91]: specified group 'xenomai' unknown
[    9.446066] udevd[92]: starting eudev-3.2.9
[    9.724802] vc_sm_cma: module is from the staging directory, the quality is unknown, you have been warned.
[    9.788021] bcm2835_vc_sm_cma_probe: Videocore shared memory driver
[    9.846630] [vc_sm_connected_init]: start
[    9.876805] [vc_sm_connected_init]: installed successfully
[   11.448824] urandom_read: 1 callbacks suppressed
[   11.448832] random: dd: uninitialized urandom read (512 bytes read)
[   12.164011] random: dbus-daemon: uninitialized urandom read (12 bytes read)
[   12.205469] random: dbus-daemon: uninitialized urandom read (12 bytes read)
[   12.982147] urandom_read: 1 callbacks suppressed
[   12.982156] random: avahi-daemon: uninitialized urandom read (4 bytes read)
[   13.014861] random: avahi-daemon: uninitialized urandom read (4 bytes read)
[   38.081810] random: crng init done
[ 6914.321476] rtdm_gpio: no symbol version for module_layout
[ 6914.337644] rtdm_gpio: loading out-of-tree module taints kernel.
[ 6914.355442] rtgpio_init
[ 6942.222167] rtgpio_exit
```

Le périphérique se trouve dans le directoire `/dev/rtdm/` :

```
root@joypinote-xenomai:/# ls /dev/rtdm/
autotune        heapcheck       memdev-private memdev-shared memdev-sys  switchtest      timerbench
root@joypinote-xenomai:/# insmod usr/rtdm_gpio.ko
root@joypinote-xenomai:/# ls /dev/rtdm/
autotune        heapcheck       memdev-private memdev-shared memdev-sys  rtgpio          switchtest
        timerbench
root@joypinote-xenomai:/# rmmod /usr/rtdm_gpio.ko
root@joypinote-xenomai:/# ls /dev/rtdm/
autotune        heapcheck       memdev-private memdev-shared memdev-sys  switchtest      timerbench
root@joypinote-xenomai:/#
```

2.1.2 Programme d'interaction RTDM

Code de notre programme :

```c
// Includes
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <alchemy/task.h>

// Constants
#define TASK_PRIO 99
#define TASK_MODE T_JOINABLE
#define TASK_STKSZ 0

// RT task that opens and closes GPIO
void file_task() {
   int fd = open("/dev/rtdm/rtgpio", O_RDONLY);
   if (fd ==-1)
   {
      rt_printf("error rt_task : open");
   }
   rt_printf("Ouverture\n");
   rt_task_sleep(10000000000);
   close(fd);
   rt_printf("Fermeture\n");
}


int main() {
   int err;
   RT_TASK task_desc;

   // Task creation
   err = rt_task_create(&task_desc, "file",
       TASK_STKSZ, TASK_PRIO, TASK_MODE);
   if (err != 0) {
      printf("error rt_task_create\n");
      return 1;
   }

   // Start task, wait for it and exit
   rt_task_start(&task_desc, &file_task, NULL);
   rt_task_join(&task_desc);
   rt_task_delete(&task_desc);
   return 0;
}
```

Résultats dans les logs :

```
[ 7917.139865] rtgpio_init
[ 7920.898502] rtgpio_open
[ 7930.911350] rtgpio_close
```

2.2 Registres

2.2.1 Adresse de base

D'après la documentation qui nous est fournie, l'adresse de base de registre GPIO que nous devons utiliser est `0x0_FC00_0000`.

2.2.2 Configuration GPIO en sortie

Il faut modifier le mot de 32 bits `GPFSEL0` situé à l'offset `0x00`. Pour mettre à jour le GPIO numéro i ($0 \leq i \leq 9$), on applique le masque 001 décalé de $3 \cdot i$ bits sur le mot.

2.2.3 Mise à 1 du GPIO

On se concentre maintenant sur le mot de 32 bits `GPSET0`, situé à l'offset `0x1C`. Pour activer le GPIO i, on passe le $n^{\text{ème}}$ bit à 1.

2.2.4 Code du driver

```c
#include <linux/module.h>
#include <rtdm/driver.h>
#include "rtdm_gpio.h"


MODULE_LICENSE("GPL");


//fonction appelee lors de l'ouverture du peripherique, on ne fait rien ici
int rtgpio_open(struct rtdm_fd *fd, int oflags) {
    rtdm_printk("rtgpio_open\n");
    return 0;
}


//fonction appelee lors de la fermeture du peripherique, on ne fait rien ici
void rtgpio_close(struct rtdm_fd *fd) {
    rtdm_printk("rtgpio_close\n");
}


// 0x0_FC00_0000
//fonction configurant un GPIO comme output, a completer
int rtgpio_direction_output(unsigned char gpio) {
    rtdm_printk("rtgpio_direction_output %i\n",gpio);

    if(gpio<10) {


        unsigned long base=(unsigned long)ioremap(0xFE200000, 4);
        int val=readl((void *)base + 0x00);


        val &= ~(0b111<<3*gpio);
        val |= (0b1 << 3*gpio);


        writel(val,(void *)base + 0x00);


        return 0;
    } else {
        rtdm_printk("rtgpio_direction_output, invalid gpio number %i\n",gpio);
        return -1;
    }
}


//fonction fixant la valeur d'un GPIO, a completer
int rtgpio_set_value(unsigned char gpio,bool value) {
    if(gpio<10) {
        unsigned long base=(unsigned long)ioremap(0xFE200000, 4);


        if(value)
            writel(1UL << gpio,(void *)base + 0x1c);
        else
            writel(1UL << gpio,(void *)base + 0x28);


        return 0;
    } else {
        rtdm_printk("rtgpio1_set_value, invalid gpio number %i\n",gpio);
        return -1;
    }
}


//ioctl pour interagir avec le programme en mode utilisateur
int rtgpio_ioctl(struct rtdm_fd *fd,unsigned int request, void *arg) {
    unsigned char pin = (long)arg;


    switch (request) {
    case RTGPIO_SET_DIRECTION_OUTPUT:
        return rtgpio_direction_output(pin);
        break;
    case RTGPIO_SET:
        return rtgpio_set_value(pin,true);
        break;
    case RTGPIO_CLEAR:
        return rtgpio_set_value(pin,false);
        break;
    default:
        rtdm_printk("rtgpio_ioctl, unsupported request %i\n",request);
    }


    return -1;
}


//structure du peripherique RTDM
//elle permet de nommer le peripherique, et d'associer les fonctions rtgpio_open, rtgpio_close, rtgpio_ioctl aux appels RTDM cote user space
static struct rtdm_driver rtgpio_driver = {
    .profile_info     =
        RTDM_PROFILE_INFO(rtgpio, //name
                          RTDM_CLASS_EXPERIMENTAL, //major
                          1, //minor
                          1), //version
```

```
    .device_flags     =
        RTDM_NAMED_DEVICE|RTDM_EXCLUSIVE,
    .device_count     =   1,
    .ops = {
        .open      =   rtgpio_open,
        .close     =   rtgpio_close,
        .ioctl_rt  =   rtgpio_ioctl,
    },
};


static struct rtdm_device rtgpio_device = {
    .driver = &rtgpio_driver,
    .label = "rtgpio",
};


//fonction appelee lors du chargement du
     module (insmod)
int __init rtgpio_init(void) {
    int err;

    rtdm_printk("rtgpio_init\n");


    //enregistrement du driver
    err = rtdm_dev_register(&rtgpio_device);
    if (err)
        return err;



    return 0;
}


//fonction appelee lors du dechargement du
     module (rmmod)
void rtgpio_exit(void) {
    rtdm_printk("rtgpio_exit\n");

    //desenregistrement du driver
    rtdm_dev_unregister(&rtgpio_device);
}


module_init(rtgpio_init);
module_exit(rtgpio_exit);
```

3 Clignotement

3.1 Premier programme

```
// Includes
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <alchemy/task.h>
#include "../rtdm_gpio/rtdm_gpio.h"

// Constants define
#define TASK_PRIO 99
#define TASK_MODE T_JOINABLE
#define TASK_STKSZ 0

// RT task that blinks LED
void file_task() {
   // Open GPIO file
   int fd = open("/dev/rtdm/rtgpio", O_RDONLY);
   if (fd ==-1)
   {
      rt_printf("error rt_task : open");
   }
   rt_printf("Ouverture\n");
   ioctl(fd, RTGPIO_SET_DIRECTION_OUTPUT, 5);

   // Infinite loop
   for(;;){
      // Turn off led
      ioctl(fd, RTGPIO_CLEAR, 5);
      rt_task_sleep(500000000);
      // Turn on led
      ioctl(fd, RTGPIO_SET, 5);
      rt_task_sleep(500000000);
   }
   close(fd);
   rt_printf("Fermeture\n");
}


int main() {
   int err;
   RT_TASK task_desc;

   // Create task
   err = rt_task_create(&task_desc, "file",
       TASK_STKSZ, TASK_PRIO, TASK_MODE);
   if (err != 0) {
      printf("error rt_task_create\n");
      return 1;
   }

   // Wait for task to finish and exit program
   rt_task_start(&task_desc, &file_task, NULL);
   rt_task_join(&task_desc);
   rt_task_delete(&task_desc);
   return 0;
}
```

3.2 Clignotement cyclique

```c
// Includes
#include <stdio.h>
#include <unistd.h>
#include <alchemy/task.h>
#include "../TP_Xen_3/rtdm_gpio/rtdm_gpio.h"

// Constants
#define TASK_PRIO 99
#define TASK_MODE T_JOINABLE
#define TASK_STKSZ 0
#define FREQ 10
#define RAPPORT 0.75

// RT task
void file_task() {
   // Open GPIO file
   int fd = open("/dev/rtdm/rtgpio", O_RDONLY);
   if (fd ==-1)
   {
      rt_printf("error rt_task : open");
   }
   rt_printf("Ouverture\n");
   ioctl(fd, RTGPIO_SET_DIRECTION_OUTPUT, 5);

   // Defininf cycle
   int duree_cycle = 1000000000/FREQ;
   int duree_on = duree_cycle * RAPPORT;
   int duree_off = duree_cycle * (1-RAPPORT);
   // Infinite loop
   for(;;){
      // Turn on LED
      ioctl(fd, RTGPIO_CLEAR, 5);
      rt_task_sleep(duree_on);
      // Turf off LED
      ioctl(fd, RTGPIO_SET, 5);
      rt_task_sleep(duree_off);
   }
   close(fd);
   rt_printf("Fermeture\n");
}


int main() {
   int err;
   RT_TASK task_desc;

   // Create task
   err = rt_task_create(&task_desc, "file",
       TASK_STKSZ, TASK_PRIO, TASK_MODE);
   if (err != 0) {
      printf("error rt_task_create\n");
      return 1;
   }

   // Wait for task to exit and exit program
   rt_task_start(&task_desc, &file_task, NULL);
   rt_task_join(&task_desc);
   rt_task_delete(&task_desc);
   return 0;
}
```

3.3 RT_PIPE

```c
// Includes
#include <stdio.h>
#include <unistd.h>
#include <alchemy/task.h>
#include <alchemy/pipe.h>
#include "../TP_Xen_3/rtdm_gpio/rtdm_gpio.h"

// Constants
#define TASK_PRIO 99
#define TASK_MODE T_JOINABLE
#define TASK_STKSZ 0
#define FREQ 10
#define RAPPORT 0.5
#define MSG_SIZE 50

// Pipe descriptos
RT_PIPE pipe_desc;


void file_task() {
   // Message buffer
   char msg[MSG_SIZE];

   // Open GPIO file
   int fd = open("/dev/rtdm/rtgpio", O_RDONLY);
   if (fd ==-1)
   {
      rt_printf("error rt_task : open\n");
   }
   rt_printf("Ouverture\n");
   ioctl(fd, RTGPIO_SET_DIRECTION_OUTPUT, 5);

   // Define blinking cycles
   int duree_cycle = 1000000000/FREQ;
   int duree_on = duree_cycle * RAPPORT;
   int duree_off = duree_cycle * (1-RAPPORT);

   // Infinite for loop
   for(;;){
      // Read pipe
      int nb = rt_pipe_read(&pipe_desc, &msg,
          MSG_SIZE, TM_NONBLOCK);
      // Convert message to int
      int nfreq = atoi(&msg);
      // Redefine blinking frequency
```

```
45      if(nfreq != 0){
46         rt_printf("Super if\n");
47         duree_cycle = 1000000000/nfreq;
48         duree_on = duree_cycle * RAPPORT;
49         duree_off = duree_cycle * (1-RAPPORT);
50      }
51      // Turn LED on
52      ioctl(fd, RTGPIO_CLEAR, 5);
53      rt_task_sleep(duree_on);
54      // Turn LED off
55      ioctl(fd, RTGPIO_SET, 5);
56      rt_task_sleep(duree_off);
57   }
58   close(fd);
59   rt_printf("Fermeture\n");
60 }
61
62
63 int main() {
64   int err;
65   RT_TASK task_desc;
66   int minor = 1;
67   char msg[MSG_SIZE];
68
69   // Create task
70   err = rt_task_create(&task_desc, "file",
         TASK_STKSZ, TASK_PRIO, TASK_MODE);
71   if (err != 0) {
72      printf("error rt_task_create\n");
73      return 1;
74   }
75
76   // Create pipe
77   err = rt_pipe_create(&pipe_desc, "tuyau",
         minor, 0) ;
78   if (err != minor) {
79      printf("error rt_pipe_create\n");
80      rt_task_delete(&task_desc);
81      return 1;
82   }
83
84   // Open pipe file
85   int fd = open("/dev/rtp1", O_WRONLY);
86   if (fd ==-1)
87   {
88      rt_task_delete(&task_desc);
89      rt_pipe_delete(&pipe_desc);
90   }
91
92   // Start task
93   rt_task_start(&task_desc, &file_task, NULL);
94
95   // Infinite for loop
96   for(;;){
97      // Scan entry from user
98      scanf(" %s", &msg);
99      // Send input to pipe
100     write(fd, &msg, MSG_SIZE);
101     sleep(1);
102  }
103
104  // Wait for task to end and exit program
105  rt_task_join(&task_desc);
106  rt_task_delete(&task_desc);
107  rt_pipe_delete(&pipe_desc);
108  return 0;
109 }
```

4 Robustesse

Après lancement de la commande donnée dans le sujet. Plus rien ne répond, mais la LED continue de clignoter. Ainsi, il n'est plus possible d'interagir avec le système, mais celui-ci maintient son fonctionnement actuel.

Ceci est dû au fait que c'est le noyau `Linux` qui crash (empêchant toute interaction avec le système), sans altérer le fonctionnement du noyau `Xenomai`, qui continue ainsi de faire tourner nos tâches temps-réel.