

Banc GSM émulé — dossier de run

`/opt/GSM/qosmo-overlay`

Banc GSM émulé — banc-max

2026-10-25

Table des matières

0.1	Synthèse	3
0.2	Resultats des tests	4
0.2.1	<code>/opt/GSM/qosmo-overlay/README.md</code>	4
0.2.2	<code>/opt/GSM/qosmo-overlay/meson_options.txt</code>	11
0.2.3	<code>/opt/GSM/qosmo-overlay/environnement/README.md</code>	21
0.2.4	<code>/opt/GSM/qosmo-overlay/hw/arm/calypso/doc/SONDES.md</code>	23
0.2.5	<code>/opt/GSM/qosmo-overlay/tests/detail.mmd</code>	27
0.2.6	<code>/opt/GSM/qosmo-overlay/tests/detail.qmd.mmd</code>	27
0.2.7	<code>/opt/GSM/qosmo-overlay/tests/full.mmd</code>	29
0.2.8	<code>/opt/GSM/qosmo-overlay/tests/pipeline.mmd</code>	29
0.2.9	<code>/opt/GSM/qosmo-overlay/tests/pipeline.qmd.mmd</code>	29
0.2.10	<code>/opt/GSM/qosmo-overlay/tmux_modules/README.md</code>	30
0.3	Fichiers	31
0.3.1	<code>/opt/GSM/qosmo-overlay/install.sh</code>	31
0.3.2	<code>/opt/GSM/qosmo-overlay/make-overlay.sh</code>	44
0.3.3	<code>/opt/GSM/qosmo-overlay/run.sh</code>	46
0.3.4	<code>/opt/GSM/qosmo-overlay/tout-en-un.py</code>	54
0.3.5	<code>/opt/GSM/qosmo-overlay/opt-gsm-scripts/gsm_sniff.py</code>	60
0.3.6	<code>/opt/GSM/qosmo-overlay/run_modules/00-prereqs.sh</code>	64
0.3.7	<code>/opt/GSM/qosmo-overlay/run_modules/02-outils.sh</code>	64
0.3.8	<code>/opt/GSM/qosmo-overlay/run_modules/03-rundir.sh</code>	66
0.3.9	<code>/opt/GSM/qosmo-overlay/run_modules/04-restes.sh</code>	68
0.3.10	<code>/opt/GSM/qosmo-overlay/run_modules/05-config.sh</code>	70
0.3.11	<code>/opt/GSM/qosmo-overlay/run_modules/08-gabarits.sh</code>	72
0.3.12	<code>/opt/GSM/qosmo-overlay/run_modules/09-logs.sh</code>	75
0.3.13	<code>/opt/GSM/qosmo-overlay/run_modules/09-teardown.sh</code>	77
0.3.14	<code>/opt/GSM/qosmo-overlay/run_modules/10-reseau.sh</code>	80

0.3.15	/opt/GSM/qosmo-overlay/run_modules/10-stp.sh	83
0.3.16	/opt/GSM/qosmo-overlay/run_modules/11-hlr.sh	85
0.3.17	/opt/GSM/qosmo-overlay/run_modules/12-mgw.sh	86
0.3.18	/opt/GSM/qosmo-overlay/run_modules/13-msc.sh	88
0.3.19	/opt/GSM/qosmo-overlay/run_modules/13-sidecar-cfg.sh	90
0.3.20	/opt/GSM/qosmo-overlay/run_modules/14-bsc.sh	93
0.3.21	/opt/GSM/qosmo-overlay/run_modules/15-apn0.sh	95
0.3.22	/opt/GSM/qosmo-overlay/run_modules/16-ggsn.sh	96
0.3.23	/opt/GSM/qosmo-overlay/run_modules/17-sgsn.sh	98
0.3.24	/opt/GSM/qosmo-overlay/run_modules/18-pcu.sh	99
0.3.25	/opt/GSM/qosmo-overlay/run_modules/19-asterisk.sh	101
0.3.26	/opt/GSM/qosmo-overlay/run_modules/20-mobile-cfg.sh	102
0.3.27	/opt/GSM/qosmo-overlay/run_modules/20-sip-connector.sh	105
0.3.28	/opt/GSM/qosmo-overlay/run_modules/21-abonnes-hlr.sh	106
0.3.29	/opt/GSM/qosmo-overlay/run_modules/22-smsc.sh	109
0.3.30	/opt/GSM/qosmo-overlay/run_modules/25-audio.sh	111
0.3.31	/opt/GSM/qosmo-overlay/run_modules/35-tmux.sh	112
0.3.32	/opt/GSM/qosmo-overlay/run_modules/38-bruit.sh	114
0.3.33	/opt/GSM/qosmo-overlay/run_modules/40-qemu.sh	119
0.3.34	/opt/GSM/qosmo-overlay/run_modules/41-pty.sh	122
0.3.35	/opt/GSM/qosmo-overlay/run_modules/44-gdb-telnet.sh	124
0.3.36	/opt/GSM/qosmo-overlay/run_modules/50-osmocon.sh	126
0.3.37	/opt/GSM/qosmo-overlay/run_modules/58-sidecar-faketrx.sh	128
0.3.38	/opt/GSM/qosmo-overlay/run_modules/59-sidecar-bts.sh	130
0.3.39	/opt/GSM/qosmo-overlay/run_modules/60-bts.sh	131
0.3.40	/opt/GSM/qosmo-overlay/run_modules/61-sidecar-pcu.sh	134
0.3.41	/opt/GSM/qosmo-overlay/run_modules/67-sidecar-trxcon.sh	136
0.3.42	/opt/GSM/qosmo-overlay/run_modules/68-sidecar-mobile.sh	137
0.3.43	/opt/GSM/qosmo-overlay/run_modules/69-modem-trxcon.sh	139
0.3.44	/opt/GSM/qosmo-overlay/run_modules/70-l2.sh	140
0.3.45	/opt/GSM/qosmo-overlay/run_modules/71-modem-gprs.sh	142
0.3.46	/opt/GSM/qosmo-overlay/run_modules/75-gsmtap.sh	146
0.3.47	/opt/GSM/qosmo-overlay/run_modules/80-panes.sh	147
0.3.48	/opt/GSM/qosmo-overlay/run_modules/95-summary.sh	149
0.3.49	/opt/GSM/qosmo-overlay/run_modules/99-attach.sh	150
0.3.50	/opt/GSM/qosmo-overlay/run_modules/_lib/core.sh	151
0.3.51	/opt/GSM/qosmo-overlay/run_modules/_lib/i18n.sh	155
0.3.52	/opt/GSM/qosmo-overlay/run_modules/_lib/mod.sh	170
0.3.53	/opt/GSM/qosmo-overlay/run_modules/_lib/modem.sh	172
0.3.54	/opt/GSM/qosmo-overlay/run_modules/_lib/radio.sh	173
0.3.55	/opt/GSM/qosmo-overlay/run_modules/_lib/temps_reel.sh	174
0.3.56	/opt/GSM/qosmo-overlay/scripts/inject_fb.py	175
0.3.57	/opt/GSM/qosmo-overlay/scripts/make_dsp_bin_L1.py	178

0.3.58	/opt/GSM/qosmo-overlay/scripts/meson-buildoptions.sh	183
0.3.59	/opt/GSM/qosmo-overlay/scripts/populate-si.sh	197
0.3.60	/opt/GSM/qosmo-overlay/scripts/replay_pm_payload.py	200
0.3.61	/opt/GSM/qosmo-overlay/scripts/test_record_replay.sh	205
0.3.62	/opt/GSM/qosmo-overlay/scripts/qosmo/test-base.sh	208
0.3.63	/opt/GSM/qosmo-overlay/scripts/qosmo/verif-liens-doc.sh	209
0.3.64	/opt/GSM/qosmo-overlay/tests/run_tests.sh	211
0.3.65	/opt/GSM/qosmo-overlay/tests/test_runtime_net_fs.py	212
0.3.66	/opt/GSM/qosmo-overlay/tests/test_timer_invariants.py	214
0.3.67	/opt/GSM/qosmo-overlay/tests/test_timer_physical_audit.py	221
0.3.68	/opt/GSM/qosmo-overlay/tmux_modules/_commun.sh	228
0.3.69	/opt/GSM/qosmo-overlay/tmux_modules/bench.sh	231
0.3.70	/opt/GSM/qosmo-overlay/tools/demo-layer1.py	232
0.3.71	/opt/GSM/qosmo-overlay/tools/gdb-telnet.py	236
0.3.72	/opt/GSM/qosmo-overlay/tools/gdb_cmd.sh	241
0.3.73	/opt/GSM/qosmo-overlay/tools/injecteur_bruit.py	259
0.3.74	/opt/GSM/qosmo-overlay/tools/qosmo-couleurs.py	276
0.3.75	/opt/GSM/qosmo-overlay/tools/rapport-run.sh	278
0.3.76	/opt/GSM/qosmo-overlay/tools/qosmo-tbring/tbring-dump.py	282

0.1 Synthèse

run	élément	valeur
(pas de verdict.txt)		

dossier de run	contenu
Verdict et couverture	échelle des barreaux et tableau de couverture couche 1
Résultats des tests	captures VTY, diagnostics, rapports pytest, grafjets (.txt .md .mmd)
Fichiers	scripts du banc et des tests (.sh .py)
Logs	journaux (.log), en dernier

i Méthode

Dossiers : /opt/GSM/qosmo-overlay. Extensions : log sh md mmd py txt. Entrée 532 Ko, sortie 524 Ko.

Compactage sans perte : fichiers identiques cités une fois ; lignes consécutives identiques « ×N » ; lignes ne différant que par des nombres : gabarit avec `k` puis la liste ordonnée des valeurs (k-uplets), chaque ligne se reconstruit en remettant les valeurs dans l'ordre. Seuls les codes couleur, la négociation telnet des captures VTY et les caractères de commande sont retirés. Les `.md` sont inclus tels quels (titres rétrogradés) et les `.mmd` rendus en diagrammes Mermaid, sans compactage.

0.2 Resultats des tests

0.2.1 /opt/GSM/qosmo-overlay/README.md

17555 octets, 310 lignes

0.2.1.1 qosmo — QEMU Calypso, une seule base, deux couches 1

Modèle QEMU du **baseband GSM TI Calypso** (Compal E88 / Motorola C1xx). Le firmware [osmocom-bb](#) d'origine, non patché, y boote et y tourne.

La couche 1 se choisit **au configure**, pas en changeant de dépôt :

```
../configure --target-list=arm-softmmu           # plateforme nue
../configure --target-list=arm-softmmu --enable-l1-grgsm  # démod par gr-gsm
../configure --target-list=arm-softmmu --enable-l1-dsp    # mask-ROM TI sur C54x
```

0.2.1.1.1 Pourquoi ce dépôt existe

qemu-calypso avait été scindé le 2026-09-01 en deux forks — `qosmo-grgsm` et `qosmo-dsp` — pour répondre à deux questions différentes : *la pile marche-t-elle de bout en bout ?* et *le vrai DSP acquiert-il ?* Le problème n'était pas la scission, c'était sa forme : **deux dépôts sans commit racine commun** (6ee0024 contre 472f69d). Ni merge, ni cherry-pick, ni base de fusion. Un correctif de plateforme devait être réappliqué à la main dans l'autre arbre — donc, en pratique, ne l'était pas.

Résultat mesuré le 2026-09-16, deux semaines après le split : **9 des 10 fichiers communs à hw/arm/calypso/ avaient divergé**, `calypso_trx.c` allant jusqu'à 478 lignes d'un côté contre 2478 de l'autre. Seul `Kconfig` était resté identique.

Mais cette divergence, mesurée en retirant les deux couches 1, n'était presque pas de la plateforme. C'était le même squelette préfixé différemment (`tpu_read` contre `calypso_tpu_read`, `cpu_kick` contre `calypso_kick_cb`), plus **deux fonctions qui sont la couche 1 déguisée en accès mémoire** : `api_read/api_write` sur la fenêtre `0xFFD00000`. Le SoC, lui, était le même des deux côtés — il l'est toujours, c'est le même silicium.

D'où ce dépôt : une plateforme, une couture explicite, deux L1 qui cohabitent et ne se disputent qu'une ligne de `configure`.

0.2.1.1.2 La couture

Neuf points d'appel, une structure : `calypso_l1_ops.h`.

```
static const CalypsoL1Ops mes_ops = {
    .name           = "grgsm",
    .init           = ma_l1_init,           /* fin de machine_init */
    .frame_tick     = ma_l1_frame_tick,    /* 1× par trame TDMA */
    .api_read_override = ma_l1_read_override, /* lecture API RAM */
    .si_valid       = ma_l1_si_valid,
    .l1s_fn         = ma_l1s_fn,
    .burst_written  = ma_l1_burst_written, /* d_burst_d */
    .rach_written   = ma_l1_rach_written,  /* d_rach */
    .page_written   = ma_l1_page_written,  /* d_dsp_page */
    .uart_tx_byte   = mon_tap_l1ctl,       /* optionnel */
};
static void enregistrer(void) { calypso_l1_register(&mes_ops); }
type_init(enregistrer)
```

Elle ne porte **que** le sens plateforme → L1. Le sens inverse reste en appels directs vers ce que la base exporte (`calypso_api_ram()`, `calypso_trx_get_fn()`, `calypso_trx_autosync_fn()`, `calypso_tsp_move()`, `calypso_tpu_run_scenario()`) : c'est du matériel, il n'en existe qu'une implémentation par construction, l'indirecter n'achèterait rien.

Le test du découpage a été de faire passer la L1 de `qosmo-grgsm` dessous : `calypso_l1_grgsm.c` et `calypso_l1ctl_tap.c` sont repris **sans aucune modification** hors le chemin de leurs deux `#include` privés. Tout le coût d'adaptation tient dans `calypso_l1_grgsm_register.c`, qui ne fait que remplir la structure. Si la couture avait été mal placée, il aurait fallu retoucher la L1 elle-même.

La couture sait aussi se retirer : `calypso_l1_disable(why)` (`calypso_l1_ops.h:138`) neutralise la L1 liée quand `CALYPSO_DSP_EXTERN` est posé. Le tap DCCH (`calypso_dcch_tap.c`), qui ne dépend d'aucune L1, remplace alors `calypso_l1ctl_tap.c` pour publier le canal dédié.

0.2.1.1.3 Le pont DSP externe

C'est le chemin **réellement utilisé** sur le banc : la mask-ROM tourne dans `c54x_exe`, hors de QEMU, et `calypso_trx.c` lui parle par [calypso_dsp_pont.h](#).

- `CALYPSO_DSP_EXTERN=1` (ou un chemin de socket) désactive la L1 liée (`calypso_l1_disable`) et branche QEMU sur `c54x_exe --arm`, à **lancer d'abord** — sinon `shm_open(...)` ... lancez d'abord `c54x_exe --arm` et `exit(1)`.
- API RAM partagée par `/dev/shm/calypso_api_ram` (16 Ko), synchronisation trame par trame sur `/tmp/calypso_dsp.sock` (`SOCK_SEQPACKET`).
- Messages : `HELLO/HELLO_OK` (magic 'C54X'), `RESET` (écriture de `DL_STATUS`), `TICK` (a = fn, b = page), `DONE` (drapeaux `IRQ/IDLE/INIT`), `GO`, `DCCH`.
- `TICK.b` bit 16 : `IRQ` trame DSP armée par l'ARM — `TPU_CTRL_DSP_EN` est à usage unique (36efec9, 2026-09-20) ; sans lui le DSP ne reçoit pas d'interruption trame.
- `TICK.b` bit 17 : `TICK` en deux phases (255cdd6, 2026-09-21) — `ISR` de la ROM, `DONE|PONT_DONE_PHASE_A`, `IRQ` trame ARM et `l1_sync()`, puis `PONT_GO` et seulement alors le burst. Avant, la page R du burst N écrasait le burst N-2 sous l'ARM (`EMPTY / BURST ID n!=m`).
- `PONT_DCCH` porte TN, genre et sous-voie du canal dédié — `SDCCH/4` (0), `SDCCH/8` (1) ou libéré (0xFF) seulement. Il est appris par `calypso_dcch_tap.c`, qui renifle les `L1CTL_DATA_CONF/DATA_IND` du flux sercomm et publie `/dev/shm/calypso_dcch_cfg` ; ce fichier, lui, annonce aussi le `TCH` (genre 2 = `TCH/F`, 3 = `TCH/H`, 2026-09-23), mais le `TCH` n'est pas relayé par `PONT_DCCH` : la bascule du BSP sur le `TCH` suit la tâche du firmware côté `c54x_exe`.

Les sources de `l1-dsp/` se compilent aussi hors QEMU grâce à [contrib/hors-qemu/cales-qemu.c](#) : équivalents POSIX des quelques symboles QEMU dont le DSP dépend, plus les en-têtes de [contrib/hors-qemu/doublures/](#). C'est ainsi que `c54x_exe` les construit, sans les recopier (`QOSMO ?= /opt/GSM/qosmo` dans son `Makefile`).

Variables d'environnement côté QEMU

variable (sur <code>qemu-system-arm</code>)	effet
<code>CALYPSO_DSP_EXTERN</code>	1 ou chemin de socket : pont vers <code>c54x_exe</code>
<code>CALYPSO_PONT_LOCKSTEP=1</code>	QEMU attend <code>DONE</code> à chaque trame — indispensable, le C54x émulé est plus lent que le temps réel
<code>CALYPSO_PONT_ARM_FIRST=0</code>	ancien ordre, sans les deux phases
<code>CALYPSO_PONT_RETRY_DIV</code>	relance vers le DSP toutes les trame/N (défaut 16, jugé sans effet dans <code>calypso_trx.c</code>) ; <code>c54x_exe/run.sh</code> pose 64 depuis le 2026-09-23 (latence de la phase A et du <code>GO</code> , mesurée par [chrono] en <code>TCH</code>)

variable (sur <code>qemu-system-arm</code>)	effet
<code>CALYPSO_CPU_KICK_NS</code>	période du kick CPU, réglage de mesure (défaut 5 ms, mesuré sans effet)
<code>CALYPSO_PACER_RATTRAPAGE</code>	trames de retard rattrapées par le pacer TDMA (20 par défaut) ; 0 = ancienne grille stricte, qui sautait des trames

`CALYPSO_BSP_STREAM`, `CALYPSO_A5`, `CALYPSO_SONDES`, `CALYPSO_C54X_OVM`, `CALYPSO_MVKD_DMAD_AVANT`, `CALYPSO_HACK_SOFT_SCALE` sont lus par le code de `l1-dsp/` : avec le DSP externe, ils se posent sur `c54x.exe` (le processus du DSP), pas sur QEMU, où ils sont sans effet.

0.2.1.1.4 Vérifier

```
mkdir build && cd build
../configure --target-list=arm-softmmu --disable-werror
ninja qemu-system-arm
./qemu-system-arm -M help | grep calypso
```

La machine **nomme la L1 réellement liée** — c’est la façon la plus courte de savoir ce qu’on a construit :

configure	-M help affiche
(rien)	couche 1 : aucune
<code>--enable-l1-grgsm</code>	couche 1 : grgsm
<code>--enable-l1-dsp</code>	couche 1 : c54x

Sans L1, le firmware boote, balaie et ne trouve jamais de cellule : le modem ne reçoit rien. Ce n’est pas une panne, c’est ce qui rend la plateforme — boot, IRQ, TDMA, UART, SIM — testable sans traîner gr-gsm ni le C54x.

Les deux options sont exclusives, et **meson** le refuse explicitement plutôt que de laisser le second `calypso_l1_register()` écraser le premier en silence :

```
$ ../configure --enable-l1-grgsm --enable-l1-dsp
ERROR: calypso: --enable-l1-grgsm et --enable-l1-dsp sont exclusives.
```

0.2.1.1.5 État

plateforme (SoC, TPU, TSP, SIM, ULPD, UART, timers, SPI/I ² C, INTH, TRF6151, <code>calypso_l1_dispatch.c</code> , <code>calypso_dcch_tap.c</code>)	construit et boote seule ; l'INTH compte la fin de service (<code>IRQ_CTRL</code> bit 0 après <code>IRQ_NUM=4</code>) pour le TICK en deux phases
couture CalypsoL10ps <code>--enable-l1-grgsm</code> <code>--enable-l1-dsp</code> (C54x dans QEMU)	9 points d'appel, vérifiée avec la L1 gr-gsm les ~28 600 lignes de <code>l1-dsp/</code> compilent et se lient ; la vtable reste vide (seul <code>.name = "c54x"</code>)
DSP externe (<code>CALYPSO_DSP_EXTERN</code>)	mask-ROM dans <code>c54x_exe</code> , couplée à l'ARM par le pont ; SCH (BSIC/FN), BCCH SI1-4 et LU sur SDCCH atteints sur le banc ; appels TCH/F de bout en bout, parole décodée par la ROM (runs du 2026-09-23 20:22 et 20:32) ; toutes les trames de parole marquées <code>B_BFI</code> , un LOS en appel à 20:32 — voir ci-dessous
devices de la carte E88	présents, compilent, pas branchés — voir ci-dessous

Neuf devices (`calypso_iota.c`, `calypso_rf3166.c`, `calypso_asm4532.c`, `calypso_xio.c`, `calypso_debug.c`, `calypso_invariants.c`, `fw_console.c`, `sercomm_gate.c`, `l1ctl_sock.c`) sont construits sous `--enable-l1-dsp` seulement, parce que leur unique point d'entrée est `calypso_pcb_init()`, qui vit dans `l1-dsp/calypso_full_pcb.c` — désormais dans ce dépôt, avec `calypso_rif.c`, `calypso_rhea_dma.c`, `calypso_tint0.c` et `calypso_twl3025.c`, construit sous `--enable-l1-dsp` — lequel prend toujours un `C54xState*` (passé en `void *`) et n'est toujours appelé par personne (rien dans `calypso_soc_realize()`). Les activer donnerait neuf fichiers de code mort — mesuré, aucun appelant hors de l'îlot fermé que forment `calypso_asm4532.c` et `calypso_rf3166.c`, qui ne s'appellent que l'un l'autre.

L'ordre pour les débloquent est dans `hw/arm/calypso/meson.build` — dont le bloc « RES-TENT DANS qosmo-dsp » est périmé depuis que ces cinq fichiers sont dans `l1-dsp/`.

Cœur C54x : correctifs récents

- **2026-09-18/19** — `calypso_c54x.c` découpé en `c54x_exec`, `c54x_decode`, `c54x_mem`, `c54x_irq`, `c54x_probes`.
- **2026-09-20** — ROL/ROR sur 32 bits (SPRU172C, c'était une rotation 40 bits), tous les modes Lmem par `resolve_lmem`, CPL respecté en adressage direct : le SCH décode (15c0d45). `getenv()` mémorisé sur les chemins chauds (6,7 → 3,7 ms/trame, d68baaf) ; chemin rapide `c54x_rapide` et file BSP de 8192 bursts (54d8320) ; modulation GMSK partagée `calypso_gmsk.c` (d8e5fef).

- **2026-09-21** — OVA/OVB et saturation OVM (CALYPSO_C54X_OVM=0 pour revenir), MPYR/MACR/MASR, 0xA1xx = ADD Xmem,Ymem (pas SQDST), **ARx(1k)% circulaire, retenue SFTA au bit 40-SHIFT : le BCCH décode, SI1-4 (4771f91).
- **2026-09-23** — sondes coupées par défaut (CALYPSO_SONDES=1 ou CALYPSO_DEBUG pour les rallumer) ; RPT *(1k) avance de 1 + lk_used (crash TCH SP-CORRUPT) ; MVKD/MVDK longs lisent lk avant dmad (SACCH du TCH, pointeur 0x3d89 ; CALYPSO_MVKD_DMAD_AVANT=1 = ancien ordre) ; coprocesseur A5 sur les ports XIO 0x2800..0x2818 ([calypso_a5.c](#), osmo_a5()).

Banc DSP : runs du 2026-09-23 20:22 et 20:32

Montage pont DSP (c54x_exe), mobile DSP = MS 1 (MSISDN 100101). Journaux : osmo-*.log, archives du pont 20260923-202448 et 20260923-203413.

Constaté :

- **Parole TCH/F décodée par la ROM.** Appel MO vers l'écho 600 à 20:22 (ACTIVE 20:22:55, DISCONNECT 20:23:27, ~32 s ; TCH dl=1607 ul=1601 trames), appel MT 100102 → 100101 (ACTIVE 20:24:28, release normal 20:24:32), appel MO à 20:33:39 → DISCONNECT 20:34:06 au run suivant. La ROM TI fait démodulation, égalisation, désentrelacement et Viterbi TCH/F descendant ; codec (GAPK FR) et codage canal montant tournent sur l'hôte. Audible dans les deux sens.
- **Ordre MVKD/MVDK / SACCH 0x3d89** : à 20:22, les deux appels vont jusqu'à la libération normale sans LOS (plus de LOS après ~20 s). Pas vrai du premier appel de 20:32, voir plus bas.
- **Coprocesseur A5** : journal [a5] actif sur les deux runs (Kc lu sur les ports XIO, NOUVEAU Kc à chaque changement) ; chiffrement descendant confirmé par la BTS 5 fois à 20:22 (LU, appel 600, SMS MT, SMS MO, appel MT : chaque établissement) et 3 fois à 20:32.
- **Temps réel** : 29 513 trames à 20:22 (28 441 à 20:32), un seul tick sauté par run, au boot (fn=0). Marge : en TCH, [chrono] à 20:22 (fn 5997-11997) A 0,32-0,34 + go 0,39-0,49 + B 0,15-0,16 + après DONE 3,37-3,68 ms ; à 20:32 A 0,29-0,33 + go 0,40-0,50 + B 0,11-0,16 + après DONE 3,33-3,59 ms. Soit ~4,2-4,6 ms de travail DSP pour 4,62 ms de trame ; l'attente de QEMU tombe à 0,04-0,45 ms. Hors TCH ~0,6-2 ms. Tenu, mais sans réserve : c'est aussi une alerte.

Anomalies ouvertes, par ordre d'importance :

1. **B_BFI sur toutes les trames de parole.** Sonde [a_dd] à 20:32 : vues=2200 bfi=2200. Sur les 42 lignes échantillonnées : 19 × c214 err=0, toutes parmi les 20 premières après la bascule (la 18e est un 8084 err=58), puis 17 × c204 err=15..80 et 5 × 80c4 err=81..93 (err = a_dd_0[2], num_biterr). Premier appel (fn 5839-9306, celui du LOS) : err 58 à 93 hors c214 ; appel de 20:33:39 (fn 21248-26881) : c204 err=15..38. Ce n'est donc pas « bits bons, BFI faux » : la ROM mesure ~3 à 20 % d'erreurs sur 456 bits codés et le FR reste intelligible (le firmware ne remonte pas le BFI, GAPK décode tel quel). Origine

à trancher : signal (BSP, IQ, égalisation) ou cœur C54x (Viterbi, recodage qui compte les erreurs). Comparaison bit à bit avec les trames BTS à faire. Le **ko=376** de la sonde compte **B_FIRE1**, sans sens défini sur la parole.

2. **LOS en appel à 20:32.** Appel MO ACTIVE 20:32:30 ; le compteur ACCH descend de 31 à 0 sans remonter (aucun SACCH descendant accepté) et le mobile jette ~400 blocs en 15 s (« Dropping frame with 54.111 bit errors », **mobile.log** de l'archive 20260923-203413) ; « LOS during dedicated mode » à 20:32:45. La tentative suivante (20:32:57) reste sans réponse jusqu'à 20:33:13, celle de 20:33:26 est relâchée aussitôt ; l'appel de 20:33:36 va au bout. Non reproduit à 20:22.
3. **SACCH descendant du SDCCH/8** : ~la moitié des blocs jetés (7 entre 20:22:42 et 20:22:47 sur le LU). Suspect : la table 45.002 du BSP pour le SACCH/8 sur 102 trames.

Pas une anomalie : les échecs CRC du moniteur TCH descendant du pont tant que le RTP ne coule pas (décodage du pont, indépendant du DSP) ; le **ko** de [**a_dd**] ; le tick sauté à **fn=0**.

Reste à valider : rattrapage du pacer pour la parole (**CALYPSO_PACER_RATTRAPAGE**), rien dans les journaux de ces runs.

0.2.1.1.6 Injecteur de bruit (sens descendant)

Inactif par défaut. [tools/injecteur_bruit.py](#) est un proxy UDP transparent entre le pont et la couche 1 qui dégrade le descendant ; le montant est relayé tel quel. Le module [run_modules/38-bruit.sh](#) le lance (avant 40-qemu) quand **BRUIT_MODE** est posé ; **c54x_exe/run.sh** fait de même pour le DSP externe (étape b).

variable	effet
BRUIT_MODE	ber (bits inversés), souple (AWGN + décision dure), iq (I/Q GMSK + AWGN, expérimental , DSP seulement, refusé sous CALYPSO_BSP_STREAM=1), relais (rien)
BRUIT_BER , BRUIT_RAFALES	taux d'inversion 0..1 ; longueur moyenne des rafales en bits (Gilbert-Elliott)
BRUIT_SNR_DB	SNR en dB, $P/(2^2)$ par échantillon complexe ; sans BRUIT_MODE : exporte CALYPSO_BSP_SNR_DB , l'AWGN du cœur C54x (11-dsp/calypso_c54x.c , c54x_bsp_load), sans effet sur la L1 gr-gsm
BRUIT_PERTE , BRUIT_TN , BRUIT_GRAINE	effacements (dsp : bits aléatoires ; gr-gsm : bloc non transmis), intervalles visés, graine (sans mode : CALYPSO_BSP_BRUIT_GRAINE)
BRUIT_STATS , BRUIT_OPTS , BRUIT_PY	période des statistiques (10 s), options en plus, chemin de l'injecteur

variable	effet
BRUIT_CIBLE, BRUIT_PORT_DSP, BRUIT_PORT_GRGSM	L1 du QEMU (auto : build/meson-info) ; ports déplacés (16702, 14730)

Avec la L1 gr-gsm, QEMU écoute GSMTAP sur 4730 **en dur** : c'est le pont qui vise PONT_GSMTAP_PORT (osmo-operator pont/config.py), que pont/pont.py met lui-même à 14730 quand BRUIT_MODE vaut **ber** ou **relais** ; le flux est fait de blocs L2 déjà décodés, d'où ces deux modes seulement. Avec la L1 DSP, le module exporte CALYPSO_BSP_PORT=16702 pour QEMU et l'injecteur prend 6702. En montage DSP de `start-native.sh` (qemu retiré du plan), le module se saute. Journal : `$LOG_DIR/bruit.log`. Détail : `osmo-operator/wiki/Injecteur-bruit.md`.

0.2.1.1.7 Documentation liée

- [hw/arm/calypso/doc/SONDES.md](#) — les sondes du cœur C54x (2026-09-18).
- [tools/rapport-run.sh](#) — rapport des journaux qemu/osmocon/mobile/bts/pont d'un run ; LOG_DIR est lu dans l'environ du QEMU vivant.

0.2.1.1.8 Licence

Code dérivé de QEMU 9.2.4 : GPL-2.0-or-later (en-têtes SPDX, COPYING). Le dépôt porte aussi, depuis le 2026-09-23, un LICENSE GPLv3 (« qosmo Copyright (C) 2026 Bastien Baranoff »).

0.2.2 /opt/GSM/qosmo-overlay/meson_options.txt

20174 octets, 393 lignes → 393 lignes

```
# These options do not correspond to a --enable/--disable-* option
# on the configure script command line. If you add more, list them in
# scripts/meson-buildoptions.py's SKIP_OPTIONS constant too.

option('qemu_suffix', type : 'string', value: 'qemu',
       description: 'Suffix for QEMU data/modules/config directories (can be empty)')
option('docdir', type : 'string', value : 'share/doc',
       description: 'Base directory for documentation installation (can be empty)')
option('qemu_firmwarepath', type : 'array', value : ['share/qemu-firmware'],
       description: 'search PATH for firmware files')
option('pkgversion', type : 'string', value : '',
       description: 'use specified string as sub-version of the package')
option('smbd', type : 'string', value : '',
```

```

        description: 'Path to smbd for slirp networking')
option('iasl', type : 'string', value : '',
        description: 'Path to ACPI disassembler')
option('tls_priority', type : 'string', value : 'NORMAL',
        description: 'Default TLS protocol/cipher priority string')
option('default_devices', type : 'boolean', value : true,
        description: 'Include a default selection of devices in emulators')
option('audio_drv_list', type: 'array', value: ['default'],
        choices: ['alsa', 'coreaudio', 'default', 'dsound', 'jack', 'oss', 'pa', 'pipewire',
        description: 'Set audio driver list')
option('block_drv_rw_whitelist', type : 'string', value : '',
        description: 'set block driver read-write whitelist (by default affects only QEMU, not
option('block_drv_ro_whitelist', type : 'string', value : '',
        description: 'set block driver read-only whitelist (by default affects only QEMU, not
option('interp_prefix', type : 'string', value : '/usr/gnemul/qemu-%M',
        description: 'where to find shared libraries etc., use %M for cpu name')
option('rtsig_map', type : 'string', value : 'NULL',
        description: 'default value of QEMU_RTSIG_MAP')
option('fuzzing_engine', type : 'string', value : '',
        description: 'fuzzing engine library for OSS-Fuzz')
option('trace_file', type: 'string', value: 'trace',
        description: 'Trace file prefix for simple backend')
option('coroutine_backend', type: 'combo',
        choices: ['ucontext', 'sigaltstack', 'windows', 'auto'],
        value: 'auto', description: 'coroutine backend to use')

# Everything else can be set via --enable/--disable-* option
# on the configure script command line. After adding an option
# here make sure to run "make update-buildoptions".

option('docs', type : 'feature', value : 'auto',
        description: 'Documentations build support')
option('fuzzing', type : 'boolean', value: false,
        description: 'build fuzzing targets')
option('gettext', type : 'feature', value : 'auto',
        description: 'Localization of the GTK+ user interface')
option('modules', type : 'feature', value : 'disabled',
        description: 'modules support (non Windows)')
option('module_upgrades', type : 'boolean', value : false,
        description: 'try to load modules from alternate paths for upgrades')
option('install_blobs', type : 'boolean', value : true,
        description: 'install provided firmware blobs')

```

```

option('sparse', type : 'feature', value : 'auto',
      description: 'sparse checker')
option('guest_agent', type : 'feature', value : 'auto',
      description: 'Build QEMU Guest Agent')
option('guest_agent_msi', type : 'feature', value : 'auto',
      description: 'Build MSI package for the QEMU Guest Agent')
option('tools', type : 'feature', value : 'auto',
      description: 'build support utilities that come with QEMU')
option('qga_vss', type : 'feature', value: 'auto',
      description: 'build QGA VSS support (broken with MinGW)')

option('malloc_trim', type : 'feature', value : 'auto',
      description: 'enable libc malloc_trim() for memory optimization')
option('malloc', type : 'combo', choices : ['system', 'tcmalloc', 'jemalloc'],
      value: 'system', description: 'choose memory allocator to use')

option('kvm', type: 'feature', value: 'auto',
      description: 'KVM acceleration support')
option('whpx', type: 'feature', value: 'auto',
      description: 'WHPX acceleration support')
option('hvf', type: 'feature', value: 'auto',
      description: 'HVF acceleration support')
option('nvm', type: 'feature', value: 'auto',
      description: 'NVMM acceleration support')
option('xen', type: 'feature', value: 'auto',
      description: 'Xen backend support')
option('xen_pci_passthrough', type: 'feature', value: 'auto',
      description: 'Xen PCI passthrough support')
option('tcg', type: 'feature', value: 'enabled',
      description: 'TCG support')
option('plugins', type: 'boolean', value: false,
      description: 'TCG plugins via shared library loading')
option('debug_tcg', type: 'boolean', value: false,
      description: 'TCG debugging')
option('debug_remap', type: 'boolean', value: false,
      description: 'syscall buffer debugging support')
option('tcg_interpreter', type: 'boolean', value: false,
      description: 'TCG with bytecode interpreter (slow)')
option('safe_stack', type: 'boolean', value: false,
      description: 'SafeStack Stack Smash Protection (requires clang/llvm and coroutine back')
option('asan', type: 'boolean', value: false,
      description: 'enable address sanitizer')

```

```

option('ubsan', type: 'boolean', value: false,
      description: 'enable undefined behaviour sanitizer')
option('tsan', type: 'boolean', value: false,
      description: 'enable thread sanitizer')
option('stack_protector', type: 'feature', value: 'auto',
      description: 'compiler-provided stack protection')
option('cfi', type: 'boolean', value: false,
      description: 'Control-Flow Integrity (CFI)')
option('cfi_debug', type: 'boolean', value: false,
      description: 'Verbose errors in case of CFI violation')
option('multiprocess', type: 'feature', value: 'auto',
      description: 'Out of process device emulation support')
option('relocatable', type: 'boolean', value: true,
      description: 'toggle relocatable install')
option('vfio_user_server', type: 'feature', value: 'disabled',
      description: 'vfio-user server support')
option('dbus_display', type: 'feature', value: 'auto',
      description: '-display dbus support')
option('tpm', type: 'feature', value: 'auto',
      description: 'TPM support')

# Do not enable it by default even for Mingw32, because it doesn't
# work on Wine.
option('membarrier', type: 'feature', value: 'disabled',
      description: 'membarrier system call (for Linux 4.14+ or Windows')

option('avx2', type: 'feature', value: 'auto',
      description: 'AVX2 optimizations')
option('avx512bw', type: 'feature', value: 'auto',
      description: 'AVX512BW optimizations')
option('keyring', type: 'feature', value: 'auto',
      description: 'Linux keyring support')
option('libkeyutils', type: 'feature', value: 'auto',
      description: 'Linux keyutils support')

option('af_xdp', type: 'feature', value: 'auto',
      description: 'AF_XDP network backend support')
option('attr', type: 'feature', value: 'auto',
      description: 'attr/xattr support')
option('auth_pam', type: 'feature', value: 'auto',
      description: 'PAM access control')
option('brlapi', type: 'feature', value: 'auto',

```

```

        description: 'brlapi character device driver')
option('bzip2', type : 'feature', value : 'auto',
        description: 'bzip2 support for DMG images')
option('cap_ng', type : 'feature', value : 'auto',
        description: 'cap_ng support')
option('blkio', type : 'feature', value : 'auto',
        description: 'libblkio block device driver')
option('bpf', type : 'feature', value : 'auto',
        description: 'eBPF support')
option('cocoa', type : 'feature', value : 'auto',
        description: 'Cocoa user interface (macOS only)')
option('curl', type : 'feature', value : 'auto',
        description: 'CURL block device driver')
option('gio', type : 'feature', value : 'auto',
        description: 'use libgio for D-Bus support')
option('glusterfs', type : 'feature', value : 'auto',
        description: 'Glusterfs block device driver')
option('hv_balloon', type : 'feature', value : 'auto',
        description: 'hv-balloon driver (requires Glib 2.68+ GTree API)')
option('libdw', type : 'feature', value : 'auto',
        description: 'debuginfo support')
option('libiscsi', type : 'feature', value : 'auto',
        description: 'libiscsi userspace initiator')
option('libnfs', type : 'feature', value : 'auto',
        description: 'libnfs block device driver')
option('mpath', type : 'feature', value : 'auto',
        description: 'Multipath persistent reservation passthrough')
option('numa', type : 'feature', value : 'auto',
        description: 'libnuma support')
option('iconv', type : 'feature', value : 'auto',
        description: 'Font glyph conversion support')
option('curses', type : 'feature', value : 'auto',
        description: 'curses UI')
option('libcbor', type : 'feature', value : 'auto',
        description: 'libcbor support')
option('gnutls', type : 'feature', value : 'auto',
        description: 'GNUTLS cryptography support')
option('nettle', type : 'feature', value : 'auto',
        description: 'nettle cryptography support')
option('gcrypt', type : 'feature', value : 'auto',
        description: 'libgcrypt cryptography support')
option('crypto_afalg', type : 'feature', value : 'disabled',

```

```

        description: 'Linux AF_ALG crypto backend driver')
option('libdaxctl', type : 'feature', value : 'auto',
        description: 'libdaxctl support')
option('libpmem', type : 'feature', value : 'auto',
        description: 'libpmem support')
option('libssh', type : 'feature', value : 'auto',
        description: 'ssh block device support')
option('libudev', type : 'feature', value : 'auto',
        description: 'Use libudev to enumerate host devices')
option('libusb', type : 'feature', value : 'auto',
        description: 'libusb support for USB passthrough')
option('linux_aio', type : 'feature', value : 'auto',
        description: 'Linux AIO support')
option('linux_io_uring', type : 'feature', value : 'auto',
        description: 'Linux io_uring support')
option('lzfse', type : 'feature', value : 'auto',
        description: 'lzfse support for DMG images')
option('lzo', type : 'feature', value : 'auto',
        description: 'lzo compression support')
option('rbd', type : 'feature', value : 'auto',
        description: 'Ceph block device driver')
option('opengl', type : 'feature', value : 'auto',
        description: 'OpenGL support')
option('rdma', type : 'feature', value : 'auto',
        description: 'Enable RDMA-based migration')
option('gtk', type : 'feature', value : 'auto',
        description: 'GTK+ user interface')
option('sdl', type : 'feature', value : 'auto',
        description: 'SDL user interface')
option('sdl_image', type : 'feature', value : 'auto',
        description: 'SDL Image support for icons')
option('seccomp', type : 'feature', value : 'auto',
        description: 'seccomp support')
option('smartcard', type : 'feature', value : 'auto',
        description: 'CA smartcard emulation support')
option('snappy', type : 'feature', value : 'auto',
        description: 'snappy compression support')
option('spice', type : 'feature', value : 'auto',
        description: 'Spice server support')
option('spice_protocol', type : 'feature', value : 'auto',
        description: 'Spice protocol support')
option('u2f', type : 'feature', value : 'auto',

```

```

        description: 'U2F emulation support')
option('canokey', type : 'feature', value : 'auto',
        description: 'CanoKey support')
option('usb_redir', type : 'feature', value : 'auto',
        description: 'libusbredir support')
option('l2tpv3', type : 'feature', value : 'auto',
        description: 'l2tpv3 network backend support')
option('netmap', type : 'feature', value : 'auto',
        description: 'netmap network backend support')
option('pixman', type : 'feature', value : 'auto',
        description: 'pixman support')
option('slirp', type: 'feature', value: 'auto',
        description: 'libslirp user mode network backend support')
option('vde', type : 'feature', value : 'auto',
        description: 'vde network backend support')
option('vmnet', type : 'feature', value : 'auto',
        description: 'vmnet.framework network backend support')
option('virglrenderer', type : 'feature', value : 'auto',
        description: 'virgl rendering support')
option('rutabaga_gfx', type : 'feature', value : 'auto',
        description: 'rutabaga_gfx support')
option('png', type : 'feature', value : 'auto',
        description: 'PNG support with libpng')
option('vnc', type : 'feature', value : 'auto',
        description: 'VNC server')
option('vnc_jpeg', type : 'feature', value : 'auto',
        description: 'JPEG lossy compression for VNC server')
option('vnc_sasl', type : 'feature', value : 'auto',
        description: 'SASL authentication for VNC server')
option('vte', type : 'feature', value : 'auto',
        description: 'vte support for the gtk UI')

# GTK Clipboard implementation is disabled by default, since it may cause hangs
# of the guest VCPUs. See gitlab issue 1150:
# https://gitlab.com/qemu-project/qemu/-/issues/1150

option('gtk_clipboard', type: 'feature', value : 'disabled',
        description: 'clipboard support for the gtk UI (EXPERIMENTAL, MAY HANG)')
option('xkbcommon', type : 'feature', value : 'auto',
        description: 'xkbcommon support')
option('zstd', type : 'feature', value : 'auto',
        description: 'zstd compression support')

```

```

option('qpl', type : 'feature', value : 'auto',
      description: 'Query Processing Library support')
option('uadk', type : 'feature', value : 'auto',
      description: 'UADK Library support')
option('qatzip', type: 'feature', value: 'auto',
      description: 'QATzip compression support')
option('fuse', type: 'feature', value: 'auto',
      description: 'FUSE block device export')
option('fuse_lseek', type : 'feature', value : 'auto',
      description: 'SEEK_HOLE/SEEK_DATA support for FUSE exports')

option('trace_backends', type: 'array', value: ['log'],
      choices: ['dtrace', 'ftrace', 'log', 'nop', 'simple', 'syslog', 'ust'],
      description: 'Set available tracing backends')

option('alsa', type: 'feature', value: 'auto',
      description: 'ALSA sound support')
option('coreaudio', type: 'feature', value: 'auto',
      description: 'CoreAudio sound support')
option('dsound', type: 'feature', value: 'auto',
      description: 'DirectSound sound support')
option('jack', type: 'feature', value: 'auto',
      description: 'JACK sound support')
option('oss', type: 'feature', value: 'auto',
      description: 'OSS sound support')
option('pa', type: 'feature', value: 'auto',
      description: 'PulseAudio sound support')
option('pipewire', type: 'feature', value: 'auto',
      description: 'PipeWire sound support')
option('sndio', type: 'feature', value: 'auto',
      description: 'sndio sound support')

option('vhost_kernel', type: 'feature', value: 'auto',
      description: 'vhost kernel backend support')
option('vhost_net', type: 'feature', value: 'auto',
      description: 'vhost-net kernel acceleration support')
option('vhost_user', type: 'feature', value: 'auto',
      description: 'vhost-user backend support')
option('vhost_crypto', type: 'feature', value: 'auto',
      description: 'vhost-user crypto backend support')
option('vhost_vdpa', type: 'feature', value: 'auto',
      description: 'vhost-vdpa kernel backend support')

```

```

option('vhost_user_blk_server', type: 'feature', value: 'auto',
      description: 'build vhost-user-blk server')
option('virtfs', type: 'feature', value: 'auto',
      description: 'virtio-9p support')
option('libvduse', type: 'feature', value: 'auto',
      description: 'build VDUSE Library')
option('vduse_blk_export', type: 'feature', value: 'auto',
      description: 'VDUSE block export support')

option('capstone', type: 'feature', value: 'auto',
      description: 'Whether and how to find the capstone library')
option('fdt', type: 'combo', value: 'auto',
      choices: ['disabled', 'enabled', 'auto', 'system', 'internal'],
      description: 'Whether and how to find the libfdt library')

option('selinux', type: 'feature', value: 'auto',
      description: 'SELinux support in qemu-nbd')
option('replication', type: 'feature', value: 'auto',
      description: 'replication support')
option('colo_proxy', type: 'feature', value: 'auto',
      description: 'colo-proxy support')
option('bochs', type: 'feature', value: 'auto',
      description: 'bochs image format support')
option('cloop', type: 'feature', value: 'auto',
      description: 'cloop image format support')
option('dmg', type: 'feature', value: 'auto',
      description: 'dmg image format support')
option('qcow1', type: 'feature', value: 'auto',
      description: 'qcow1 image format support')
option('vdi', type: 'feature', value: 'auto',
      description: 'vdi image format support')
option('vhdx', type: 'feature', value: 'auto',
      description: 'vhdx image format support')
option('vmdk', type: 'feature', value: 'auto',
      description: 'vmdk image format support')
option('vpc', type: 'feature', value: 'auto',
      description: 'vpc image format support')
option('vvfat', type: 'feature', value: 'auto',
      description: 'vvfat image format support')
option('qed', type: 'feature', value: 'auto',
      description: 'qed image format support')
option('parallels', type: 'feature', value: 'auto',

```

```

        description: 'parallels image format support')
option('block_drv_whitelist_in_tools', type: 'boolean', value: false,
        description: 'use block whitelist also in tools instead of only QEMU')
option('rng_none', type: 'boolean', value: false,
        description: 'dummy RNG, avoid using /dev/(u)random and getrandom()')
option('coroutine_pool', type: 'boolean', value: true,
        description: 'coroutine freelist (better performance)')
option('debug_graph_lock', type: 'boolean', value: false,
        description: 'graph lock debugging support')
option('debug_mutex', type: 'boolean', value: false,
        description: 'mutex debugging support')
option('debug_stack_usage', type: 'boolean', value: false,
        description: 'measure coroutine stack usage')
option('qom_cast_debug', type: 'boolean', value: true,
        description: 'cast debugging support')
option('slirp_smbd', type: 'feature', value: 'auto',
        description: 'use smbd (at path --smbd=*) in slirp networking')

option('qemu_ga_manufacturer', type: 'string', value: 'QEMU',
        description: '"manufacturer" name for qemu-ga registry entries')
option('qemu_ga_distro', type: 'string', value: 'Linux',
        description: 'second path element in qemu-ga registry entries')
option('qemu_ga_version', type: 'string', value: '',
        description: 'version number for qemu-ga installer')

option('hexagon_idef_parser', type: 'boolean', value: true,
        description: 'use idef-parser to automatically generate TCG code for the Hexagon front-end')

option('x86_version', type: 'combo', choices: ['0', '1', '2', '3', '4'], value: '1',
        description: 'tweak required x86_64 architecture version beyond compiler default')

option('rust', type: 'feature', value: 'disabled',
        description: 'Rust support')
option('strict_rust_lints', type: 'boolean', value: false,
        description: 'Enable stricter set of Rust warnings')

# [2026-09-16] Le choix de la couche 1 de la machine Calypso. Voir
# include/hw/arm/calypso/calypso_l1_ops.h. Les deux sont exclusives : deux L1
# liées dans le même binaire, c'est deux appels à calypso_l1_register(), donc
# la dernière qui gagne en silence. meson.build refuse la combinaison.
option('l1_grgsm', type: 'feature', value: 'disabled',
        description: 'Calypso: couche 1 gr-gsm (demodulation cote hôte)')

```

```
option('l1_dsp', type: 'feature', value: 'disabled',
      description: 'Calypso: couche 1 DSP C54x (mask-ROM TI reelle)')
```

0.2.3 /opt/GSM/qosmo-overlay/environnement/README.md

3788 octets, 85 lignes

0.2.3.1 environnement/ — la configuration, par domaine

calypso.env mélangeait 312 variables : chemins d'installation, cadences du modèle, sondes de mesure et contournements, sans distinction. Ce répertoire les sépare **par ce qu'elles font**, pour qu'on sache ce qu'on a le droit de toucher.

Fichier	Domaine	Touchez-y ?
paths.env	où sont les dépendances sur votre machine	oui , c'est le premier à régler
modes.env	les profils prêts à l'emploi (le mode qui campe, le mode natif...)	oui , choisissez-en un
bsp.env	BSP / DMA / DARAM / feed I-Q / horloge TDMA	si vous savez ce que vous faites
dsp.env	cœur c54x : interruptions, IMR, cadence d'exécution	idem
fbsb.env	FB / FBSB / corrélateur / démodulateur	idem
shunt.env	canaux shunt DL/UL, injections	idem
rf.env	TPU / TSP / RF / AFC	idem
debug.env	sondes et traces — sans effet sur l'émulation	oui , librement
fixes.env	le sas CALYPSO_FIXES : correctifs en attente de validation	temporairement
crutches.env	les béquilles — contournements de branches non implémentées	non , sauf pour diagnostiquer

0.2.3.1.1 Trois choses à savoir avant de modifier quoi que ce soit

1. La vérité est le manifeste, pas la ligne de commande.

```
grep "calypso-manifest" /root/qemu.log
```

Certaines variables en reposent d'autres silencieusement : CALYPSO_NATIVE_HELPED=1 allume aussi FB_CORR_ENTRY, FB_ENERGY et FB_IQ_*. Retirer l'une d'elles de la ligne de commande ne la supprime pas — elle revient à son défaut.

2. =0 ne coupe pas tout. Quatre idiomes de gate coexistent :

Idiome	Actif quand	Comment couper
getenv("X") ? 1 : 0	la variable existe , même à 0	unset X
atoi(...) > 0	valeur > 0	X=0
*e == '1'	exactement "1"	toute autre valeur
défaut ON + X_OFF	par défaut	poser X_OFF

3. := contre =. Dans ces fichiers, : "\${VAR:=valeur}" laisse la ligne de commande gagner ; VAR=valeur la **verrouille**. N'utilisez = que pour ce qui ne doit jamais être surchargé.

0.2.3.1.2 Référence complète

Les 312 variables, avec défaut, effet mesuré, mode, idiome, catégorie et dépendances :
../hw/arm/calypso/doc/VARIABLES_ENVIRONNEMENT.md.

0.2.3.1.3 Comment ça se charge

Tout passe par load.env, sourcé par start-clean.sh :

```
set -a ; . ./environnement/load.env ; set +a
```

L'ordre compte. Tous les fichiers utilisent := (« si pas déjà défini »), donc **le premier qui pose une valeur gagne** :

1. la **ligne de commande** — VAR=x ./run.sh gagne toujours ;
2. modes.env — le profil choisi ;
3. les fichiers par domaine — les défauts ;
4. calypso.env — l'ancien monolithe, conservé le temps de la migration. Il n'utilise que :=, donc il ne peut rien écraser : il ne fournit que ce qui n'a pas encore été réparti. À vider progressivement.

0.2.3.1.4 Les fichiers hérités

`calypso.env`, `calypso_hack.env`, `calypso_native*.env`, `calypso_shunt_*.env`, `calypso_wire.env` datent d'avant ce découpage. Ils sont conservés — on ne détruit pas ce qui marche — mais la configuration nouvelle vit dans les fichiers par domaine.

0.2.3.1.5 Ce que contient chaque domaine

Fichier	Variables	dont béquilles
<code>bsp.env</code>	52	10
<code>dsp.env</code>	52	15
<code>fbsb.env</code>	52	25
<code>shunt.env</code>	58	34
<code>armdsp.env</code>	47	16
<code>opcodes.env</code>	50	16

116 béquilles sur 311 variables — plus d'un tiers du projet est du contournement. Chacune est annotée à son site de lecture dans le code (`grep -rn "@BEQUILLE" hw/arm/calypso/`), avec ce qu'elle masque et la condition qui la rendrait inutile.

0.2.4 /opt/GSM/qosmo-overlay/hw/arm/calypso/doc/SONDES.md

7894 octets, 118 lignes

0.2.4.1 Sonder le DSP Calypso emule : la regle d'or

Sur ce projet, la premiere hypothese a tester n'est pas « le DSP emule se trompe » mais « la sonde se trompe ».

Ce n'est pas une precaution de style. Sur les seules journees des 17 et 18 septembre 2026, **cinq** diagnostics spectaculaires se sont reveles etre des defauts de mesure, et chacun a coute des heures de chasse a un bug qui n'existait pas.

symptome « evident »	cause reelle
« aucun etage de la chaine SB ne s'execute » (<code>b219=0 ... 770a=0</code>)	les compteurs n'etaient renseignes que dans le chemin de sondage, mais le bilan les imprimait toujours
« les bits souples sont reduits a ± 1 , la magnitude est annihilee » (piste PM/FRCT/ASM/SXM/OVM)	la sonde lisait <code>0x2a00</code> , le tampon du correlateur FB, au lieu de <code>0x2c72</code>

symptome « evident »	cause reelle
« les magnitudes sont saines et l'argmax est fautif »	comparaison en <code>uint16_t</code> : les « max » 65502/65104 valaient -34/-432 en <code>int16_t</code> , et le rang associe etait du bruit
« le champ max signe contient 0x2f06 »	15 conversions de format pour 13 arguments : toute la fin de ligne glissait d'un cran
« FIRS multiplie par la constante 0xf4e4 au lieu du filtre »	lecture de <code>prog[]</code> BRUT : avec <code>PMST_OVLY</code> arme et le plancher d'alias a 0x0060, une lecture programme dans 0x0060-0x27FF est redirigee vers <code>data[]</code> , ou les coefficients sont bien la

0.2.4.1.1 Les six regles qui en decoulent

1. **Prouver qu'une sonde est active dans le mode ou l'on tourne.** Ne jamais imprimer un compteur renseigne dans une seule branche sans ecrire « non mesure » dans l'autre.
2. **Localiser un tampon par difference de RAM**, en photographiant avant et en diffant apres, jamais par supposition sur une adresse.
3. **Respecter les types.** La memoire du DSP est un tableau de `uint16_t` qui contient des `int16_t` signes. Toute comparaison de magnitude se caste, et pour une grandeur signee on donne le maximum en valeur absolue A COTE du maximum signe.
4. **Passer par la traduction d'adresse du coeur**, pas par les tableaux bruts. `prog[]` et `data[]` ne sont pas ce que le DSP lit : l'alias `OVLY`, le banking `XPC` et les `MMR` s'interposent. Reproduire la traduction dans la sonde, ou appeler la fonction du coeur.
5. **Lire les avertissements du compilateur, pas seulement les erreurs.** Le decalage de la liste d'arguments etait signale par `-Wformat` ; il a survécu parce que la sortie de `make` etait filteree sur `error:`.
6. **Qualifier un succes dans le code, pas en relisant le log.** Sur ce banc un CRC OK n'est un decodage que s'il rend le BSIC INJECTE, un `T3 <= 50` ET une FN egale a la trame. Le compteur imprime les deux nombres : CRC OK: N, dont VRAIES: M.

0.2.4.1.2 La regle du PC : la sonde journalise le PC APRES avancement

Le dispatcheur avance le PC avant que la sonde ne l'observe. **Tous les PC rapportes par les sondes de ce banc sont donc decales d'une instruction**, et les frontieres de region avec eux. Les vrais ecrivains du correlateur de midambule sont 0x84bc et 0x84c5, la ou la trace affiche 0x84bd et 0x84c6 ; de meme 0x7cb8 et non 0x7cb9.

C'est la defaillance de mesure la plus insidieuse du lot, parce qu'elle ne produit **aucune valeur absurde** : tout reste plausible, seulement decale. Les six autres finissaient par sortir un chiffre impossible ; celle-la, non. Verifier le decalage AVANT de nommer une instruction fautive.

0.2.4.1.3 Le temoin de zero : la regle qui aurait attrape les cinq

Definir une zone dont on SAIT qu'elle doit peser zero, et la mesurer a chaque campagne. Sur ce banc c'est la marge : les 21 echantillons avant le burst et les 21 apres ne portent aucune information, donc perturber l'un d'eux ne doit rien changer a la sortie. Mesure : 42 points sur 42 a la valeur plancher, contre 66 a 19534 a l'interieur du burst. Le jour ou un echantillon de marge se met a peser, ce n'est pas le DSP qui a change, c'est la sonde ou le protocole de mesure qui est casse — et on le sait AVANT d'avoir bati une hypothese dessus.

C'est exactement ce temoin qui a revele que les « poids absurdes » du §9.30 etaient un artefact : la perturbation etait appliquee a toutes les trames et modifiait l'historique du run. Chacune des cinq fausses sondes precedentes aurait ete attrapee par un temoin equivalent, et chacune a coute un tour complet.

Corollaire de metrique : distinguer un VRAI zero d'un PLANCHER DE SENSIBILITE. Compter les changements de signe rate tout ce qui ne change que la magnitude, et fait passer une influence faible pour une absence. Mesurer en $\max |\delta|$ de valeur, et verifier que le plancher de la zone temoin est bien separe du plancher de la zone utile.

0.2.4.1.4 Pieges specifiques a ce banc

- `getenv("X")` est VRAI pour `X=0`. Une ligne ecrite `REJEU_SCH_PARTOUT=0 REJEU_SB_FORCE=0` ACTIVAIT les hacks. Utiliser un helper qui traite 0, "", off, non comme faux.
- `REJEU_SCH_PARTOUT=1` **corrompt la verite de reference** : `cellule.c` calcule `t3p = p51 / 10` pour tout `p51`, ce qui n'est juste que sur {1, 11, 21, 31, 41}. La FN ne peut pas etre validee sous ce drapeau.
- Le decalage d'echantillonnage GMSK doit valoir **0,5** (centre du symbole). A 0,0 (frontiere) le voisin suivant pese 0,492 contre 0,454 pour le symbole visee : l'offset devient indetermine et TOUT en aval devient instable.
- La fenetre de la SB fait **380 mots** (190 complexes), pas 296. Suivre la longueur que le DSP programme (`ALGTH`), ne pas la figer.

0.2.4.1.5 Piege 8 (18/09, soir) : le filtre de verbosite de `c54x_exe` AVALE les sondes

`c54x_exe` ne se contente pas de relayer `stderr` : `src/verbosite.c` detourne `stderr` dans un tube et classe chaque ligne par MOT-CLEF du nom de la sonde. Une sonde dont le nom contient WATCH, DUMP, SCAN, PROBE ou MAP est de niveau 4, TRACE, LOOP, CYCLE de niveau 5. **Sans `-vvvv` (ou `-vvvvv`), elles ne s'affichent pas du tout.**

Le symptome est le pire possible : la sonde est compilee, active, elle s'execute — et ne sort RIEN. On conclut « ce code n'est jamais atteint », ce qui est faux. Une sonde de plage posee sur 0x0000-0xffff a ainsi rapporte ZERO ecriture sur tout l'espace memoire.

Regle : avant de conclure d'un silence, relancer avec -vvvvvv. Et pour une sonde ponctuelle dont on veut la sortie inconditionnelle, choisir un nom qui ne contient AUCUN de ces mots-clefs.

0.2.4.1.6 Piege 9 : « le PC vu moins 1 » n'est PAS universel

La regle du PC (plus haut) dit que les sondes de `c54x_mem.c` journalisent le PC APRES avancement, donc que l'ecrivain est `pc-1`. **C'est vrai pour certaines familles et faux pour d'autres.** Mesure : dans la boucle de compaction, la sonde d'ecriture rapporte `ecr=0x7e70`, or le pas-a-pas montre `0x7e70 = ld` (aucune ecriture) et `0x7e71 = sth`, le vrai ecrivain. L'ecart d'un cran change donc de sens selon l'instruction.

Regle : ne jamais nommer une instruction fautive d'apres le PC d'une sonde memoire. Confirmer par un pas-a-pas qui imprime le PC AVANT execution avec l'opcode reel (`CALYPSO_PISTE_LO/HI`, dans `calypso_c54x.c`), et decoder cet opcode contre `doc/opcodes/tic54x-opc.c`, pas de memoire.

0.2.4.1.7 Les sondes ajoutes le 18/09 au soir

env	effet
<code>CALYPSO_WATCH_WR_LO/HI/N</code>	journal chronologique de toute ecriture dans une plage data (idx, valeur, ancienne valeur, PC, insn)
<code>CALYPSO_PISTE_LO/HI/N</code>	pas-a-pas d'une plage de PC : PC AVANT execution, opcode reel, mot suivant (litteral), A/B/T/AR0-7/BRC, plus les mots pointes par AR2..AR5, DP/CPL/SP et les adresses directes candidates
<code>CALYPSO_PISTE_DUMP_LO/HI</code>	dump d'une plage data a chaque passage dans la plage de PC
<code>CALYPSO_T_LO/HI</code>	dans une fenetre d'instructions : chaque changement de T (<code>PISTE-T</code>) et chaque annulation d'un accumulateur non nul (<code>PISTE-NUL</code>)

Toutes exigent -vvvv/-vvvvv (piege 8) et sont inertes tant que les bornes ne sont pas posees.

0.2.5 /opt/GSM/qosmo-overlay/tests/detail.mmd

13556 octets, 188 lignes



0.2.6 /opt/GSM/qosmo-overlay/tests/detail.qmd.mmd

26127 octets, 393 lignes

0.2.7 /opt/GSM/qosmo-overlay/tests/full.mmd

15096 octets, 226 lignes



Syntax error in te
mermaid version 10.2.0-rc.2

0.2.8 /opt/GSM/qosmo-overlay/tests/pipeline.mmd

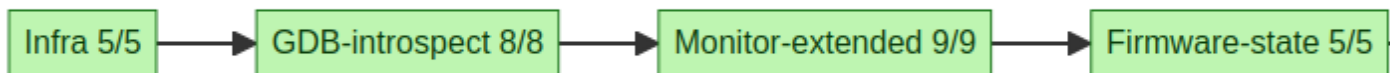
1120 octets, 34 lignes



Syntax error in te
mermaid version 10.2.0-rc.2

0.2.9 /opt/GSM/qosmo-overlay/tests/pipeline.qmd.mmd

1945 octets, 64 lignes



0.2.10 /opt/GSM/qosmo-overlay/tmux_modules/README.md

1811 octets, 39 lignes

0.2.10.1 tmux_modules — une disposition par profil

run_modules/35-tmux.sh ne décrit plus aucune fenêtre : il charge _commun.sh puis le fichier du profil courant, et appelle tmux_layout.

_commun.sh	couleurs, _w / _split / _tail, fenêtres coeur/voix/shell, habillage
calypso.sh	un téléphone émulé (QEMU) + le cœur
hybrid.sh	DEUX téléphones : MS#1 sur QEMU, MS#2 sur fake_trx - un seul cœur
faketrx.sh	pas de QEMU : téléphone logiciel, radio simulée
default.sh	repli (profil « core », ou tout profil sans fichier dédié)

0.2.10.1.1 Ajouter un profil

Créer <profil>.sh définissant trois choses :

TMUX_FENETRE_PREMIERE	nom de la fenêtre créée avec la session
tmux_layout_premiere()	la commande qu'elle exécute
tmux_layout()	toutes les autres fenêtres, via _w / _split

Rien d'autre à modifier : le chargeur trouve le fichier par le nom du profil, et retombe sur default.sh s'il n'existe pas.

0.2.10.1.2 Règles

- **La couleur porte le rôle**, pas la position — voir l'en-tête de _commun.sh. Quatre tail côte à côte sont indiscernables sans elle.
- **Les panes du cœur suivent /var/log/osmocom/osmo-*.log**, les journaux des SERVICES. Suivre \$LOG_DIR/mod/*.log ne montrerait que les traces de démarrage des modules — l'erreur qui a rendu ces fenêtres inutiles longtemps.
- **tail -F**, jamais -f : le journal peut ne pas exister encore, et le garde-fou de 40-qemu tronque qemu.log au-delà du plafond.
- **|| sleep infinity** en repli : sans lui, un journal absent ferait mourir le pane et refermerait la fenêtre en emportant les autres.
- Une disposition ne lance **aucun service**. tmux ne sert qu'à REGARDER ; sa session est cosmétique et son absence n'empêche rien de tourner.

0.3 Fichiers

0.3.1 /opt/GSM/qosmo-overlay/install.sh

29149 octets, 545 lignes → 545 lignes

```
#!/bin/bash
# =====
# install.sh - construit et installe qosmo (QEMU, machine Calypso)
# =====
#
# Autonome : ce depot et quelques paquets apt suffisent, osmo-operator n est
# pas necessaire. C est aussi CE script qu appellent le Dockerfile
# d osmo-operator (stage qemu), son Dockerfile.run, start-docker.sh et l installation
# native (install_modules/45-calypso.sh) : une seule liste de commandes.
#
# ./install.sh          tout, dans l ordre (voir --list)
# ./install.sh --check  prerequis et etat, NE MODIFIE RIEN
# ./install.sh --list   les etapes et les valeurs retenues
# ./install.sh --print-deps les paquets apt, sur une ligne
# sudo ./install.sh --deps installe les paquets apt manquants, rien d autre
# sudo ./install.sh --with-deps les paquets manquants, puis tout
# ./install.sh --only build une ou des etapes (liste a virgules)
# ./install.sh --skip bin   toutes sauf celles-la
# ./install.sh --reinstall  rejoue meme ce qui est deja fait (reconfigure)
# ./install.sh -v           la sortie des commandes a l ecran (sinon : journal)
#
# Etapes : venv configure build install bin verify
#
# Options, et la variable qui fait la meme chose (l option l emporte) :
# --prefix DIR      PREFIX      $GSM_ROOT/qemu-install ou `make install` pose QEMU
# --build-dir DIR   BUILD_DIR   $QOSMO/build          construire hors de l arbre
# --venv DIR        VENV        $HOME/.venv-qemu          venv de configure (tomli) ; none =
# --bin-dir DIR     BIN_DIR     /usr/local/bin            copie de qemu-system-arm et qosmo
# --destdir DIR     DESTDIR     (vide)                    installer sous une autre racine (i
# --cc NOM          QOSMO_CC    (vide)                    compilateur impose, ex. gcc-13 (b
# -j N              JOBS        nproc
#                  GSM_ROOT     /opt/GSM      QOSMO = l arbre a construire, ce depot par d
#                  LOG_DIR      /tmp/qosmo-install (un journal par etape)
#
# Les lanceurs lisent $QOSMO/build/qemu-system-arm (c54x_exe/run.sh,
# environnement/bench.env) et c54x_exe / grgsm_exe compilent les sources de cet
```

```

# arbre : gardez l arbre ET build/.
#
# Correspondance avec osmo-operator/Dockerfile : chaque etape cite ses lignes,
# celles du Dockerfile au commit 4266c8b, quand il portait encore ces commandes
# (stage qemu, 691-705 ; paquets 156-205). Il appelle désormais ce script.
# -----
set -uo pipefail
ICI="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
#
# LE CONTRAT - celui de osmo-operator/install_modules/_lib/inst.sh
#
# Copie IDENTIQUE dans qosmo/install.sh, c54x_exe/install.sh et
# grgsm_exe/install.sh : chaque depot doit pouvoir s installer seul, sans
# osmo-operator. Une correction ici se reporte dans les deux autres.
#
# Quatre fonctions par etape, `run` seule obligatoire :
#   inst_<etape>_check      prerequis, LECTURE SEULE      0 ok · 1 manque · 4 sans objet
#   inst_<etape>_done       deja fait ?                  0 oui · 1 non
#   inst_<etape>_run        fait le travail              0 ok · 1 echec · 3 deja fait
#   inst_<etape>_verify     controle APRES coup          0 ok · 1 echec
# -----
readonly INST_RC_OK=0 INST_RC_FAIL=1 INST_RC_DONE=3 INST_RC_NA=4
declare -a INST_ORDER=()
declare -A INST_DESC=()
INST_REGISTER() { INST_ORDER+=("$1"); INST_DESC[$1]="$2"; }

_INST_REASON=""
_INST_HINT=""
inst_ok() { _INST_REASON="" ; return $INST_RC_OK; }
inst_fail() { _INST_REASON="$*" ; return $INST_RC_FAIL; }
inst_done() { _INST_REASON="$*" ; return $INST_RC_DONE; }
inst_na() { _INST_REASON="$*" ; return $INST_RC_NA; }
inst_hint() { _INST_HINT="$*"; }
inst_say() { printf '%s\n' "$*"; }

have_cmd() { command -v "$1" >/dev/null 2>&1; }
have_file() { [ -f "$1" ]; }
have_pkg() { dpkg -s "$1" >/dev/null 2>&1; }
# Un chemin est inscriptible s il l est, ou si son plus proche parent existant
# l est (on pourra le creer).
ecrivable() { local d="$1"; while [ ! -e "$d" ] && [ "$d" != / ]; do d="$(dirname "$d)"; done
# Chemin absolu sans exiger qu il existe (configure, make -C et les --dest

```

```

# changent de dossier : un chemin relatif n y voudrait plus rien dire).
absolu() { case "$1" in /*) printf '%s\n' "$1" ;; *) printf '%s\n' "$PWD/$1" ;; esac; }

# exige "ce qu il faut" "conseil si absent" commande... : une condition de
# prerequis. En --check elle est imprimee (ok / MANQUE + conseil) ; sinon elle
# est memorisee, et la premiere qui manque donne la raison de l echec.
_MANQUE=()
exige() {
    local quoi="$1" conseil="$2"; shift 2
    if "$@" >/dev/null 2>&1; then
        [ "$ACTION" = check ] && printf '          ok          %s\n' "$quoi"
        return 0
    fi
    if [ "$ACTION" = check ]; then
        printf '          %sMANQUE%s %s\n' "$C_KO" "$C_Z" "$quoi"
        [ -n "$conseil" ] && printf '                                → %s\n' "$conseil"
    fi
    [ ${#_MANQUE[@]} -eq 0 ] && [ -n "$conseil" ] && inst_hint "$conseil"
    _MANQUE+=("$quoi")
    return 1
}

# Verdict d une fonction check qui a appele exige.
bilan_exige() {
    [ ${#_MANQUE[@]} -eq 0 ] && { inst_ok; return $INST_RC_OK; }
    local plus=""; [ ${#_MANQUE[@]} -gt 1 ] && plus=" (+${(( ${#_MANQUE[@]} - 1 ))} autre(s), "
    inst_fail "manque : ${_MANQUE[0]}$plus"
}

#
# LE COMPOSANT - qosmo
#
COMPOSANT=qosmo
: "${GSM_ROOT:=/opt/GSM}"
: "${QOSMO:=ICI}"
: "${PREFIX:=GSM_ROOT/qemu-install}"
: "${BUILD_DIR:=}"
: "${VENV:=${HOME:-/root}/.venv-qemu}"
: "${BIN_DIR:=usr/local/bin}"
: "${DESTDIR:=}"
: "${QOSMO_CC:=}"
: "${JOBS:=$(nproc 2>/dev/null || echo 2)}"
: "${LOG_DIR:=tmp/qosmo-install}"

```

```

# Paquets apt, repris de la liste unique d osmo-operator/Dockerfile :
# :159      build-essential git pkg-config          outils de build
# :166      python3
# :189      python3-venv python3-pip                le venv de configure (tomli)
# :190      libglib2.0-dev libpixman-1-dev libslirp-dev ninja-build
#                                                  la cible arm-softmmu, meson
# :190      socat                                  run.sh / 40-qemu.sh parlent au moniteur (
PAQUETS=(build-essential git pkg-config python3 python3-venv python3-pip
          libglib2.0-dev libpixman-1-dev libslirp-dev ninja-build socat)

options_composant() {
    case "$1" in
        --prefix)    _val "$@"; PREFIX="$2";      _n=2 ;;
        --build-dir) _val "$@"; BUILD_DIR="$2";    _n=2 ;;
        --venv)       _val "$@"; VENV="$2";        _n=2 ;;
        --bin-dir)    _val "$@"; BIN_DIR="$2";     _n=2 ;;
        --destdir)    _val "$@"; DESTDIR="$2";     _n=2 ;;
        --cc)         _val "$@"; QOSMO_CC="$2";    _n=2 ;;
        -j)           _val "$@"; JOBS="$2";        _n=2 ;;
        -j[0-9]*)     JOBS="${1#-j}";             _n=1 ;;
    esac
}

apres_options() {
    QOSMO="$(absolu "$QOSMO")"
    [ -n "$BUILD_DIR" ] || BUILD_DIR="$QOSMO/build"
    BUILD_DIR="$(absolu "$BUILD_DIR")"; PREFIX="$(absolu "$PREFIX")"
    [ "$VENV" = none ] || VENV="$(absolu "$VENV")"
    [ "$BIN_DIR" = none ] || BIN_DIR="$(absolu "$BIN_DIR")"
    [ -z "$DESTDIR" ] || DESTDIR="$(absolu "$DESTDIR")"
    # Dockerfile:698-699 - les options de configure, a l identique.
    CONFIGURE_ARGS=(--target-list=arm-softmmu --enable-l1-grgsm "--prefix=$PREFIX"
                    --disable-werror --disable-docs)
}

resume_composant() {
    printf 'QOSMO=%s\nBUILD_DIR=%s\nPREFIX=%s\nVENV=%s\nBIN_DIR=%s\nJOBS=%s\n' \
        "$QOSMO" "$BUILD_DIR" "$PREFIX" "${DESTDIR:+ (sous DESTDIR=$DESTDIR)}" "$VENV" "$B
        "$JOBS" "${QOSMO_CC:+ QOSMO_CC=$QOSMO_CC}"
    printf 'configure %s\n' "${CONFIGURE_ARGS[*]}"
}

```

```

# --cc : le PATH local de Dockerfile.run:183-185. build/ garde le compilateur
# du configure (meson memorise « cc ») ; si ce cc n est plus le meme, ninja
# echoue des la premiere unite. On place donc devant le PATH un dossier ou
# cc/gcc (et c++/g++) designent le compilateur demande.
_CCDIR=""
avant_etapes() {
    [ -n "$QOSMO_CC" ] || return 0
    local cc cxx
    cc="$(command -v "$QOSMO_CC")" || { printf -- '--cc %s : introuvable\n' "$QOSMO_CC" >&2;
    cxx="$(command -v "${QOSMO_CC}/gcc/g++)" 2>/dev/null || true)
    _CCDIR="$(mktemp -d)" || exit 1
    trap 'rm -rf "$_CCDIR"' EXIT
    ln -s "$cc" "$_CCDIR/cc"; ln -s "$cc" "$_CCDIR/gcc"
    [ -n "$cxx" ] && { ln -s "$cxx" "$_CCDIR/c++"; ln -s "$cxx" "$_CCDIR/g++"; }
    return 0
}

# avec_env COMMANDE... : le venv active (Dockerfile:695) et le --cc eventuel.
avec_env() {
    local p="$PATH"
    [ "$VENV" != none ] && [ -d "$VENV/bin" ] && p="$VENV/bin:$p"
    [ -n "$_CCDIR" ] && p="$_CCDIR:$p"
    if [ "$VENV" != none ] && [ -d "$VENV/bin" ]; then
        PATH="$p" VIRTUAL_ENV="$VENV" "$@"
    else
        PATH="$p" "$@"
    fi
}

apres_etapes() {
    [ -x "$BUILD_DIR/qemu-system-arm" ] && printf '  qemu-system-arm : %s\n' "$BUILD_DIR/qemu-system-arm"
    printf '  ensuite : c54x_exe/install.sh (le DSP) - wiki osmo-operator, Telephone
}

# venv : python de configure
INST_REGISTER venv "Venv python de configure (tomli)"
inst_venv_done() { [ "$VENV" != none ] && "$VENV/bin/python3" -c 'import tomli' 2>/dev/null;
inst_venv_check() {
    [ "$VENV" = none ] && { inst_na "VENV=none : python du systeme"; return $INST_RC_NA; }
    _MANQUE=()
    exige "python3" "apt-get install python3" have_cmd python3
    exige "module venv + ensurepip (python3-venv)" "apt-get install python3-venv" python3 -c
    exige "$VENV inscriptible" "--venv DIR vers un dossier a vous, ou --venv none" ecrivable
}

```

```

    bilan_exige
}
inst_venv_run() {
    # Dockerfile:694-696 : python3 -m venv /root/.venv-qemu, activate, pip install tomli
    python3 -m venv "$VENV" \
        && "$VENV/bin/pip" install --no-cache-dir tomli \
        || { inst_hint "pip a besoin du reseau (PyPI)"; inst_fail "venv $VENV ou pip install
    inst_ok
}
inst_venv_verify() { inst_venv_done && inst_ok || inst_fail "tomli introuvable dans $VENV"; }

# configure
INST_REGISTER configure "configure --target-list=arm-softmmu --enable-l1-grgsm"
# config.log garde la ligne d appel : « # Configured with: '../configure' '--...' ».
# Deja fait = arbre configure AVEC les memes options (le 1er mot, le chemin de
# configure, ne compte pas).
_cfg_attendu() { printf "'%s' " "${CONFIGURE_ARGS[@]}" | sed 's/ $//'; }
_cfg_actuel() { sed -n "s/^# Configured with: '[^']*' //p" "$BUILD_DIR/config.log" 2>/dev/n
inst_configure_done() { [ -f "$BUILD_DIR/build.ninja" ] && [ "$(_cfg_actuel)" = "$(_cfg_atten
inst_configure_check() {
    _MANQUE=()
    exige "sources qosmo ($QOSMO/configure, hw/arm/calypso/)" "QOSMO=/chemin/vers/qosmo, ou g
        test -x "$QOSMO/configure" -a -d "$QOSMO/hw/arm/calypso"
    exige "compilateur C (${QOSMO_CC:-cc})" "apt-get install build-essential" have_cmd "${QOS
    exige "make" "apt-get install build-essential" have_cmd make
    exige "ninja (ninja-build)" "apt-get install ninja-build" have_cmd ninja
    exige "pkg-config" "apt-get install pkg-config" have_cmd pkg-config
    exige "glib-2.0 vu de pkg-config (libglib2.0-dev)" "apt-get install libglib2.0-dev" pkg-c
    exige "pixman-1 vu de pkg-config (libpixman-1-dev)" "apt-get install libpixman-1-dev" pkg
    exige "$BUILD_DIR inscriptible" "--build-dir DIR, ou les droits sur $QOSMO" ecrivable "$
    # configure lance DEPUIS l arbre source cree ./build et EFFACE un build/ qu il
    # a cree lui-meme (configure:16-45) : jamais de build dans l arbre source.
    exige "BUILD_DIR distinct de l arbre source" "--build-dir $QOSMO/build" test "$BUILD_DIR
    bilan_exige
}
inst_configure_run() {
    # Dockerfile:697-699 : mkdir -p build && cd build && ../configure <options>
    local conf="$QOSMO/configure"
    [ "$BUILD_DIR" = "$QOSMO/build" ] && conf=../configure
    mkdir -p "$BUILD_DIR" || { inst_fail "impossible de creer $BUILD_DIR"; return $INST_RC_F
    ( cd "$BUILD_DIR" && avec_env "$conf" "${CONFIGURE_ARGS[@]}" ) \
        || { inst_hint "$BUILD_DIR/config.log et $BUILD_DIR/meson-logs/"; inst_fail "configu

```

```

    inst_ok
}
inst_configure_verify() {
    [ -f "$BUILD_DIR/build.ninja" ] || { inst_fail "$BUILD_DIR/build.ninja absent apres conf
    grep -q '^TARGET_DIRS=arm-softmmu' "$BUILD_DIR/config-host.mak" 2>/dev/null \
        || { inst_fail "config-host.mak ne vise pas arm-softmmu"; return $INST_RC_FAIL; }
    inst_ok
}

# build
# Pas de « deja fait » : make est incremental, il ne refait que ce qui a bouge.
INST_REGISTER build "Compilation (make)"
inst_build_check() {
    _MANQUE=()
    etape_retenue configure || exige "arbre configure ($BUILD_DIR/build.ninja)" "./install.sl
        test -f "$BUILD_DIR/build.ninja"
    exige "make" "apt-get install build-essential" have_cmd make
    exige "ninja (ninja-build)" "apt-get install ninja-build" have_cmd ninja
    [ -n "$QOSMO_CC" ] && exige "compilateur $QOSMO_CC (--cc)" "" have_cmd "$QOSMO_CC"
    exige "$BUILD_DIR inscriptible" "les droits sur $BUILD_DIR" ecrivable "$BUILD_DIR"
    bilan_exige
}
inst_build_run() {
    # Dockerfile:700 : make -j$(nproc) dans build/ (le Makefile de QEMU mene ninja).
    # Dockerfile.run:185 faisait `ninja` avec le PATH --cc : meme cible.
    avec_env make -C "$BUILD_DIR" -j"$JOBS" || { inst_fail "echec de compilation"; return $INST_RC_FAIL; }
    inst_ok
}
inst_build_verify() {
    [ -x "$BUILD_DIR/qemu-system-arm" ] && inst_ok || inst_fail "$BUILD_DIR/qemu-system-arm a
}

# install : make install dans le prefixe
INST_REGISTER install "make install (prefixe, lanceur qosmo compris)"
inst_install_done() {
    cmp -s "$BUILD_DIR/qemu-system-arm" "$DESTDIR$PREFIX/bin/qemu-system-arm" \
        && [ -x "$DESTDIR$PREFIX/bin/qosmo" ]
}
inst_install_check() {
    _MANQUE=()
    etape_retenue configure || exige "arbre configure ($BUILD_DIR/build.ninja)" "./install.sl
        test -f "$BUILD_DIR/build.ninja"
}

```

```

    exige "$DESTDIR$PREFIX inscriptible" "--prefix DIR, ou sudo" ecrivable "$DESTDIR$PREFIX"
    bilan_exige
}
inst_install_run() {
    # Dockerfile:701 : make install (meson install ; DESTDIR respecte, cf. ISO)
    if [ -n "$DESTDIR" ]; then
        avec_env env DESTDIR="$DESTDIR" make -C "$BUILD_DIR" install
    else
        avec_env make -C "$BUILD_DIR" install
    fi || { inst_fail "make install en echec"; return $INST_RC_FAIL; }
    inst_ok
}
inst_install_verify() {
    [ -x "$DESTDIR$PREFIX/bin/qemu-system-arm" ] || { inst_fail "$DESTDIR$PREFIX/bin/qemu-sy
    [ -x "$DESTDIR$PREFIX/bin/qosmo" ] || { inst_fail "lanceur $DESTDIR$PREFIX/bin/qosmo abs
    inst_ok
}

# bin : copie dans le PATH
INST_REGISTER bin "qemu-system-arm et qosmo dans le PATH"
inst_bin_done() {
    [ "$BIN_DIR" != none ] || return 1
    local b; for b in qemu-system-arm qosmo; do
        cmp -s "$DESTDIR$PREFIX/bin/$b" "$DESTDIR$BIN_DIR/$b" || return 1
    done
}
inst_bin_check() {
    [ "$BIN_DIR" = none ] && { inst_na "--bin-dir none"; return $INST_RC_NA; }
    ecrivable "$DESTDIR$BIN_DIR" \
        || { inst_na "$DESTDIR$BIN_DIR non inscriptible : binaires laisses dans $PREFIX/bin
    [ "$ACTION" = check ] && printf '      ok      %s inscriptible\n' "$DESTDIR$BIN_DIR"
    inst_ok
}
inst_bin_run() {
    # Dockerfile:702-703 : cp qemu-install/bin/{qemu-system-arm,qosmo} /usr/local/bin/
    # --remove-destination : un qemu-system-arm en cours d execution garde son
    # inode, la copie ne bute pas sur « Text file busy ».
    local b
    mkdir -p "$DESTDIR$BIN_DIR" || { inst_fail "impossible de creer $DESTDIR$BIN_DIR"; return
    for b in qemu-system-arm qosmo; do
        [ -x "$DESTDIR$PREFIX/bin/$b" ] || { inst_fail "$DESTDIR$PREFIX/bin/$b absent (etape
        cp --remove-destination "$DESTDIR$PREFIX/bin/$b" "$DESTDIR$BIN_DIR/$b" \

```

```

        || { inst_fail "copie de $b vers $DESTDIR$BIN_DIR impossible"; return $INST_RC_FAIL; }
    inst_say "$DESTDIR$BIN_DIR/$b"
done
inst_ok
}

inst_bin_verify() { inst_bin_done && inst_ok || inst_fail "$DESTDIR$BIN_DIR differe de $DESTDIR$BIN_DIR"

# verify : la machine calypso, avec la couche 1 gr-gsm
# Le meme binaire sert aux deux montages : compile --enable-l1-grgsm, lance avec
# CALYPSO_DSP_EXTERN=1 pour le DSP (Dockerfile:679-682). Sans la couche 1
# gr-gsm, la description de la machine le dit (hw/arm/calypso/calypso_mb.c).
INST_REGISTER verify "Controle : -M help liste calypso (couche 1 : grgsm)"
inst_verify_run() {
    local q="$BUILD_DIR/qemu-system-arm" 1
    [ -x "$q" ] || { inst_hint "./install.sh --only build"; inst_fail "$q absent"; return $INST_RC_FAIL; }
    l="$(timeout 30 "$q" -M help 2>/dev/null | grep -E '^calypso[[:space:]]')"
    inst_say "$q -M help : ${l:-<pas de machine calypso>}"
    [ -n "$l" ] || { inst_hint "le configure a-t-il garde --enable-l1-grgsm ? ($BUILD_DIR/calypso)";
        inst_fail "machine calypso absente de -M help"; return $INST_RC_FAIL; }

    case "$l" in
        *"couche 1 : grgsm"*) ;;
        *) inst_fail "machine calypso sans la couche 1 gr-gsm : $l"; return $INST_RC_FAIL ;;
    esac
    if [ -z "$DESTDIR" ] && [ -x "$PREFIX/bin/qosmo" ]; then
        inst_say "lanceur : $("PREFIX/bin/qosmo" -V 2>&1 | head -1)"
    fi
    inst_ok
}

#
# LE MOTEUR - copie IDENTIQUE dans les trois depots (voir LE CONTRAT)
#
# Meme deroule que osmo-operator/install.sh : pour chaque etape retenue,
# « deja fait ? » (sauf --reinstall), prerequis, travail, controle. Le premier
# echec arrete tout, dit pourquoi et montre la fin du journal de l etape.
# Le composant fournit : COMPOSANT, PAQUETS, LOG_DIR, les etapes, et les
# crochets options_composant / apres_options / resume_composant /
# avant_etapes / apres_etapes.
# -----
ACTION=install ONLY="" SKIP="" WITH_DEPS=0
: "${REINSTALL:=0}" "${VERBOSE:=0}"
_ARGS_ORIG=("${@}")
# _val OPTION [VALEUR] : une option qui attend une valeur doit l avoir.

```

```

_val() { [ $# -ge 2 ] && [ -n "$2" ] || { printf '%s : valeur attendue  (--help)\n' "$1" >&2;
while [ $# -gt 0 ]; do
    case "$1" in
        -h|--help)    ACTION=help ;;
        --check)      ACTION=check ;;
        --list)       ACTION=list ;;
        --deps)       ACTION=deps ;;
        --print-deps) ACTION=print-deps ;;
        --with-deps)  WITH_DEPS=1 ;;
        --only)       _val "$@"; ONLY="$2"; shift ;;
        --skip)       _val "$@"; SKIP="$2"; shift ;;
        --reinstall)  REINSTALL=1 ;;
        -v|--verbose) VERBOSE=1 ;;
        *)            _n=0; options_composant "$@"
                     [ "$_n" -gt 0 ] || { printf 'option inconnue : %s  (--help)\n' "$1" >&2; exit 2
                     shift $((_n - 1)) ;;
    esac
    shift
done

if [ "$ACTION" = help ]; then
    awk 'NR > 2 && /^# -{10}/ { exit } NR > 2 && !/^# =+$/ { sub(/^# ?/, ""); print }' "$0"
    exit 0
fi
apres_options

if [ -t 1 ] && [ -z "${NO_COLOR:-}" ]; then
    TTY=1; C_OK=$'\033[32m'; C_KO=$'\033[31m'; C_SK=$'\033[33m'; C_DIM=$'\033[2m'; C_Z=$'\033[30m';
else
    TTY=0; C_OK=""; C_KO=""; C_SK=""; C_DIM=""; C_Z=""
fi

begin() { if [ $TTY -eq 1 ] && [ "$VERBOSE" != 1 ]; then printf '[ %s..%s ] %s' "$C_DIM" "$C_SK" "$C_Z"; }
end()   { [ $TTY -eq 1 ] && [ "$VERBOSE" != 1 ] && printf '\r\033[K'
         printf '[%s%s%s] %s' "$2" "$1" "$C_Z" "$3"
         [ -n "${4:-}" ] && printf ' %s(%s)%s' "$C_DIM" "$4" "$C_Z"; printf '\n'; }

# --- selection des etapes (--only / --skip, listes a virgules) -----
in_csv() { case ",$1," in *",$2,") return 0;; esac; return 1; }
for _s in ${ONLY//,/ } ${SKIP//,/ }; do
    [ -n "${INST_DESC[_s]+x}" ] || { printf 'etape inconnue : %s  (etapes : %s)\n' "$_s" "$INST_DESC[@]"; }
done
SELECTED=()

```

```

for _s in "${INST_ORDER[@]"}"; do
    [ -n "$ONLY" ] && ! in_csv "$ONLY" "$_s" && continue
    [ -n "$SKIP" ] && in_csv "$SKIP" "$_s" && continue
    SELECTED+="$_s"
done
# Une etape qui produit ce qu'une autre exige : en --check, on ne reproche pas
# a la seconde l'absence de ce que la premiere va justement fabriquer.
etape_retenue() { in_csv "$(IFS=,; echo "${SELECTED[*]}")" "$1"; }

# --- paquets apt -----
# rend la liste des absents ; code 2 sans dpkg (pas une Debian/Ubuntu)
paquets_absents() {
    have_cmd dpkg || return 2
    local p; for p in "${PAQUETS[@]"}"; do have_pkg "$p" || printf '%s ' "$p"; done
    return 0
}
installer_paquets() {
    local manque rc
    manque="$(paquets_absents)"; rc=$?
    if [ $rc -eq 2 ]; then
        printf 'dpkg absent : installez l'equivalent de : %s\n' "${PAQUETS[*]}" >&2; return 1
    fi
    if [ -z "${manque// }" ]; then
        printf 'paquets apt : les %d sont presents\n' "${#PAQUETS[@]"}; return 0
    fi
    if [ "$(id -u)" -ne 0 ]; then
        printf 'paquets apt manquants : %s\n' "$(manque)"; sudo apt-get install -y --no-install-recommends
        return 1
    fi
    printf 'apt-get install : %s\n' "$manque"
    apt-get update -qq && DEBIAN_FRONTEND=noninteractive apt-get install -y --no-install-recommends
    manque="$(paquets_absents)"
    [ -z "${manque// }" ] || { printf 'toujours absents apres apt-get : %s\n' "$manque" >&2;
}

case "$ACTION" in
    print-deps) printf '%s\n' "${PAQUETS[*]}"; exit 0 ;;
    deps)       installer_paquets; exit $? ;;
    list)
        printf '%s - %d etape(s), dans cet ordre\n\n' "$COMPOSANT" "${#SELECTED[@]}"
        for _s in "${SELECTED[@]"}"; do printf ' %-10s %s\n' "$_s" "${INST_DESC[$_s]}"; done
        printf '\n'; resume_composant | sed 's/^/ /'

```

```

        exit 0 ;;
check)
    printf '%s : prerequisites et etat (rien n est modifie)\n\n' "$COMPOSANT"
    resume_composant | sed 's/^/ /'
    _manque="$(paquets_absents)"; _rc=$?
    if [ $_rc -eq 2 ]; then printf '\n paquets apt : dpkg absent, non verifiees (%s)\n'
    elif [ -n "${_manque// }" ]; then printf '\n paquets apt absents (installes autrement ? si
    else printf '\n paquets apt : les %d sont presents\n' "${#PAQUETS[@]}"; fi
    _rc=0
    for _s in "${SELECTED[@]}"; do
        _p="inst_${_s//-/}_/"
        if ! declare -F "${_p}_done" >/dev/null; then _etat="pas de controle"
        elif "${_p}_done" >/dev/null 2>&1; then _etat="deja fait"
        else _etat="a faire"; fi
        printf '\n %-10s %s %s[%s]\n' "$_s" "${INST_DESC[$_s]}" "$C_DIM" "$_etat" "$C_Z"
        declare -F "${_p}_check" >/dev/null || continue
        _INST_REASON=""; _INST_HINT=""; _MANQUE=()
        "${_p}_check"; _r=$?
        case $_r in
            "$INST_RC_FAIL") _rc=1 ;;
            "$INST_RC_NA") printf '          sans objet : %s\n' "$_INST_REASON" ;;
        esac
    done
    printf '\n'
    if [ $_rc -eq 0 ]; then printf '%sprerequisites satisfaites\n' "$C_OK" "$C_Z"
    else printf '%sprerequisites manquantes (voir MANQUE ci-dessus)\n' "$C_KO" "$C_Z"; fi
    exit $_rc ;;
esac

# --- installation -----
if [ "$WITH_DEPS" = 1 ]; then
    installer_paquets || { printf 'paquets apt : echec - rien n a ete construit\n' >&2; exit 1; }
else
    _manque="$(paquets_absents 2>/dev/null)"
    [ -n "${_manque// }" ] && printf '%snote : paquets apt absents (installes autrement ? si
fi
if [ "$VERBOSE" != 1 ]; then
    mkdir -p "$LOG_DIR" || { printf 'journal impossible : %s (LOG_DIR=...)\n' "$LOG_DIR" >&2; }
fi
# _jouer FONCTION : la sortie va au journal de l etape, ou a l ecran en -v.
# Pas de tube : la fonction tourne dans CE shell (sa raison d echec survit).
_jouer() { if [ "$VERBOSE" = 1 ]; then "$1"; else "$1" >>"$log" 2>&1; fi; }

```

```

_echec() { # _echec ETAPE RAISON [journal] : dit tout ce qu on sait, et s arrete
end FAIL "$C_KO" "${INST_DESC[$1]}" "$2"
[ -n "$_INST_HINT" ] && printf '          → %s\n' "$_INST_HINT"
# la fin du journal, quand c est le travail (run, verify) qui a echoue
if [ -n "${3:-}" ] && [ "$VERBOSE" != 1 ] && [ -s "$_log" ]; then
    printf '          %sjournal : %s (fin ci-dessous)%s\n' "$C_DIM" "$_log" "$C_Z"
    tail -n 25 "$_log" | sed 's/^/          | /'
fi
printf '\n%s : installation interrompue a l etape %s\n' "$COMPOSANT" "$1"
exit 1
}

avant_etapes
_nb_ok=0; _nb_skip=0
for _s in "${SELECTED[@]}; do
    _p="inst_${_s//-/}_"; _log="$LOG_DIR/${_s}.log"
    _INST_REASON=""; _INST_HINT=""; _MANQUE=()
    [ "$VERBOSE" = 1 ] || printf '\n==== %s %s =====\n' "$(date '+%F %T')" "$_s" >>"$_log"
    begin "${INST_DESC[$_s]}"
    if [ "$REINSTALL" != 1 ] && declare -F "${_p}_done" >/dev/null && _jouer "${_p}_done"; then
        end SKIP "$C_SK" "${INST_DESC[$_s]}" "deja fait"; _nb_skip=$(( _nb_skip + 1 )); continue
    fi
    if declare -F "${_p}_check" >/dev/null; then
        _jouer "${_p}_check"; _r=$?
        case $_r in
            "$INST_RC_OK") ;;
            "$INST_RC_NA") end SKIP "$C_SK" "${INST_DESC[$_s]}" "${_INST_REASON:-sans objet}";
                           _nb_skip=$(( _nb_skip + 1 )); continue ;;
            *) _echec "$_s" "${_INST_REASON:-prerequis non satisfait}" ;;
        esac
    fi
    _jouer "${_p}_run"; _r=$?
    case $_r in
        "$INST_RC_OK") ;;
        "$INST_RC_DONE"|"${INST_RC_NA}")
            end SKIP "$C_SK" "${INST_DESC[$_s]}" "${_INST_REASON:-rien a faire}";
            _nb_skip=$(( _nb_skip + 1 )); continue ;;
        *) _echec "$_s" "${_INST_REASON:-echec}" journal ;;
    esac
    # installer n est pas avoir installe : on controle apres coup
    if declare -F "${_p}_verify" >/dev/null && ! _jouer "${_p}_verify"; then
        _echec "$_s" "faite, mais le controle echoue : ${_INST_REASON:-}" journal
    fi
done

```

```

    end " OK " "$C_OK" "${INST_DESC[$_s]}" "${_INST_REASON:-}"; _nb_ok=$(( _nb_ok + 1 ))
done
printf '\n%s : %d ok · %d ignoree(s) · 0 echec\n' "$COMPOSANT" "$_nb_ok" "$_nb_skip"
apres_etapes
exit 0

```

0.3.2 /opt/GSM/qosmo-overlay/make-overlay.sh

3714 octets, 90 lignes → 90 lignes

```

#!/bin/bash
# make-overlay.sh - range dans un dossier les fichiers propres a qosmo.
#
#     QEMU vanilla (v$(cat VERSION)) + overlay = qosmo
#
# « Propre a qosmo » = suivi par git ici ET (absent du QEMU vanilla OU different
# de lui). Tout le reste est du QEMU d'origine et reste dehors. Les liens
# symboliques ne comptent que si leur cible differe.
#
# Le dossier est reconstruit a chaque passage (pas de fichier perime) et porte :
#   MANIFEST.ajoutes  fichiers absents du vanilla
#   MANIFEST.modifies fichiers vanilla modifies par qosmo
#   MANIFEST.retires  fichiers vanilla que qosmo ne suit pas (CI gitlab, roms/...),
#                       pour information : l'overlay ne les supprime pas
#
# Reconstituer l'arbre : cp -a <vanilla> arbre && cp -a <overlay>/. arbre/
#                       (puis rm arbre/MANIFEST.*)
#
# Le vanilla est pris dans $QEMU_GENUINE (default ../qemu-<VERSION>) ; absent, il
# est clone (depth 1, tag v<VERSION>) depuis gitlab.com/qemu-project/qemu.
#
# Usage : ./make-overlay.sh [--dry-run] [DOSSIER] (default ../qosmo-overlay)
set -euo pipefail
DRY=0; [ "${1:-}" = "--dry-run" ] && { DRY=1; shift; }
SRC="$(cd "$(dirname "$0")" && pwd)"
VER="$(cat "$SRC/VERSION)"
OUT="${1:-$SRC/../qosmo-overlay}"
GEN="${QEMU_GENUINE:-$SRC/../qemu-$VER}"

if [ ! -f "$GEN/VERSION" ]; then
    echo "vanilla absent : clone de v$VER dans $GEN"

```

```

    git clone -q --depth 1 --branch "v$VER" https://gitlab.com/qemu-project/qemu.git "$GEN"
fi
[ "$(cat "$GEN/VERSION")" = "$VER" ] || {
    echo "!! $GEN est en $(cat "$GEN/VERSION"), qosmo en $VER" >&2; exit 1; }

# Artefacts qui n'ont rien a faire dans un overlay, meme suivis par erreur.
exclu() {
    case "$1" in
        */__pycache__/*|*.pyc|*.o|*.d|*.orig|*.rej|*~|*.swp) return 0;;
    esac
    return 1
}

# Meme contenu ? Un lien se compare sur sa cible, pas sur ce qu'il pointe.
pareil() {
    local a="$SRC/$1" b="$GEN/$1"
    if [ -L "$a" ] || [ -L "$b" ]; then
        [ -L "$a" ] && [ -L "$b" ] && [ "$(readlink "$a")" = "$(readlink "$b")" ]
    else
        [ -f "$b" ] && cmp -s "$a" "$b"
    fi
}

tmp="$(mktemp -d)"; trap 'rm -rf "$tmp"' EXIT
(cd "$GEN" && git ls-files) | sort > "$tmp/vanilla"
(cd "$SRC" && git ls-files) | sort > "$tmp/qosmo"
: > "$tmp/ajoutes"; : > "$tmp/modifies"; skip=0
while IFS= read -r f; do
    if exclu "$f"; then skip=$((skip+1)); continue; fi
    [ -e "$SRC/$f" ] || [ -L "$SRC/$f" ] || continue # suivi mais supprime du disque
    echo "$f" >> "$tmp/ajoutes"
done < <(comm -13 "$tmp/vanilla" "$tmp/qosmo")
while IFS= read -r f; do
    pareil "$f" || echo "$f" >> "$tmp/modifies"
done < <(comm -12 "$tmp/vanilla" "$tmp/qosmo")
comm -23 "$tmp/vanilla" "$tmp/qosmo" > "$tmp/retires"

na=$(wc -l < "$tmp/ajoutes"); nm=$(wc -l < "$tmp/modifies"); nr=$(wc -l < "$tmp/retires")
echo "qosmo      : $SRC ($VER)"
echo "vanilla    : $GEN"
echo "overlay     : $OUT"
echo "-----"

```

```

sed 's/^/ ajoute /' "$tmp/ajoutes"
sed 's/^/ modifie /' "$tmp/modifies"
echo "-----"
echo "$na ajoute(s), $nm modifie(s), $nr vanilla non suivi(s), $skip artefact(s) ecarte(s)"
[ "$DRY" = 1 ] && { echo "DRY-RUN : rien d'ecrit."; exit 0; }

# On ne vide que ce que ce script a lui-meme produit.
if [ -e "$OUT" ] && [ -n "$(ls -A "$OUT")" ] && [ ! -f "$OUT/MANIFEST.ajoutes" ]; then
    echo "!! $OUT existe, n'est pas vide et n'est pas un overlay qosmo : je n'y touche pas"
    exit 1
fi
rm -rf "$OUT"; mkdir -p "$OUT"
(cd "$SRC" && cat "$tmp/ajoutes" "$tmp/modifies" | xargs -r -d "\n" cp -a --parents -t "$OUT"
cp "$tmp/ajoutes" "$OUT/MANIFEST.ajoutes"
cp "$tmp/modifies" "$OUT/MANIFEST.modifies"
cp "$tmp/retires" "$OUT/MANIFEST.retires"
echo "ecrit : $(find "$OUT" -type f -o -type l | wc -l) entree(s) dans $OUT"

```

0.3.3 /opt/GSM/qosmo-overlay/run.sh

12847 octets, 311 lignes → 311 lignes

```

#!/bin/bash
set -uo pipefail

HERE="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
MODDIR="$HERE/run_modules"

DRY=0 ONLY="" SKIP="" ACTION=start FORCE=0
NO_ATTACH="${CALYPSO_NO_ATTACH:-0}"

usage() {
    cat <<'USAGE'
Usage : ./run.sh [options]
    --list           affiche le plan et sort, sans rien lancer
    --dry-run        déroule le plan sans effet de bord
    --only <slugs>   ne joue que ces modules (séparés par des virgules)
    --skip <slugs>   saute ces modules
    --stop           arrête la pile (plan en ordre inverse)
    --status         interroge l'état de chaque module
    --force          relance même les modules déjà démarrés

```

```

--no-attach      ne pas s'attacher à tmux à la fin
--check-paths    vérifie que les dépendances déclarées existent
--restart        = --reset puis démarrage - la relance sûre au quotidien
--reset          repart d'un état propre : arrête la pile, tue le serveur
                  tmux, archive les journaux, purge sockets et fichiers partagés
--lang <code>    langue de l'affichage (fr de en es pt it hi zh ar)
--icount[=SPEC]  QEMU avec -icount SPEC (défaut : auto) ; = QEMU_ICOUNT
--asm[=FICHIER]  QEMU avec -d in_asm -D FICHIER (défaut : $LOG_DIR/qemu-asm.log) ; = QEMU_ASM
--profile <nom>  accepté pour compatibilité, sans effet (un seul banc)
--verbose        accepté pour compatibilité, sans effet
-h, --help      cette aide
USAGE
}

_reset() {
    local horodat rep n=0
    printf '\nRemise à zéro - rien ne sera relancé.\n\n' >&2
    printf '  arrêt de la pile...\n' >&2
    "$0" --stop >/dev/null 2>&1 || true
    local logdir="$LOG_DIR"
    if [ -d "$logdir" ] && [ -n "$(ls -A "$logdir" 2>/dev/null)" ]; then
        horodat="$(date +%Y%m%d-%H%M%S)"
        rep="$logdir/./archives/$horodat"
        mkdir -p "$rep" 2>/dev/null && \
        if cp -a "$logdir/." "$rep/" 2>/dev/null; then
            printf '  journaux archivés      %s\n' "$rep" >&2
        fi
        local arcdir="$logdir/./archives"
        local keep="${CALYPSO_ARCHIVES_KEEP:-3}"
        local maxmb="${CALYPSO_ARCHIVE_MAX_MB:-32}"
        local f taille avant apres
        for f in "$rep"/*; do
            [ -f "$f" ] || continue
            taille=$(stat -c %s "$f" 2>/dev/null || echo 0)
            if [ "$taille" -gt $(( maxmb * 1024 * 1024 )) ]; then
                head -c $(( maxmb * 1024 * 1024 )) "$f" > "$f.tete" 2>/dev/null &&
                mv -f "$f.tete" "$f" 2>/dev/null &&
                printf '  archive tronquée      %s (%s Mo -> %s Mo, tête conservée)\n' \
                    "$(basename "$f")" "$(( taille / 1048576 ))" "$maxmb" >&2
            fi
        done
        if [ -d "$arcdir" ]; then

```

```

    avant=$(du -sm "$arcdire" 2>/dev/null | cut -f1)
    ls -lt "$arcdire" 2>/dev/null | tail -n +$( ( keep + 1 ) ) | while read -r vieux; do
        [ -n "$vieux" ] && rm -rf -- "$arcdire/$vieux"
    done
    apres=$(du -sm "$arcdire" 2>/dev/null | cut -f1)
    [ "${avant:-0}" -gt "${apres:-0}" ] 2>/dev/null &&
        printf '    archives purgées      %s gardées (%s Mo -> %s Mo)\n' \
            "$keep" "$avant" "$apres" >&2

fi
rm -f "$logdir"/*.log 2>/dev/null
printf '    journaux remis à zéro %s\n' "$logdir" >&2
df -h "$logdir" 2>/dev/null | tail -1 | \
    awk '{printf "    espace disponible      %s (%s utilisé)\n", $4, $5}' >&2
fi
if command -v tmux >/dev/null 2>&1 && tmux ls >/dev/null 2>&1; then
    tmux kill-server 2>/dev/null || true
    printf '    serveur tmux tué      (environnement global purgé)\n' >&2
fi
local motifs="qemu-system-arm osmocon osmo-bts-trx fake_trx trxcon"
local m p pids
for m in $motifs; do
    pids="$(pgrep -f -- "$m" 2>/dev/null || true)"
    for p in $pids; do
        [ "$p" = "$$" ] && continue
        [ "$p" = "${PPID:-0}" ] && continue
        kill -TERM "$p" 2>/dev/null && n=$((n + 1))
    done
done
[ "$n" -gt 0 ] && printf '    processus arrêtés      %d\n' "$n" >&2
local art
for art in /tmp/osmocom_l2 /tmp/ms2_l2 /dev/shm/calypso_*; do
    [ -e "$art" ] && rm -f "$art" 2>/dev/null && printf '    supprimé      %s\n' "$art" >&2
done
printf '\nÉtat propre. Relancez :\n' >&2
printf '    ./run.sh\n\n' >&2
}

while [ $# -gt 0 ]; do
    case "$1" in
        --list)      ACTION=list ;;
        --dry-run)    DRY=1 ;;
        --only)       ONLY="${2:-}"; shift ;;
    esac
done

```

```

--skip)      SKIP="${2:-}"; shift ;;
--profile)   shift ;;
--stop)      ACTION=stop ;;
--status)    ACTION=status ;;
--force)     FORCE=1 ;;
--no-attach) NO_ATTACH=1 ;;
--verbose)   ;;
--check-paths) ACTION=checkpaths ;;
--reset)     ACTION=reset ;;
--restart)   ACTION=restart ;;
--lang)      CALYPSO_LANG="${2:-}"; export CALYPSO_LANG; shift ;;
--icount)    export QEMU_ICOUNT=auto ;;
--icount=*)  export QEMU_ICOUNT="${1#--icount=}" ;;
--asm)       export QEMU_ASM=1 ;;
--asm=*)     export QEMU_ASM="${1#--asm=}" ;;
-h|--help)   usage; exit 0 ;;
*) printf 'option inconnue : %s\n\n' "$1" >&2; usage >&2; exit 2 ;;
esac
shift
done

if [ -f "$HERE/environnement/bench.env" ]; then
    set -a; . "$HERE/environnement/bench.env"; set +a
else
    printf 'configuration introuvable : %s\n' "$HERE/environnement/bench.env" >&2
    exit 2
fi

if [ "$ACTION" = reset ]; then _reset; exit 0; fi
if [ "$ACTION" = restart ]; then _reset; ACTION=start; fi

LOGDIR="$LOG_DIR"
mkdir -p "$LOGDIR/mod" 2>/dev/null || true

if [ -t 1 ] && [ -z "${NO_COLOR:-}" ]; then
    TTY=1; C_OK='${\033[32m}'; C_KO='${\033[31m}'; C_SK='${\033[33m}'; C_DIM='${\033[2m}'; C_Z='${\033[30m}';
else
    TTY=0; C_OK=""; C_KO=""; C_SK=""; C_DIM=""; C_Z=""
fi

say_begin() { if [ $TTY -eq 1 ]; then printf '[ %s.. %s] %s' "$C_DIM" "$C_Z" "$1"; else printf '%s\n' "$1"; fi }
say_end() { if [ $TTY -eq 1 ]; then printf '\r\033[K'; fi }

```

```

    printf "[%s%-4s%s] %s" "$2" "$1" "$C_Z" "$3"
    [ -n "${4:-}" ] && printf " %s(%s)%s" "$C_DIM" "$4" "$C_Z"
    printf '\n'
}

. "$MODDIR/_lib/mod.sh"
shopt -s nullglob
for f in "$MODDIR"/[0-9][0-9]*.sh; do . "$f"; done
shopt -u nullglob

if [ ${#MOD_ORDER[@]} -eq 0 ]; then
    printf '%s\n' "$(t aucun_mod "$MODDIR")" >&2
    exit 2
fi

in_csv() { case ",$1," in *",$2,") return 0;; esac; return 1; }
selected=()
for m in "${MOD_ORDER[@]}; do
    [ -n "$ONLY" ] && { in_csv "$ONLY" "$m" || continue; }
    [ -n "$SKIP" ] && { in_csv "$SKIP" "$m" && continue; }
    selected+=("$m")
done

if [ "$ACTION" = list ]; then
    printf '%s\n\n' "$(t plan "${#selected[@]}")"
    for m in "${selected[@]}; do
        printf " %-18s %-42s %s\n" "$m" "$(t_mod "$m")" \
            "$([ "${MOD_REQUIRED[$m]}" = 1 ] && t obligatoire || t optionnel)" \
            "$([ -n "${MOD_DEPS[$m]}" ] && printf " %s" "$(t apres "${MOD_DEPS[$m]}")")"
    done
    exit 0
fi

if [ "$ACTION" = checkpaths ]; then
    rc=0
    for v in QEMU_BIN FIRMWARE_ELF FIRMWARE_BIN OSMOCON OSMOCOM_CFG; do
        p="${!v:-}"
        if [ -z "$p" ]; then say_end "$(t fail)" "$C_KO" "$v" "non défini"; rc=1
        elif [ -e "$p" ]; then say_end " $(t ok) " "$C_OK" "$v" "$p"
        else say_end "$(t fail)" "$C_KO" "$v" "introuvable : $p"; rc=1;
        done
    exit $rc

```

```

fi

declare -A STATE=()
nb_ok=0 nb_fail=0 nb_skip=0 nb_warn=0
[ "$ACTION" = stop ] && { tmp=(); for ((i=${#selected[@]}-1;i>=0;i--)); do tmp+="${selected[@]}"; done

for m in "${selected[@]"; do
    p="$(mod_prefix "$m")"
    log="${MOD_LOG[$m]:-$LOGDIR/mod/$m.log}"
    _MOD_REASON=""; _MOD_HINT=""

    if [ "$ACTION" = stop ]; then
        say_begin "$(t arret "$(t_mod "$m")")"
        declare -F "${p}_stop" >/dev/null && "${p}_stop" >>"$log" 2>&1
        say_end " $(t ok) " "$C_OK" "$(t arret "$(t_mod "$m")")"
        continue
    fi

    if [ "$ACTION" = status ]; then
        if declare -F "${p}_status" >/dev/null && "${p}_status" >>"$log" 2>&1; then
            say_end "$(t up)" "$C_OK" "$(t_mod "$m")"
        else
            say_end "$(t down)" "$C_SK" "$(t_mod "$m")"
        fi
        continue
    fi

    if ! eval "${MOD_ENABLED_IF[$m]}" 2>/dev/null; then
        say_end "$(t skip)" "$C_SK" "$(t_mod "$m")" "$(t desactive "${MOD_ENABLED_IF[$m]}")"
        STATE[$m]=skip; nb_skip=$((nb_skip+1)); continue
    fi

    dep_ko=""
    for d in ${MOD_DEPS[$m]}; do
        case "${STATE[$d]:-absent}" in ok|skip) ;; *) dep_ko="$d"; break;; esac
    done
    if [ -n "$dep_ko" ]; then
        say_end "$(t skip)" "$C_SK" "$(t_mod "$m")" "$(t dep_ko "$dep_ko")"
        STATE[$m]=skip; nb_skip=$((nb_skip+1)); continue
    fi

    say_begin "$(t_mod "$m")"

```

```

if [ $FORCE -eq 0 ] && declare -F "${p}_status" >/dev/null && "${p}_status" >>"$log" 2>&1;
    say_end "$(t skip)" "$C_SK" "$(t_mod "$m")" "$(t deja)"
    STATE[$m]=skip; nb_skip=$((nb_skip+1)); continue
fi

if declare -F "${p}_check" >/dev/null; then
    "${p}_check" >>"$log" 2>&1; rc=$?
    if [ $rc -eq $MOD_RC_FAIL ] && [ $DRY -eq 1 ]; then
        say_end "$(t dry)" "$C_DIM" "$(t_mod "$m")" \
            "$(t prereq_sim "${_MOD_REASON:-}")"
        STATE[$m]=skip; nb_skip=$((nb_skip+1)); continue
    fi
    if [ $rc -eq $MOD_RC_FAIL ]; then
        say_end "$(t fail)" "$C_KO" "$(t_mod "$m")" "${_MOD_REASON:-$(t sans_raison)}"
        [ -n "$_MOD_HINT" ] && printf '          %s→ %s%s\n' "$C_DIM" "$_MOD_HINT" "$C_Z"
        printf '          %s%s%s\n' "$C_DIM" "$(t journal "$log")" "$C_Z"
        STATE[$m]=fail; nb_fail=$((nb_fail+1))
        [ "${MOD_REQUIRED[$m]}" = 1 ] && { printf '\n%s\n' "$(t interrompu)"; exit 1; }
        nb_warn=$((nb_warn+1)); continue
    elif [ $rc -eq $MOD_RC_SKIP ]; then
        say_end "$(t skip)" "$C_SK" "$(t_mod "$m")" "${_MOD_REASON:-}"
        STATE[$m]=skip; nb_skip=$((nb_skip+1)); continue
    fi
fi

if [ $DRY -eq 1 ] && [ "${MOD_PURE[$m]}" != 1 ]; then
    say_end " -- " "$C_DIM" "$(t_mod "$m")" "$(t simule)"
    STATE[$m]=ok; continue
fi

"${p}_start" >>"$log" 2>&1; rc=$?
case $rc in
    $MOD_RC_ALREADY) say_end "$(t skip)" "$C_SK" "$(t_mod "$m")" "${_MOD_REASON:-$(t deja)}"
                      STATE[$m]=skip; nb_skip=$((nb_skip+1)); continue ;;
    $MOD_RC_SKIP)    say_end "$(t skip)" "$C_SK" "$(t_mod "$m")" "${_MOD_REASON:-}"
                      STATE[$m]=skip; nb_skip=$((nb_skip+1)); continue ;;
    $MOD_RC_FAIL)    say_end "$(t fail)" "$C_KO" "$(t_mod "$m")" "${_MOD_REASON:-$(t sans_raison)}"
                      [ -n "$_MOD_HINT" ] && printf '          %s→ %s%s\n' "$C_DIM" "$_MOD_HINT" "$C_Z"
                      printf '          %s%s%s\n' "$C_DIM" "$(t journal "$log")" "$C_Z"
                      STATE[$m]=fail; nb_fail=$((nb_fail+1))
                      [ "${MOD_REQUIRED[$m]}" = 1 ] && { printf '\n%s\n' "$(t interrompu)"; exit 1; }
                      nb_warn=$((nb_warn+1)); continue ;;

```

```

esac

if declare -F "${p}_wait" >/dev/null; then
    if ! "${p}_wait" >>"$log" 2>&1; then
        say_end "$(t fail)" "$C_KO" "$(t_mod "$m")" "$(t pas_pret "${_MOD_REASON:-délai (
        printf '          %s%s\n' "$C_DIM" "$(t journal "$log")" "$C_Z"
        STATE[$m]=fail; nb_fail=$((nb_fail+1))
        [ "${MOD_REQUIRED[$m]}" = 1 ] && { printf '\n%s\n' "$(t interrompu)"; exit 1; }
        nb_warn=$((nb_warn+1)); continue
    fi
fi

say_end " $(t ok) " "$C_OK" "$(t_mod "$m")"
STATE[$m]=ok; nb_ok=$((nb_ok+1))
done

printf '\n%s\n' "$(t bilan "$nb_ok" "$nb_skip" "$nb_fail")"

if [ "$ACTION" = start ] && [ $DRY -eq 0 ] && [ $nb_fail -eq 0 ]; then
    _sess="${TMUX_SESSION:-calypso}"
    printf '\n'
    if tmux has-session -t "$_sess" 2>/dev/null; then
        printf '  %s%s    tmux attach -t %s\n' "$C_OK" "$(t connecter)" "$C_Z" "$_sess"
    else
        printf '  %s%s    %s\n' "$C_DIM" "$(t connecter)" "$C_Z" "$(t pas_session "$_sess")"
    fi
    printf '  %s%s    %s\n' "$C_DIM" "$(t journaux)" "$C_Z" "$LOGDIR"
    printf '  %s%s    ./run.sh --status\n' "$C_DIM" "$(t etat)" "$C_Z"
    printf '  %s%s    ./run.sh --stop\n' "$C_DIM" "$(t arreter)" "$C_Z"
    printf '\n'
    if [ $TTY -eq 1 ] && [ "$NO_ATTACH" != 1 ] && [ -z "${TMUX:-}" ] \
        && tmux has-session -t "$_sess" 2>/dev/null; then
        printf '  %sconnexion a la session... (Ctrl-b d pour detacher)%s\n\n' "$C_DIM" "$C_Z"
        exec tmux attach -t "$_sess"
    fi
fi

[ $nb_fail -gt 0 ] && exit 1
exit 0

```

0.3.4 /opt/GSM/qosmo-overlay/tout-en-un.py

13747 octets, 283 lignes → 283 lignes

```
#!/usr/bin/env python3
"""tout-en-un.py - rassemble les .log .sh .md .mmd .py .txt d'un ou plusieurs
dossiers dans UN fichier Markdown, raccourci SANS PERTE :

    * fichiers identiques (meme contenu) : une seule copie, les autres renvoient
      a la premiere ;
    * lignes consecutives identiques : « ligne  xN » ;
    * lignes consecutives qui ne different que par des nombres (horodatage,
      compteur, fn...) : un gabarit ou les nombres constants restent en place et
      les nombres variables deviennent 1 2 ..., suivi de la liste ordonnee des
      valeurs. On reconstruit chaque ligne en remettant les valeurs dans l'ordre.
    * codes couleur ANSI et retours chariot retires (seule perte, volontaire).

    ./tout-en-un.py [dossier...]          default : /opt/GSM/qosmo-overlay et le dernier /root/qosmo
    SORTIE=/chemin.md EXT="log sh md mmd py txt" SEUIL=3 (taille mini d'un groupe)
"""
import glob
import hashlib
import os
import re
import sys
import time

EXT = os.environ.get("EXT", "log sh md mmd py txt").split()
SEUIL = int(os.environ.get("SEUIL", "3"))
FORMAT = os.environ.get("FORMAT", "qmd") # qmd (Quarto) ou md
SORTIE = os.environ.get("SORTIE") or "/root/qosmo-overlay-%s.%s" % (time.strftime("%Y%m%d-%H%M%S"), "md")
AUTEUR = os.environ.get("AUTEUR", "Banc GSM émulé - banc-max")
IGNORES = {".git", "__pycache__", ".pytest_cache", "node_modules"}
ANSI = re.compile(r"\x1b\[([0-9;]*)[A-Za-z]")
TELNET = re.compile(rb"\xff[\xfb-\xfe].|\xff.") # negociation telnet (IAC) des cap
CONTROLE = re.compile(r"[\x00-\x08\x0b\x0c\x0e-\x1f\x7f\ufffd]") # caracteres de commande, o
NUM = re.compile(r"\d+")
LANG = {"sh": "bash", "py": "python", "md": "markdown", "mmd": "mermaid", "txt": "text", "log": "log"}

def dossiers_par_defaut():
    runs = sorted(glob.glob("/root/qosmo-overlay-*"), key=os.path.getmtime)
    runs = [r for r in runs if os.path.isdir(r)]
```

```

return ["/opt/GSM/qosmo-overlay"] + ([runs[-1]] if runs else [])

def fichiers(dossier):
    for racine, dirs, noms in os.walk(dossier):
        dirs[:] = sorted(d for d in dirs if d not in IGNORES)
        for n in sorted(noms):
            if n.startswith("tout-en-un-"):      # nos propres sorties
                continue
            if n.rsplit(".", 1)[-1].lower() in EXT and "." in n:
                yield os.path.join(racine, n)

def compacter(lignes):
    """Rend (lignes de sortie, nb de groupes compactes)."""
    out, groupes, i, n = [], 0, 0, len(lignes)
    while i < n:
        cle = NUM.sub("\0", lignes[i])
        j = i + 1
        while j < n and NUM.sub("\0", lignes[j]) == cle:
            j += 1
        taille = j - i
        if taille < SEUIL:
            out.extend(lignes[i:j])
            i = j
            continue
        groupes += 1
        if "\0" not in cle:      # repetition exacte
            out.append("%s  x%d" % (lignes[i], taille))
            i = j
            continue
        valeurs = [NUM.findall(l) for l in lignes[i:j]]
        nb = len(valeurs[0])
        varie = [len({v[k] for v in valeurs}) > 1 for k in range(nb)]
        # gabarit : nombres constants remis en place, variables numerotes k
        morceaux, k, idx = cle.split("\0"), 0, 0
        gab = morceaux[0]
        for m in morceaux[1:]:
            if varie[k]:
                idx += 1
                gab += " %d " % idx
            else:

```

```

        gab += valeurs[0][k]
    gab += m
    k += 1
out.append("%s  x%d" % (gab, taille))
if idx:
    tuples = [",".join(v[k] for k in range(nb) if varie[k]) for v in valeurs]
    # lignes de valeurs de ~100 colonnes
    ligne, courant = [], ""
    for t in tuples:
        if len(courant) + len(t) + 3 > 110:
            ligne.append(courant)
            courant = ""
            courant += " (" + t + ")"
        ligne.append(courant)
    out.extend(ligne)
    i = j
return out, groupes

def synthese(tous):
    """Run, echelle, echecs, bilan de couverture : lus dans verdict.txt / couverture.txt."""
    lignes, vus = [], set()
    for f in tous:
        nom = os.path.basename(f)
        if nom not in ("verdict.txt", "couverture.txt"):
            continue
        run = os.path.basename(os.path.dirname(f))
        if (run, nom) in vus:
            # meme run copie a deux endroits
            continue
        vus.add((run, nom))
        try:
            txt = ANSI.sub("", open(f, "rb").read().decode("utf-8", "replace"))
        except OSError:
            continue
        for l in txt.splitlines():
            l = re.sub(r"\s+", " ", l.strip())
            if l.startswith("ELEMENT MAX"):
                lignes.append((0, "| %s | échelle | %s |" % (run, l)))
            elif l.startswith("bilan :"):
                lignes.append((2, "| %s | couverture | %s |" % (run, l[8:])))
            elif l.startswith("couverture couche 1"):
                lignes.append((1, "| %s | mode, date | %s |" % (run, l.split("-", 1)[-1].strip())))

```

```

        else:
            m = re.match(r"(\d+) (\S+) (ECHEC|SAUTE) ?(.*)", l)
            if m:
                lignes.append((3, "| %s | barreau %s %s | %s %s |" % (run, m.group(1), m.group(2), m.group(3), m.group(4))))
    return [l for _, l in sorted(lignes, key=lambda x: x[0])]

MERMAID_OUVRE = "```{mermaid}" if FORMAT == "qmd" else "```mermaid"

def rendre_mmd(lignes):
    return "%s\n%s\n```" % (MERMAID_OUVRE, "\n".join(lignes))

def rendre_md(lignes):
    out, dans_bloc, cloture = [], False, ""
    i = 0
    if lignes and lignes[0].strip() == "---": # front matter YAML : montre en bloc
        j = next((k for k in range(1, len(lignes)) if lignes[k].strip() in ("---", "...")), len(lignes))
        if j:
            out.append("```yaml"); out.extend(lignes[1:j]); out.append("```")
            i = j + 1
    while i < len(lignes):
        l = lignes[i]; i += 1
        m = re.match(r"^(\\s*)(`{3,}|~{3,})(.*)$", l)
        if m and not dans_bloc:
            dans_bloc, cloture = True, m.group(2)
            info = m.group(3).strip()
            if info.startswith("{mermaid}") or info.startswith("mermaid"):
                l = m.group(1) + MERMAID_OUVRE
            elif info.startswith("{"): # ````{r ...} : jamais execute
                l = m.group(1) + m.group(2) + info.strip("{").split(" ")[0].split(",")[0]
            out.append(l); continue
        if m and dans_bloc and m.group(2)[0] == cloture[0] and len(m.group(2)) >= len(cloture):
            dans_bloc = False; out.append(l); continue
        if not dans_bloc:
            h = re.match(r"^(#{1,6})\\s", l)
            if h:
                l = "#" * min(6, len(h.group(1))) + 3 + l[len(h.group(1)):]
            elif l.strip() in ("---", "..."): # pas de bloc YAML ni de regle au r
                l = "* * *"
        out.append(l)

```

```

if dans_bloc:
    out.append(cloture)
return "\n".join(out)

def main():
    dossiers = sys.argv[1:] or dossiers_par_defaut()
    vus, sections, table = {}, [], []
    total_in = total_out = 0
    # Ordre : les resultats des tests (.txt .md .mmd : verdicts, couverture, rapports, graphes)
    # d'abord, les fichiers du banc (.sh .py) ensuite, les .log en dernier
    CATEGORIES = (("Verdict et couverture", ()), ("Resultats des tests", ("txt", "md", "mmd")),
                  ("Fichiers", ("sh", "py")), ("Logs", ("log",)))
    EN_TETE = ("verdict.txt", "couverture.txt") # tout en haut, dans cet ordre
    tous = []
    for d in dossiers:
        if not os.path.isdir(d):
            table.append("| (absent) %s | | | |" % d)
            continue
        tous.extend(fichiers(d))
    def rang(f):
        if os.path.basename(f) in EN_TETE:
            return 0
        e = f.rsplit(".", 1)[-1].lower()
        for i, (_, exts) in enumerate(CATEGORIES):
            if e in exts:
                return i
        return len(CATEGORIES)
    categorie_courante = None
    def cle(f):
        r = rang(f)
        return (r, EN_TETE.index(os.path.basename(f)) if r == 0 else 0, tous.index(f))
    for f in sorted(tous, key=cle):
        if rang(f) != categorie_courante:
            categorie_courante = rang(f)
            nom = CATEGORIES[categorie_courante][0] if categorie_courante < len(CATEGORIES) else ""
            sections.append("## %s\n" % nom)
        if True:
            try:
                brut = open(f, "rb").read()
            except OSError as e:
                table.append("| %s | | | | illisible : %s |" % (f, e))

```

```

        continue
    texte = ANSI.sub("", TELNET.sub(b"", brut).decode("utf-8", "replace")).replace("\n", "\r")
    texte = CONTROLE.sub("", texte)
    lignes = texte.split("\n")
    if lignes and lignes[-1] == "":
        lignes.pop()
    h = hashlib.sha1(texte.encode()).hexdigest()
    total_in += len(brut)
    if h in vus:
        sections.append("### %s\n\nidentique à %s\n" % (f, vus[h]))
        continue
    vus[h] = f
    ext = f.rsplit(".", 1)[-1].lower()
    if ext in ("md", "mmd"):
        # Rendu, pas cite : les .mmd deviennent des diagrammes Mermaid, les .md
        # sont inclus tels quels (titres retrogrades sous le titre du fichier,
        # blocs mermaid rendus, blocs {r}/{python} neutralises). Pas de compactage.
        total_out += len(texte) + 1
        sections.append("### %s\n\n%d octets, %d lignes\n\n%s\n"
                        % (f, len(brut), len(lignes), rendre_md(lignes) if ext == "md" else texte))
        continue
    compact, groupes = compacter(lignes)
    total_out += sum(len(l) + 1 for l in compact)
    note = "%d groupes compactés" % groupes if groupes else ""
    # cloture plus longue que toute suite de ` du contenu : un ``` dans un
    # .py ou un .md ne referme plus le bloc (pandoc lisait alors la suite
    # comme du Markdown : « Could not fetch resource »)
    plus_long = max((len(m) for m in re.findall(r"``+", "\n".join(compact))), default=0)
    cloture = "`" * max(3, plus_long + 1)
    sections.append("### %s\n\n%d octets, %d lignes → %d lignes%s\n\n%s\n\n%s\n\n"
                    % (f, len(brut), len(lignes), len(compact), (" (%s)" % note) if note else "",
                       cloture, LANG.get(ext, "text"), "\n".join(compact), cloture))
runs = sorted({os.path.basename(os.path.dirname(f)) for f in tous if os.path.basename(f) != ""})
with open(SORTIE, "w") as o:
    o.write("---\n")
    o.write("title: \"Banc GSM émulé - dossier de run\"\n")
    o.write("subtitle: \"%s\"\n" % (" ".join(runs) if runs else "", ".join(dossiers)))
    o.write("author: \"%s\"\n" % AUTEUR)
    o.write("date: \"%s\"\n" % time.strftime("%Y-%m-%d %H:%M"))
    o.write("lang: fr\n")
    if FORMAT == "qmd":
        o.write("engine: markdown\n") # pandoc seul : knitr voyait des ```{r} dans l

```

```

if FORMAT == "qmd":
    o.write("format:\n  html:\n    toc: true\n    toc-depth: 3\n    toc-location: le
          "    number-sections: true\n    embed-resources: true\n    theme: cosmo\
          "    code-overflow: wrap\n    fontsize: 0.9em\n")
o.write("---\n\n")
o.write("## Synthèse\n\n")
o.write("| run | élément | valeur |\n|---|---|---|\n")
lignes_syn = synthese(tous)
o.write("\n".join(lignes_syn) + "\n\n" if lignes_syn else "| (pas de verdict.txt) |
o.write("| dossier de run | contenu | |\n|---|---|---|\n")
o.write("| Verdict et couverture | échelle des barreaux et tableau de couverture cou
o.write("| Résultats des tests | captures VTY, diagnostics, rapports pytest, grafcet
o.write("| Fichiers | scripts du banc et des tests (.sh .py) | |\n")
o.write("| Logs | journaux (.log), en dernier | |\n\n")
o.write("::: {.callout-note collapse=\"true\"}\n## Méthode\n\n" if FORMAT == "qmd" e
o.write("Dossiers : %s. Extensions : %s. Entrée %d Ko, sortie %d Ko.\n\n"
      % ("", ".join(dossiers), " ".join(EXT), total_in // 1024, total_out // 1024))
o.write("Compactage sans perte : fichiers identiques cités une fois ; lignes consécut
      "« xN » ; lignes ne différant que par des nombres : gabarit avec k puis la
      "des valeurs (k-uplets), chaque ligne se reconstruit en remettant les valeurs
      "Seuls les codes couleur, la négociation telnet des captures VTY et les caract
      "sont retirés. Les .md sont inclus tels quels (titres rétrogradés) et les .mm
      "diagrammes Mermaid, sans compactage.\n")
o.write(":::\n\n" if FORMAT == "qmd" else "\n")
for t in table:
    # ne reste que les dossiers absents / fichiers :
    o.write(t.strip("| ").split(" |")[0] + "\n")
o.write("\n")
o.write("\n".join(sections))
print("%s : %d fichiers (%d uniques), %d Ko -> %d Ko" % (SORTIE, len(tous), len(vus), to
      os.path.getsize(SORTIE) // 1024))

if __name__ == "__main__":
    main()

```

0.3.5 /opt/GSM/qosmo-overlay/opt-gsm-scripts/gsm_sniff.py

5519 octets, 129 lignes → 129 lignes

```

#!/usr/bin/env python3
# gsm_sniff.py - sniff PASSIF (raw socket) des sockets UDP GSMTAP/SCH/TRXD et

```

```

# TRANSFORME en SI decode / burst decode. NE bind AUCUN port (AF_PACKET) -> ne
# vole rien aux vrais destinataires (grgsm/qemu), ne touche AUCUNE fifo.
#
# Usage : gsm_sniff.py si      # GSMTAP -> num+type SI, canal BCCH/CCCH, FN (4729,4730)
#         gsm_sniff.py burst   # SCH (4731) + TRXD v1 (5700-5702/5800-5802) -> burst
import socket, struct, sys, os

MODE = sys.argv[1] if len(sys.argv) > 1 else "si"
IFACE = os.environ.get("SNIFF_IFACE", "lo")
ETH_P_IP = 0x0800
GSMTAP_HDR_LEN = 16                                # GSMTAP v2 header (octets)

# --- GSM 04.08 / 44.018 RR message types (PD = 0x06), 10.4 -----
# key = RR message-type octet (L2[2]) ; value = (nom, is_si, canal DL attendu)
RR_MT = {
    0x19: ("SI1",      True,  "BCCH"),
    0x1a: ("SI2",      True,  "BCCH"),
    0x1b: ("SI3",      True,  "BCCH"),
    0x1c: ("SI4",      True,  "BCCH"),
    0x1d: ("SI2bis",   True,  "BCCH"),
    0x1e: ("SI2ter",   True,  "BCCH"),
    0x02: ("SI2quater", True,  "BCCH"),
    0x03: ("SI9",      True,  "BCCH"),
    0x00: ("SI13",     True,  "CCCH"), # SI13 = mt 0x00 (44.018 tbl 10.4.1)
    # --- CCCH downlink non-SI ---
    0x3f: ("IMM-ASSIGN", False, "CCCH"), # Immediate Assignment (AGCH) <- le LU !
    0x39: ("IMM-ASSIGN-EXT", False, "CCCH"),
    0x3a: ("IMM-ASSIGN-REJ", False, "CCCH"),
    0x21: ("PAGING-REQ-1", False, "CCCH"), # 0x21 = PAGING type 1 (PAS SI13)
    0x22: ("PAGING-REQ-2", False, "CCCH"),
    0x24: ("PAGING-REQ-3", False, "CCCH"),
    0x06: ("CHAN-RELEASE", False, "*"),
    0x2e: ("ASSIGN-CMD", False, "*"),
    0x10: ("CIPHER-MODE-CMD", False, "*"),
    0x12: ("RR-STATUS", False, "*"),
}

GSMTAP_CHAN = {0x00:"UNKNOWN",0x01:"BCCH",0x02:"CCCH",0x03:"RACH",0x04:"AGCH",
               0x05:"PCH",0x06:"SDCCH",0x07:"SDCCH4",0x08:"SDCCH8",0x09:"TCH/F",
               0x0a:"TCH/H",0x0b:"PACCH",0x0c:"CBCH52",0x0d:"PDTCH",0x0e:"PTCCH",0x0f:"CBCH5
def gsmtap_chan_name(st):
    base = GSMTAP_CHAN.get(st & 0x7f, "0x%02x" % (st & 0x7f))

```

```

    return base + ("ACCH" if (st & 0x80) else "")
def gsmtap_chan_class(st):
    b = st & 0x7f
    if b == 0x01: return "BCCH"
    if b in (0x02,0x04,0x05): return "CCCH"
    if b == 0x03: return "RACH"
    if b in (0x06,0x07,0x08): return "SDCCH"
    return "OTHER"

def fn51_role(fn):
    m = fn % 51
    if m in (0,10,20,30,40): return "FCCH"
    if m in (1,11,21,31,41): return "SCH"
    if m in (2,3,4,5): return "BCCH"
    if m == 50: return "idle"
    return "CCCH"
def fn51_class(fn):
    r = fn51_role(fn)
    return r if r in ("BCCH","CCCH") else "-"

if MODE == "si":
    PORTS = {int(p) for p in os.environ.get("SNIFF_PORTS","4729,4730").split(",")}
else:
    PORTS = {int(p) for p in os.environ.get("SNIFF_PORTS","4731,5700,5701,5702,5800,5801,5802").split(",")}

s = socket.socket(socket.AF_PACKET, socket.SOCK_DGRAM, socket.htons(ETH_P_IP))
s.bind((IFACE, 0))
print("[gsm-sniff:%s] iface=%s ports=%s (passif, aucune fifo)"
      % (MODE, IFACE, sorted(PORTS)), flush=True)

def udp_payload(ip):
    if len(ip) < 20 or (ip[0] >> 4) != 4: return None, None
    ihl = (ip[0] & 0x0f) * 4
    if ip[9] != 17: return None, None
    u = ip[ihl:]
    if len(u) < 8: return None, None
    return struct.unpack(">H", u[2:4])[0], u[8:]

def decode_si(pl):
    if len(pl) < GSMTAP_HDR_LEN: return None
    ts = pl[3]
    fn = struct.unpack(">I", pl[8:12])[0] # frame_number reel (BE @ offset 8)

```

```

sub = pl[12] # GSMTAP sub_type / canal
l2 = pl[16:]
if len(l2) >= 3 and l2[1] == 0x06:
    mt = l2[2]
    name, is_si, exp = RR_MT.get(mt, ("RR/mt=0x%02x" % mt, False, "*"))
    name = "%s(0x%02x)" % (name, mt)
else:
    name, is_si, exp = "?", False, "*"
tag_cls = gsmtap_chan_class(sub)
fn_role = fn51_role(fn); fn_cls = fn51_class(fn)
flags = []
if tag_cls != fn_cls and fn_cls != "-": flags.append("CHAN!=FN(%s/%s)" % (tag_cls, fn_cls))
if fn_role in ("FCCH", "SCH", "idle"): flags.append("BAD-FN(sur %s)" % fn_role)
if is_si and exp == "BCCH" and fn_cls == "CCCH": flags.append("SI-sur-CCCH")
fl = (" [" + " ".join(flags) + "]") if flags else ""
return "FN=%d ts%d (%51=%d %s) CHAN=%-12s %-18s L2: %s%s" % (
    fn, ts, fn % 51, fn_role, gsmtap_chan_name(sub), name, l2[:23].hex(" "), fl)

def decode_burst(dport, pl):
    if dport == 4731 and len(pl) >= 16 and pl[:4] == b"SCH1":
        bsic, fn, toa = struct.unpack("<iii", pl[4:16])
        return "SCH bsic=%d (ncc=%d bcc=%d) fn=%d (%51=%d %s) toa=%d" % (
            bsic, (bsic>>3)&7, bsic&7, fn, fn%51, fn51_role(fn), toa)
    if dport in (5700, 5701, 5702, 5800, 5801, 5802) and len(pl) >= 6:
        ver = pl[0] >> 4; tn = pl[0] & 0x07
        fn = int.from_bytes(pl[1:5], "big")
        if len(pl) >= 6 + 148:
            b = "".join("1" if x else "0" for x in pl[-148:])
            return "TRXDv%d ts%d fn=%d (%51=%d %s) 148b %s" % (ver, tn, fn, fn%51, fn51_role(fn), b)
            return "TRXDv%d ts%d fn=%d (%51=%d %s) NOPE(len=%d, burst vide)" % (
                ver, tn, fn, fn%51, fn51_role(fn), len(pl))
    return None

n = 0
while True:
    pkt = s.recv(65535)
    dport, pl = udp_payload(pkt)
    if dport not in PORTS or pl is None:
        continue
    line = decode_si(pl) if MODE == "si" else decode_burst(dport, pl)
    if line:
        n += 1

```

```
print("%6d  :%-4d  %s" % (n, dport, line), flush=True)
```

0.3.6 /opt/GSM/qosmo-overlay/run_modules/00-prereqs.sh

862 octets, 29 lignes → 29 lignes

```
MOD_REGISTER prereqs "Vérification des prérequis"
MOD_REQUIRED[prereqs]=1
MOD_PURE[prereqs]=1

mod_prereqs_check() {
    local missing="" v p
    for v in QEMU_BIN FIRMWARE_ELF FIRMWARE_BIN OSMOCON; do
        p="${!v:-}"
        [ -n "$p" ] && [ -e "$p" ] || missing="$missing $v"
    done
    if [ -n "$missing" ]; then
        mod_hint "réglez les chemins dans environnement/bench.env, puis ./run.sh --check-path"
        mod_fail "dépendances introuvables :$missing"
        return $MOD_RC_FAIL
    fi
    mod_ok
}

mod_prereqs_start() {
    mod_say "arbre          : ${QEMU_TREE}"
    mod_say "binaire QEMU      : ${QEMU_BIN}"
    mod_say "firmware          : ${FIRMWARE_ELF}"
    mod_say "osmocon           : ${OSMOCON}"
    mod_say "exécution         : ${RUN_DIR}"
    mod_say "journaux          : ${LOG_DIR}"
    mod_ok
}

mod_prereqs_status() { return $MOD_RC_FAIL; }
```

0.3.7 /opt/GSM/qosmo-overlay/run_modules/02-outils.sh

2151 octets, 62 lignes → 62 lignes

```

MOD_REGISTER outils "Outillage de l'hôte"
MOD_REQUIRED[outils]=1
MOD_PURE[outils]=1
MOD_TIMEOUT[outils]=10

: "${OUTILS_REQUIS:=bash sleep pgrep pkill kill stat grep sed awk ip mkfifo rm}"

if ! declare -p OUTILS_OPTIONNELS >/dev/null 2>&1; then
    OUTILS_OPTIONNELS=(
        "ss|sonde des ports d'écoute (repli : /dev/tcp, sans nom de tenant)"
        "tmux|fenêtres de la pile et épilogue « se connecter »"
        "setsid|détachement des processus lancés en arrière-plan"
        "flock|verrou d'unicité du démon audio"
        "socat|passerelles série et sockets de test"
        "nc|alimentation du HLR (repli telnet : fermeture différente, échec silencieux)"
        "python3|outils d'analyse, gr-gsm, transceiver de test"
    )
fi

mod_outils_check() {
    local c manquants=""
    for c in $OUTILS_REQUIS; do
        command -v "$c" >/dev/null 2>&1 || manquants="$manquants $c"
    done
    if [ -n "$manquants" ]; then
        mod_hint "installez-les : apt-get install -y coreutils procps iproute2 grep sed gawk"
        mod_fail "commandes indispensables absentes du PATH :$manquants"
        return $MOD_RC_FAIL
    fi
    mod_ok
}

mod_outils_status() { return $MOD_RC_FAIL; }

mod_outils_start() {
    local c n d
    for c in $OUTILS_REQUIS; do
        mod_say "requis      $c -> $(command -v "$c" 2>/dev/null)"
    done
    for c in "${OUTILS_OPTIONNELS[@]}; do
        n="${c%%|*}"; d="${c#*|}"
        if command -v "$n" >/dev/null 2>&1; then

```

```

        mod_say "optionnel  $n -> $(command -v "$n")"
    else
        mod_say "ABSENT      $n - dégrade : $d"
    fi
done
mod_ok
}

_outils_enfant_resout() {
    bash -c 'command -v ip >/dev/null 2>&1 && command -v pgrep >/dev/null 2>&1 && command -v'
}

mod_outils_wait() {
    if ! wait_until "${MOD_TIMEOUT[outils]}" "outillage visible d'un enfant" _outils_enfant_
        mod_hint "PATH courant : $PATH - exportez-le, ou donnez les chemins absolus dans env"
        mod_fail "un shell enfant ne résout pas l'outillage (PATH non hérité)"
        return $MOD_RC_FAIL
    fi
    mod_ok
}

```

0.3.8 /opt/GSM/qosmo-overlay/run_modules/03-rundir.sh

2239 octets, 66 lignes → 66 lignes

```

MOD_REGISTER rundir "Répertoire d'exécution"
MOD_REQUIRED[rundir]=1
MOD_DEPS[rundir]="profil"
MOD_TIMEOUT[rundir]=15

mod_rundir_check() {
    local d="${RUN_DIR:-}" parent
    [ -n "$d" ] || { mod_hint "RUN_DIR est posé par environnement/paths.env"
        mod_fail "RUN_DIR non défini"
        return $MOD_RC_FAIL; }
    if [ -e "$d" ] && [ ! -d "$d" ]; then
        mod_hint "rm -f $d (ou choisissez un autre RUN_DIR)"
        mod_fail "$d existe et n'est pas un répertoire"
        return $MOD_RC_FAIL
    fi
    if [ ! -d "$d" ]; then

```

```

    parent="$(dirname "$d")"
    while [ ! -d "$parent" ] && [ "$parent" != "/" ]; do parent="$(dirname "$parent)";
    if [ ! -w "$parent" ]; then
        mod_hint "choisissez un emplacement inscriptible : RUN_DIR=\$HOME/calypso ./star
        mod_fail "impossible de créer $d : $parent n'est pas inscriptible"
        return $MOD_RC_FAIL
    fi
fi
mod_ok
}

mod_rundir_status() { [ -d "${RUN_DIR:-}" ] && [ -w "${RUN_DIR:-}" ]; }

mod_rundir_start() {
    mkdir -p "${RUN_DIR}" 2>/dev/null || { mod_fail "création impossible : ${RUN_DIR}"; return
    chmod 0755 "${RUN_DIR}" 2>/dev/null
    mod_say "répertoire d'exécution : ${RUN_DIR}"
    local f n=0
    for f in "${RUN_DIR}"/*.pid "${RUN_DIR}"/*.sock; do
        [ -e "$f" ] && { mod_say "reste du run précédent : $f"; n=$((n+1)); }
    done
    [ "$n" -gt 0 ] && mod_say "$n reste(s) - 04-restes puis 10-teardown s'en chargent"
    mod_ok
}

_rundir_ecriture_ok() {
    local t="${RUN_DIR}/.temoin.$$"
    : > "$t" 2>/dev/null || return 1
    rm -f "$t" 2>/dev/null
    return 0
}

mod_rundir_wait() {
    if ! wait_until "${MOD_TIMEOUT[rundir]}" "écriture dans ${RUN_DIR}" _rundir_ecriture_ok;
        mod_hint "montage en lecture seule ou volume plein : df -h ${RUN_DIR}"
        mod_fail "${RUN_DIR} existe mais l'écriture y échoue"
        return $MOD_RC_FAIL
    fi
    mod_ok
}

mod_rundir_stop() {

```

```

local f pid
for f in "${RUN_DIR:-/nonexistent}"/*.pid; do
    [ -f "$f" ] || continue
    pid="$(cat "$f" 2>/dev/null || echo 0)"
    kill -0 "$pid" 2>/dev/null || rm -f "$f"
done
return 0
}

```

0.3.9 /opt/GSM/qosmo-overlay/run_modules/04-restes.sh

2518 octets, 82 lignes → 82 lignes

```

MOD_REGISTER restes "Extinction des processus enregistrés"
MOD_REQUIRED[restes]=0
MOD_DEPS[restes]="rundir"
MOD_TIMEOUT[restes]=20
MOD_ENABLED_IF[restes]='[ "${NO_STARTUP_STOP:-0}" != 1 ]'

_res_pids() {
    local f pid
    for f in "${RUN_DIR:-/nonexistent}"/*.pid; do
        [ -f "$f" ] || continue
        pid="$(cat "$f" 2>/dev/null || echo 0)"
        [ "$pid" -gt 0 ] 2>/dev/null || continue
        [ "$pid" = "$$" ] && continue
        kill -0 "$pid" 2>/dev/null && printf '%s\n' "$pid"
    done
}

_res_vide() { [ -z "$(_res_pids)" ]; }

_res_cmdline() { tr '\0' ' ' < "/proc/$1/cmdline" 2>/dev/null | cut -c1-140; }

_res_signal() {
    local sig="$1" p
    for p in $(_res_pids); do
        kill "-$sig" -- "-$p" 2>/dev/null || kill "-$sig" "$p" 2>/dev/null
    done
    return 0
}

```

```

_res_menage() {
    local f pid
    for f in "${RUN_DIR:-/nonexistent}"/*.pid; do
        [ -f "$f" ] || continue
        pid="$(cat "$f" 2>/dev/null || echo 0)"
        kill -0 "$pid" 2>/dev/null || { rm -f "$f"; mod_say "fichier PID retiré : $f"; }
    done
    return 0
}

mod_restes_check() {
    command -v kill >/dev/null 2>&1 || { mod_fail "commande kill introuvable"; return $MOD_RC }
    mod_ok
}

mod_restes_status() { _res_vide; }

mod_restes_start() {
    local liste p
    liste="$_res_pids | tr '\n' ' '"
    if [ -z "${liste// /}" ]; then
        _res_menage
        mod_already "aucun processus enregistré n'a survécu au run précédent"
        return $MOD_RC_ALREADY
    fi
    for p in $liste; do mod_say "encore vivant : $p $_res_cmdline "$p""; done
    _res_signal TERM
    mod_ok
}

mod_restes_wait() {
    local moitie=$(( ${MOD_TIMEOUT[restes]} / 2 ))
    [ "$moitie" -lt 2 ] && moitie=2

    if ! wait_until "$moitie" "extinction douce" _res_vide; then
        mod_say "toujours vivants après TERM : $_res_pids | tr '\n' ' ' - passage en KILL"
        _res_signal KILL
        if ! wait_until "$moitie" "extinction forcée" _res_vide; then
            mod_hint "identifiez le tenace : ps -o pid,ppid,stat,cmd -p \$(cat ${RUN_DIR:-/}
            mod_fail "processus enregistrés impossibles à arrêter : $_res_pids | tr '\n' '
            return $MOD_RC_FAIL
        fi
    fi
}

```

```

    fi
    _res_menage
    mod_ok
}

mod_restes_stop() {
    _res_vide && { _res_menage; return 0; }
    _res_signal TERM
    wait_until 5 "extinction douce" _res_vide || { _res_signal KILL; wait_until 5 "extinction
    _res_menage
    return 0
}

```

0.3.10 /opt/GSM/qosmo-overlay/run_modules/05-config.sh

2431 octets, 64 lignes → 64 lignes

```

MOD_REGISTER config "Résolution des chemins et des ports"
MOD_REQUIRED[config]=1
MOD_DEPS[config]="prereqs"
MOD_PURE[config]=1
MOD_TIMEOUT[config]=5

mod_config_check() {
    if [ ! -d "${QEMU_TREE:-}/run_modules" ]; then
        mod_hint "posez QEMU_TREE sur la racine du dépôt, ou lancez ./run.sh depuis cette ra
        mod_fail "QEMU_TREE ne désigne pas ce dépôt : ${QEMU_TREE:-<non défini>}"
        return $MOD_RC_FAIL
    fi
    mod_ok
}

mod_config_status() { return $MOD_RC_FAIL; }

mod_config_start() {
    : "${BTS_CFG:=${OSMOCOM_CFG:-/etc/osmocom}/osmo-bts-trx.cfg}"
    : "${MOBILE_CFG_SRC:=${QEMU_CFGS:-$QEMU_TREE/cfgs}/mobile_group1.cfg}"
    : "${MOBILE_CFG:=${OSMOCOM_HOME:-$HOME/.osmocom}/bb/mobile_group1.cfg}"
    : "${QEMU_MON_SOCKET:=${RUN_DIR:-/tmp/calypso}/qemu-monitor.sock}"
    : "${L1CTL_SOCKET_PATH:=/tmp/osmocom_l2}"
    : "${TMUX_SESSION:=calypso}"
}

```

```

: "${PORT_TRXD_CLOCK:=5700}"
: "${PORT_TRXD_CTRL:=5701}"
: "${PORT_TRXD_DATA:=5702}"
: "${PORT_GSM_TAP:=4729}"
: "${PORT_GSM_TAP_SI:=4730}"
: "${PORT_GSM_TAP_SCH:=4731}"
mod_say "BTS_CFG          = $BTS_CFG"
mod_say "MOBILE_CFG_SRC   = $MOBILE_CFG_SRC"
mod_say "MOBILE_CFG       = $MOBILE_CFG"
mod_say "QEMU_MON SOCK    = $QEMU_MON SOCK"
mod_say "L1CTL SOCK_PATH  = $L1CTL SOCK_PATH (non exportée)"
mod_say "ports           : TRXD $PORT_TRXD_CLOCK-$PORT_TRXD_DATA (pont) · GSM_TAP $PORT_
local f
for f in "$BTS_CFG" "$MOBILE_CFG_SRC"; do
    [ -r "$f" ] && mod_say "lisible : $f" || mod_say "ABSENT : $f (le module qui l'uti
done
mod_ok
}

_config_vars() {
    printf '%s\n' BTS_CFG MOBILE_CFG MOBILE_CFG_SRC QEMU_MON SOCK L1CTL SOCK_PATH TMUX_SESSI
        PORT_TRXD_CLOCK PORT_GSM_TAP PORT_GSM_TAP_SI PORT_GSM_TAP_SCH
}

_config_missing() {
    local v out=""
    while read -r v; do [ -n "${!v:-}" ] || out="$out $v"; done <<(_config_vars)
    printf '%s' "${out# }"
}

_config_all_set() { [ -z "$(_config_missing)" ]; }

mod_config_wait() {
    if ! wait_until "${MOD_TIMEOUT[config]}" "variables de configuration" _config_all_set; t
        mod_hint "vérifiez environnement/bench.env (QEMU_TREE, OSMOCOM_CFG, OSMOCOM_HOME, RU
        mod_fail "variables de configuration vides : $_config_missing)"
        return $MOD_RC_FAIL
    fi
    mod_ok
}

mod_config_stop() { return 0; }

```

0.3.11 /opt/GSM/qosmo-overlay/run_modules/08-gabarits.sh

5374 octets, 130 lignes → 130 lignes

```
MOD_REGISTER gabarits "Gabarits de configuration du cœur"
MOD_REQUIRED[gabarits]=0
MOD_DEPS[gabarits]="config"
MOD_TIMEOUT[gabarits]=20

MOD_ENABLED_IF[gabarits]='[ "${NO_OSMO_START:-0}" != 1 ] && [ "${NO_GABARITS:-0}" != 1 ]'

: "${EGPRS_DIR:=${NITB_ROOT}}"
: "${EGPRS_LIB:=${EGPRS_DIR}/lib/gabarits.sh}"
: "${OPERATOR_ID:=1}"
: "${MCC:=001}"
: "${MNC:=01}"
: "${OPERATOR_NAME:=OsmoDirect}"
: "${ENCRYPTION:=a5 0}"

_gab_installes() {
    printf '%s\n' "${OSMOCOM_CFG:-/etc/osmocom}/osmo-bsc.cfg" \
        "${OSMOCOM_CFG:-/etc/osmocom}/osmo-msc.cfg"
}

mod_gabarits_check() {
    if [ ! -d "$EGPRS_DIR/configs" ]; then
        mod_hint "posez EGPRS_DIR=<racine osmo_egprs> si l'arborescence a bougé"
        mod_skip "osmo_egprs introuvable ($EGPRS_DIR) : les gabarits ne sont pas de ce dépôt"
        return $MOD_RC_SKIP
    fi
    if [ ! -r "$EGPRS_LIB" ]; then
        mod_hint "la bibliothèque est l'extraction de start-direct.sh.legacy L174-258 ; elle"
        mod_fail "bibliothèque de gabarits absente : $EGPRS_LIB"
        return $MOD_RC_FAIL
    fi
    if [ "${id -u}" -ne 0 ]; then
        mod_fail "root requis pour écrire dans ${OSMOCOM_CFG:-/etc/osmocom} et /etc/asterisk"
        return $MOD_RC_FAIL
    fi
    if [ "${N_OPERATORS:-1}" -gt 1 ]; then
        mod_hint "le multi-opérateur pose une configuration par netns ; aucun module du cœur"
        mod_fail "N_OPERATORS=${N_OPERATORS} : ce module ne sait poser que le mono-opérateur"
        return $MOD_RC_FAIL
    fi
}
```

```

fi
mod_ok
}

mod_gabarits_status() { return $MOD_RC_FAIL; }

mod_gabarits_start() {
    local retour="$PWD" rc=0
    cd "$EGPRS_DIR" || { mod_fail "impossible d'entrer dans $EGPRS_DIR"; return $MOD_RC_FAIL; }

    . "$EGPRS_LIB" || { cd "$retour"; mod_fail "bibliothèque illisible : $EGPRS_LIB"; return $MOD_RC_FAIL; }

    local tmpdir; tmpdir="$(mktemp -d)" || { cd "$retour"; mod_fail "mktemp -d a échoué"; return $MOD_RC_FAIL; }

    local cip gw
    cip="$(op_private_ip "$OPERATOR_ID")"
    gw="$(op_private_gw "$OPERATOR_ID")"

    mod_say "opérateur $OPERATOR_ID · MCC/MNC ${MCC}/${MNC} · chiffrement « ${ENCRYPTION} »"
    mod_say "gabarits : $EGPRS_DIR/configs -> ${OSMOCOM_CFG:-/etc/osmocom}, /etc/asterisk, ..."

    local _inter_shutdown="shutdown" _multi="/etc/osmocom/osmo-multi.conf"
    if [ -r "$_multi" ] && grep -q '^MULTI_HUB_IP=' "$_multi" 2>/dev/null; then
        _inter_shutdown=""
        mod_say "inter-STP déclaré (osmo-multi.conf) : ASP as-inter laissé ACTIF"
    fi

    if apply_config_templates "$tmpdir" "$cip" "$gw" "$OPERATOR_ID" \
        "1.1.1" "1.1.2" "1.1.3" "$MCC" "$MNC" "$OPERATOR_NAME" \
        "${INTER_STP_IP:-127.0.0.1}" "$_inter_shutdown" "1"; then
        install_configs_native "$tmpdir" "" || rc=1
        # Les gabarits écrasent pjsip/extensions/rtp/sms-routing : sans WAN dans
        # ces fichiers, aucun appel vers un autre noeud (503 cote SIP). On rejoue
        # le maillage APRES eux, si la table WAN a plusieurs noeuds.
        _wan_rejoue() {
            local t="${WAN_CONF_FILE:-/etc/osmo-wan.conf}"
            local sh="${WAN_MESH_SH:-/opt/GSM/osmo-operator/network/setup-wan-mesh.sh}"
            local n
            [ -r "$t" ] && [ -r "$sh" ] || return 0
            n="$(sed -n 's/^WAN_NODE_COUNT=//p' "$t" | tail -1)"
            [ "${n:-1}" -ge 2 ] 2>/dev/null || return 0
            if WAN_AUTO=1 bash "$sh" --native --config "$t" --no-restart </dev/null >/dev/null; then
                mod_say "WAN rejoue apres les gabarits (table $t, $n noeuds)"
            fi
        }
    fi
}

```

```

        else
            mod_say "WAN NON rejoue apres les gabarits (voir $sh)"
            return 1
        fi
    }
    _wan_rejoue || rc=1
else
    rc=1
fi

rm -rf "$tmpdir"
cd "$retour"
[ $rc -eq 0 ] || { mod_fail "la pose des gabarits a échoué (détail dans le journal du mo
mod_ok
}

mod_gabarits_wait() {
    local f restants=""
    while IFS= read -r f; do
        if [ ! -f "$f" ]; then
            mod_hint "vérifiez que $EGPRS_DIR/configs contient bien le gabarit correspondant
            mod_fail "configuration non installée : $f"
            return $MOD_RC_FAIL
        fi
        if grep -qE '__[A-Z0-9_]+__' "$f" 2>/dev/null; then
            restants="$restants $(basename "$f"):$$(grep -oE '__[A-Z0-9_]+__' "$f" | sort -u
        fi
    done < <(_gab_installes)

    if [ -n "$restants" ]; then
        mod_hint "ces jetons n'ont pas de valeur dans apply_config_templates (lib/gabarits.s
        mod_fail "jetons non substitués :$restants"
        return $MOD_RC_FAIL
    fi

    local cfg
    for cfg in "${OSMOCOM_CFG:-/etc/osmocom}/osmo-bsc.cfg" \
        "${OSMOCOM_CFG:-/etc/osmocom}/osmo-msc.cfg"; do
        if grep -q '.__ENCRIPTION__' "$cfg" 2>/dev/null || ! grep -qF "$ENCRIPTION" "$cfg" 2>
            mod_hint "attendu « $ENCRIPTION » dans $cfg ; vérifiez que le gabarit porte bien
            mod_fail "le chiffrement demandé n'est pas dans la configuration installée ($cfg
            return $MOD_RC_FAIL

```

```

        fi
        mod_say "chiffrement « $ENCRYPTION » confirmé dans $cfg"
    done
    mod_ok
}

mod_gabarits_stop() { return 0; }

```

0.3.12 /opt/GSM/qosmo-overlay/run_modules/09-logs.sh

3037 octets, 81 lignes → 81 lignes

```

MOD_REGISTER logs "Journaux et espace de travail"
MOD_REQUIRED[logs]=1
MOD_DEPS[logs]="config"
MOD_TIMEOUT[logs]=10

: "${LOG_ARCHIVES_KEEP:=10}"
: "${OSMO_LOG_DIR:=/var/log/osmocom}"
: "${OSMO_LOG_USER:=osmocom}"

mod_logs_check() {
    local d="${LOG_DIR:-}" p
    [ -n "$d" ] || { mod_fail "LOG_DIR non défini"; return $MOD_RC_FAIL; }
    p="$d"
    while [ ! -d "$p" ] && [ "$p" != "/" ] && [ "$p" != "." ]; do p="$(dirname "$p)"; done
    if [ ! -w "$p" ]; then
        mod_hint "posez LOG_DIR sur un chemin inscriptible (environnement/bench.env, LOG_DIR)"
        mod_fail "aucun parent inscriptible pour LOG_DIR=$d (bloqué à $p)"
        return $MOD_RC_FAIL
    fi
    mod_ok
}

mod_logs_status() { return $MOD_RC_FAIL; }

mod_logs_start() {
    local _n _cible _lien _faits=0
    if [ "${CALYPSO_LIENS_ROOT:-1}" = 1 ] && [ -w /root ]; then
        for _n in qemu osmocon mobile bts sidecar-mobile; do
            _cible="$LOG_DIR/${_n}.log"; _lien="/root/${_n}.log"

```

```

        if [ -e "$_lien" ] && [ ! -L "$_lien" ]; then
            mod_say " /root/$_n.log est un vrai fichier - laissé en place"
            continue
        fi
        ln -sfn "$_cible" "$_lien" 2>/dev/null && _faits=$(( _faits + 1 ))
    done
    [ "$_faits" -gt 0 ] && mod_say "$_faits liens de compatibilité dans /root (→ $LOG_DIR)
fi
[ -d "$OSMO_LOG_DIR" ] || mkdir -p "$OSMO_LOG_DIR" 2>/dev/null || true
if id "$OSMO_LOG_USER" >/dev/null 2>&1; then
    chown "$OSMO_LOG_USER":"$OSMO_LOG_USER" "$OSMO_LOG_DIR" 2>/dev/null || true
fi
if [ -d "$OSMO_LOG_DIR" ]; then
    mod_say "journaux des services : $OSMO_LOG_DIR (propriétaire $OSMO_LOG_USER)"
else
    mod_say " $OSMO_LOG_DIR absent - les services n'écriront que sur journald"
fi
mkdir -p "$LOG_DIR" "$LOG_DIR/mod" "${RUN_DIR:-/tmp/calypso}" "${CAPTURE_DIR:-$RUN_DIR/capture}"
local osmocon_log="$LOG_DIR/osmocon.log"
if [ -s "$osmocon_log" ]; then
    local arch="${osmocon_log%.log}.${date +%Y%m%d_%H%M%S}.log"
    if mv "$osmocon_log" "$arch" 2>/dev/null; then
        mod_say "osmocon.log du run précédent archivé -> $arch"
    fi
fi
ls -lt "${osmocon_log%.log}."[0-9]*.log 2>/dev/null \
    | tail -n +$((LOG_ARCHIVES_KEEP + 1)) | xargs -r rm -f 2>/dev/null || true
for _n in qemu osmocon mobile bts; do
    : > "$LOG_DIR/$_n.log" 2>/dev/null
done
mod_say "LOG_DIR      = $LOG_DIR"
mod_say "RUN_DIR      = ${RUN_DIR:-/tmp/calypso}"
mod_ok
}

_logs_writable() {
    local probe="$LOG_DIR/.probe.$$"
    : > "$probe" 2>/dev/null || return 1
    rm -f "$probe" 2>/dev/null
    return 0
}

```

```

mod_logs_wait() {
    if ! wait_until "${MOD_TIMEOUT[logs]}" "espace de journaux" _logs_writable; then
        mod_hint "df -h $LOG_DIR - volume plein ou monté en lecture seule ?"
        mod_fail "écriture impossible dans $LOG_DIR (test réel, pas un test de droits)"
        return $MOD_RC_FAIL
    fi
    mod_ok
}

mod_logs_stop() { return 0; }

```

0.3.13 /opt/GSM/qosmo-overlay/run_modules/09-teardown.sh

4618 octets, 147 lignes → 147 lignes

```

MOD_REGISTER teardown "Nettoyage du run précédent"
MOD_REQUIRED[teardown]=1
MOD_DEPS[teardown]="logs"
MOD_TIMEOUT[teardown]=25

_td_patterns() {
    cat <<'EOF'
f:gemu-system-arm
f:gardien-bts
x:osmo-bts-trx
x:osmocon
x:mobile
x:trxcon
f:pont/pont.py
f:pont/pont_dsp.py
f:tools/injecteur_bruit.py
EOF
}

_td_port_busy() {
    ss -ln 2>/dev/null | grep -qE "[:.]\${1}[:,space:]"
}

_td_ports() {
    printf '%s\n' "${PORT_TRXD_CLOCK:-5700}" "${PORT_TRXD_CTRL:-5701}" "${PORT_TRXD_DATA:-5702}"
    "${PORT_GSM_TAP_SI:-4730}" "${PORT_GSM_TAP_SCH:-4731}"
}

```

```

}

_td_sockets() {
    printf '%s\n' "${L1CTL_SOCKET_PATH:-/tmp/osmocom_l2}" \
        "${SC_L2_SOCKET:-/tmp/ms2_l2}" \
        "${QEMU_MON_SOCKET:-${RUN_DIR:-/tmp/calypso}/qemu-monitor.sock}"
}

_td_live() {
    local _p _st
    for _p in $(pgrep "$1" "$2" 2>/dev/null); do
        _st=$(ps -o stat= -p "$_p" 2>/dev/null | tr -d ' ')
        case "$_st" in
            '|Z*) continue ;;
            *) return 0 ;;
        esac
    done
    return 1
}

_td_leftovers() {
    local line kind pat out="" p
    while IFS= read -r line; do
        kind="${line%:*}"; pat="${line#*:}"
        case "$kind" in
            x) _td_live -x "$pat" && out="$out $pat" ;;
            f) _td_live -f "$pat" && out="$out $pat" ;;
        esac
    done <<(_td_patterns)
    while IFS= read -r p; do
        _td_port_busy "$p" && out="$out port:$p"
    done <<(_td_ports)
    while IFS= read -r p; do
        [ -e "$p" ] && out="$out socket:$p"
    done <<(_td_sockets)
    printf '%s' "${out# }"
}

_td_clean() { [ -z "$(_td_leftovers)" ]; }

mod_tearardown_check() {
    local t
    for t in pgrep pkill; do

```

```

        command -v "$t" >/dev/null 2>&1 || {
            mod_hint "installez procps (paquet procps / procps-ng)"
            mod_fail "$t introuvable : impossible de garantir une machine propre"
            return $MOD_RC_FAIL
        }
    done
    command -v ss >/dev/null 2>&1 || mod_say "ss absent : la libération des ports UDP ne sera pas possible"
    mod_ok
}

mod_tearardown_status() { _td_clean; }

_td_kill_all() {
    local line kind pat
    while IFS= read -r line; do
        kind="${line%%:*}"; pat="${line#*:}"
        case "$kind" in
            x) _td_live -x "$pat" && { mod_say "kill -x $pat"; pkill -9 -x "$pat" 2>/dev/null }
            f) _td_live -f "$pat" && { mod_say "kill -f $pat"; pkill -9 -f "$pat" 2>/dev/null }
        esac
    done <<(_td_patterns)
    return 0
}

_td_kill_ports() {
    command -v ss >/dev/null 2>&1 || { mod_say "ss absent : passe par port impossible"; return 1 }
    local p pid cmd busy=0 named=0
    while IFS= read -r p; do
        _td_port_busy "$p" || continue
        busy=1
        for pid in $(ss -lnp "sport = :$p" 2>/dev/null \
            | grep -oE 'pid=[0-9]+' | cut -d= -f2 | sort -u); do
            named=1
            [ "$pid" -gt 1 ] 2>/dev/null || continue
            [ "$pid" = "$$" ] && continue
            cmd=$(tr '\0' ' ' < "/proc/$pid/cmdline" 2>/dev/null)
            mod_say "port $p tenu par PID $pid (${cmd:-cmdline illisible}) - kill -9"
            kill -9 "$pid" 2>/dev/null
        done
    done <<(_td_ports)
    if [ "$busy" = 1 ] && [ "$named" = 0 ]; then
        mod_say "ports occupés mais ss n'a rendu aucun pid= (ss sans -p, ou propriétaire hors ligne)"
    fi
}

```

```

    fi
    return 0
}

mod_tearardown_start() {
    if [ -n "${TMUX:-}" ]; then
        mod_say "lancé depuis tmux : kill-server évité, on ne touche qu'à la session ${TMUX_SESSION:-}"
        tmux kill-session -t "${TMUX_SESSION:-calypso}" 2>/dev/null
    else
        tmux kill-server 2>/dev/null
    fi
    _td_kill_all
    local p
    while IFS= read -r p; do rm -f "$p" 2>/dev/null; done < <(_td_sockets)
    rm -f /dev/shm/calypso_kc /dev/shm/calypso_kc_l1 2>/dev/null
    rm -f "${RUN_DIR:-/tmp/calypso}/qemu.pid" 2>/dev/null
    mod_ok
}

mod_tearardown_wait() {
    local half=$(( ${MOD_TIMEOUT[teardown]} / 2 ))
    if wait_until "$half" "machine propre" _td_clean; then mod_ok; return $MOD_RC_OK; fi
    mod_say "restes après $half s : ${_td_leftovers} - deuxième passe"
    _td_kill_all
    _td_kill_ports
    if wait_until "$half" "machine propre" _td_clean; then mod_ok; return $MOD_RC_OK; fi
    mod_hint "identifiez le tenace : pgrep -a -f <motif> ; ss -lnp | grep <port>"
    mod_fail "restes du run précédent : ${_td_leftovers}"
    return $MOD_RC_FAIL
}

mod_tearardown_stop() {
    _td_kill_all
    local p
    while IFS= read -r p; do rm -f "$p" 2>/dev/null; done < <(_td_sockets)
    return 0
}

```

0.3.14 /opt/GSM/qosmo-overlay/run_modules/10-reseau.sh

4158 octets, 108 lignes → 108 lignes

```

MOD_REGISTER reseau "Interfaces et plan d'adressage"
MOD_REQUIRED[reseau]=1
MOD_DEPS[reseau]="rundir"
MOD_TIMEOUT[reseau]=15

: "${RESEAU_PORTS_LIBRES:=${PORT_GSM TAP_SI:-4730} ${PORT_GSM TAP_SCH:-4731}}"
: "${NEED_TUN:=0}"

_net_tenant() {
    local port="$1" out=""
    if command -v ss >/dev/null 2>&1; then
        out="$(ss -lntp "sport = :$port" 2>/dev/null | awk 'NR>1{print $0}')"
    fi
    [ -n "$out" ] && printf '%s' "$out" || printf 'tenant inconnu (ss absent ou sans droits)'
}

_net_lo_up() { ip link show lo 2>/dev/null | grep -q 'state UNKNOWN\|state UP\|<.*UP'; }
_net_route_loopback() { ip route get 127.0.0.2 >/dev/null 2>&1; }
_net_port_occupe() {
    if command -v ss >/dev/null 2>&1; then
        ss -ln 2>/dev/null | grep -qE "[:.]${1}[:space:]" && return 0
        return 1
    fi
    have_port "$1"
}

_net_ports_libres() {
    local p
    for p in $RESEAU_PORTS_LIBRES; do
        _net_port_occupe "$p" && return 1
    done
    return 0
}

_net_tun_ok() { [ -c /dev/net/tun ]; }

mod_reseau_check() {
    command -v ip >/dev/null 2>&1 || {
        mod_hint "apt-get install -y iproute2"
        mod_fail "commande « ip » absente : le réseau ne peut pas être constaté"
        return $MOD_RC_FAIL; }
    local p occupees=""
    for p in $RESEAU_PORTS_LIBRES; do
        _net_port_occupe "$p" && occupees="$occupees $p"
    done
}

```

```

    if [ -n "$occupes" ]; then
        for p in $occupes; do mod_say "port $p occupé : $( _net_tenant "$p" )"; done
        mod_hint "arrêtez le run précédent : ./run.sh --stop (le module teardown balaie a
        mod_fail "ports déjà occupés :$occupes - le plan ne pourra pas les ouvrir"
        return $MOD_RC_FAIL
    fi
    mod_ok
}

mod_reseau_status() {
    _net_lo_up || return $MOD_RC_FAIL
    _net_route_loopback || return $MOD_RC_FAIL
    if [ "${NEED_TUN}" = 1 ]; then _net_tun_ok || return $MOD_RC_FAIL; fi
    return $MOD_RC_OK
}

mod_reseau_start() {
    _net_lo_up || ip link set lo up 2>/dev/null
    if ! _net_tun_ok; then
        if [ "${id -u}" != 0 ]; then
            mod_say "/dev/net/tun absent et nous ne sommes pas root : création impossible"
        else
            modprobe tun 2>/dev/null
            if ! _net_tun_ok; then
                mkdir -p /dev/net 2>/dev/null
                mknod /dev/net/tun c 10 200 2>/dev/null && chmod 0666 /dev/net/tun 2>/dev/nul
            fi
        fi
    fi
    if _net_tun_ok; then
        mod_say "/dev/net/tun : créé"
    elif [ "${NEED_TUN}" = 1 ]; then
        mod_hint "relancez le conteneur avec --device /dev/net/tun (ou --privileged), ou
        mod_fail "/dev/net/tun indisponible alors que NEED_TUN=1"
        return $MOD_RC_FAIL
    else
        mod_say "/dev/net/tun indisponible - sans effet (NEED_TUN=0)"
    fi
fi
mod_say "lo          : $(ip -o -4 addr show dev lo 2>/dev/null | awk '{print $4}' | tr '\n'
mod_say "ports      : $RESEAU_PORTS_LIBRES (attendus libres)"
mod_ok
}

```

```

mod_reseau_wait() {
    wait_until "${MOD_TIMEOUT[reseau]}" "interface lo active" _net_lo_up || {
        mod_hint "ip link set lo up"
        mod_fail "la boucle locale n'est pas active"
        return $MOD_RC_FAIL; }
    wait_until "${MOD_TIMEOUT[reseau]}" "route vers 127.0.0.2" _net_route_loopback || {
        mod_hint "la pile écrit en dur sur 127.0.0.2 (HLR, contrôle BTS) : vérifiez ip route"
        mod_fail "127.0.0.2 n'est pas routable"
        return $MOD_RC_FAIL; }
    wait_until "${MOD_TIMEOUT[reseau]}" "libération des ports" _net_ports_libres || {
        mod_hint "un processus tient encore un port : ss -lnp | grep -E '$(printf '%s' "$RESEAU_PORTS_LIBRES")'"
        mod_fail "ports toujours occupés parmi : $RESEAU_PORTS_LIBRES"
        return $MOD_RC_FAIL; }
    if [ "${NEED_TUN}" = 1 ]; then
        wait_until "${MOD_TIMEOUT[reseau]}" "périphérique TUN" _net_tun_ok || {
            mod_hint "--device /dev/net/tun au lancement du conteneur"
            mod_fail "/dev/net/tun absent (NEED_TUN=1)"
            return $MOD_RC_FAIL; }
    fi
    mod_ok
}

mod_reseau_stop() { return 0; }

```

0.3.15 /opt/GSM/qosmo-overlay/run_modules/10-stp.sh

2472 octets, 61 lignes → 61 lignes

```

: "${MODDIR:=$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)}"
. "$MODDIR/_lib/core.sh"

MOD_REGISTER stp "Cœur - OsmoSTP (SS7/M3UA)"
MOD_REQUIRED[stp]=1
MOD_JOURNAL[stp]="osmo-stp"
MOD_TIMEOUT[stp]=25
MOD_ENABLED_IF[stp]='[ "${NO_OSMO_START:-0}" != 1 ]'

: "${STP_UNIT:=osmo-stp}"
: "${STP_VTY_PORT:=4239}"

_stp_cfg() { core_cfg osmo-stp; }

```

```

_stp_m3ua() { core_cfg_field "$(_stp_cfg)" '^[:space:]*listen[:space:]+m3ua[:space:]' }

mod_stp_check() {
    core_bin osmo-stp >/dev/null || {
        mod_hint "paquet osmo-stp absent - vérifiez l'image, ou réglez OSMOCOM_CFG/PATH"
        mod_fail "binaire osmo-stp introuvable"; return $MOD_RC_FAIL; }
    [ -r "$(_stp_cfg)" ] || {
        mod_hint "déployez la configuration : cp configs/osmo-stp.cfg $OSMOCOM_CFG/"
        mod_fail "configuration illisible : $(_stp_cfg)"; return $MOD_RC_FAIL; }
    if [ ! -e /proc/net/sctp/snmp ] && ! lsmod 2>/dev/null | grep -q '^sctp'; then
        mod_hint "chargez le module côté hôte : modprobe sctp (un conteneur ne peut pas le faire)"
        mod_fail "SCTP indisponible dans ce noyau - l'écoute M3UA est impossible"
        return $MOD_RC_FAIL
    fi
    mod_ok
}

mod_stp_status() { core_unit_active "$STP_UNIT" || core_vty_listen "$STP_VTY_PORT"; }

mod_stp_start() {
    core_svc_start "$STP_UNIT" "$(core_bin osmo-stp)" -c "$(_stp_cfg)" \
        || { mod_fail "systemctl start $STP_UNIT a échoué"
            mod_hint "journalctl -u $STP_UNIT -n 30"; return $MOD_RC_FAIL; }
    mod_ok
}

mod_stp_wait() {
    local to="${MOD_TIMEOUT[stp]}" m3ua; m3ua="$(_stp_m3ua)"

    if ! wait_until "$to" "VTY OsmoSTP ($STP_VTY_PORT)" core_vty_listen "$STP_VTY_PORT"; then
        mod_hint "journalctl -u $STP_UNIT -n 30 ; port $STP_VTY_PORT déjà pris ?"
        return $MOD_RC_FAIL
    fi
    if ! wait_until "$to" "écoute M3UA SCTP ($m3ua)" core_sctp_listen "$m3ua"; then
        mod_hint "vérifiez « listen m3ua $m3ua » dans $(_stp_cfg) et le support SCTP (ss -ln)"
        return $MOD_RC_FAIL
    fi
    if core_restarted_since "$STP_UNIT"; then
        mod_hint "journalctl -u $STP_UNIT -n 50 : le service redémarre en boucle"
        mod_fail "OsmoSTP a redémarré depuis le lancement"
        return $MOD_RC_FAIL
    fi
}

```

```

    mod_ok
}

# Motif exact : "osmo-stp -c" seul tuait aussi le hub (osmo-stp-interop.cfg),
# lance par start-native.sh --wan sur le noeud 1.
mod_stp_stop() { core_svc_stop "$STP_UNIT" "osmo-stp -c $_stp_cfg"; }

```

0.3.16 /opt/GSM/qosmo-overlay/run_modules/11-hlr.sh

2466 octets, 62 lignes → 62 lignes

```

: "${MODDIR:=$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)}"
. "$MODDIR/_lib/core.sh"

MOD_REGISTER hlr "Cœur - OsmoHLR (registre GSUP)"
MOD_REQUIRED[hlr]=1
MOD_JOURNAL[hlr]="osmo-hlr"
MOD_TIMEOUT[hlr]=30
MOD_ENABLED_IF[hlr]='[ "${NO_OSMO_START:-0}" != 1 ]'

: "${HLR_UNIT:=osmo-hlr}"
: "${HLR_VTY_PORT:=4258}"
: "${HLR_GSUP_PORT:=4222}"
: "${HLR_DB:=/var/lib/osmocom/hlr.db}"

_hlr_cfg() { core_cfg osmo-hlr; }
_hlr_gsup_ip() { core_cfg_field "$(_hlr_cfg)" '^[:space:]*bind[:space:]+ip[:space:]' 3

mod_hlr_check() {
    core_bin osmo-hlr >/dev/null || {
        mod_fail "binaire osmo-hlr introuvable"; return $MOD_RC_FAIL; }
    [ -r "$(_hlr_cfg)" ] || {
        mod_hint "déployez la configuration : cp configs/osmo-hlr.cfg $OSMOCOM_CFG/"
        mod_fail "configuration illisible : $(_hlr_cfg)"; return $MOD_RC_FAIL; }
    local dbdir; dbdir="$(dirname "$HLR_DB")"
    [ -d "$dbdir" ] || {
        mod_hint "mkdir -p $dbdir && chown osmocom:osmocom $dbdir"
        mod_fail "répertoire de base absent : $dbdir"; return $MOD_RC_FAIL; }
    core_ip_local "$(_hlr_gsup_ip)" || {
        mod_hint "ip addr add $(_hlr_gsup_ip)/24 dev lo, ou changez « gsup / bind ip » dans s
        mod_fail "adresse GSUP $(_hlr_gsup_ip) portée par aucune interface"; return $MOD_RC_1

```

```

    mod_ok
}

mod_hlr_status() { core_unit_active "$HLR_UNIT" || core_vty_listen "$HLR_VTY_PORT"; }

mod_hlr_start() {
    core_svc_start "$HLR_UNIT" "$(core_bin osmo-hlr)" -c "$(_hlr_cfg)" -l "$HLR_DB" \
        || { mod_fail "systemctl start $HLR_UNIT a échoué"
            mod_hint "journalctl -u $HLR_UNIT -n 30"; return $MOD_RC_FAIL; }
    mod_ok
}

mod_hlr_wait() {
    local to="${MOD_TIMEOUT[hlr]}" gip; gip="$(_hlr_gsup_ip)"

    if ! wait_until "$to" "VTY OsmoHLR ($HLR_VTY_PORT)" core_vty_listen "$HLR_VTY_PORT"; then
        mod_hint "journalctl -u $HLR_UNIT -n 30"
        return $MOD_RC_FAIL
    fi
    if ! wait_until "$to" "GSUP ($gip:$HLR_GSUP_PORT)" core_tcp_listen "$HLR_GSUP_PORT" "$gip"; then
        mod_hint "base verrouillée ou illisible ? ls -l $HLR_DB ; journalctl -u $HLR_UNIT -n 30"
        return $MOD_RC_FAIL
    fi
    if core_restarted_since "$HLR_UNIT"; then
        mod_hint "journalctl -u $HLR_UNIT -n 50 : le service redémarre en boucle"
        mod_fail "OsmoHLR a redémarré depuis le lancement"
        return $MOD_RC_FAIL
    fi
    mod_ok
}

mod_hlr_stop() { core_svc_stop "$HLR_UNIT" "osmo-hlr -c"; }

```

0.3.17 /opt/GSM/qosmo-overlay/run_modules/12-mgw.sh

2227 octets, 57 lignes → 57 lignes

```

: "${MODDIR:=$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)}"
. "$MODDIR/_lib/core.sh"

MOD_REGISTER mgw "Cœur - OsmoMGW (média MGCP/RTP)"

```

```

MOD_REQUIRED[mgw]=1
MOD_JOURNAL[mgw]="osmo-mgw"
MOD_TIMEOUT[mgw]=25
MOD_ENABLED_IF[mgw]=' [ "${NO_OSMO_START:-0}" != 1 ] '

: "${MGW_UNIT:=osmo-mgw}"
: "${MGW_VTY_PORT:=4243}"

_mgw_cfg() { core_cfg osmo-mgw; }
_mgw_ip() { core_cfg_field "$(_mgw_cfg)" '^[[[:space:]]*bind[[[:space:]]+ip[[[:space:]]]' 3
_mgw_port() { core_cfg_field "$(_mgw_cfg)" '^[[[:space:]]*bind[[[:space:]]+port[[[:space:]]]' 3

mod_mgw_check() {
    core_bin osmo-mgw >/dev/null || { mod_fail "binaire osmo-mgw introuvable"; return $MOD_RC
    [ -r "$(_mgw_cfg)" ] || {
        mod_hint "déployez la configuration : cp configs/osmo-mgw.cfg $OSMOCOM_CFG/"
        mod_fail "configuration illisible : $(_mgw_cfg)"; return $MOD_RC_FAIL; }
    core_ip_local "$(_mgw_ip)" || {
        mod_hint "corrigez « mgcp / bind ip » dans $(_mgw_cfg), ou ajoutez l'adresse à une in
        mod_fail "adresse MGCP $(_mgw_ip) portée par aucune interface"; return $MOD_RC_FAIL;
    mod_ok
}

mod_mgw_status() { core_unit_active "$MGW_UNIT" || core_vty_listen "$MGW_VTY_PORT"; }

mod_mgw_start() {
    core_svc_start "$MGW_UNIT" "$(core_bin osmo-mgw)" -s -c "$(_mgw_cfg)" \
        || { mod_fail "systemctl start $MGW_UNIT a échoué"
            mod_hint "journalctl -u $MGW_UNIT -n 30"; return $MOD_RC_FAIL; }
    mod_ok
}

mod_mgw_wait() {
    local to="${MOD_TIMEOUT[mgw]}" ip port
    ip="$(_mgw_ip)"; port="$(_mgw_port)"

    if ! wait_until "$to" "VTY OsmoMGW ($MGW_VTY_PORT)" core_vty_listen "$MGW_VTY_PORT"; then
        mod_hint "journalctl -u $MGW_UNIT -n 30"
        return $MOD_RC_FAIL
    fi
    if ! wait_until "$to" "MGCP UDP ($ip:$port)" core_udp_listen "$port" "$ip"; then
        mod_hint "port $port déjà pris ? ss -lun | grep $port"

```

```

        return $MOD_RC_FAIL
    fi
    if core_restarted_since "$MGW_UNIT"; then
        mod_hint "journalctl -u $MGW_UNIT -n 50 : le service redémarre en boucle"
        mod_fail "OsmoMGW a redémarré depuis le lancement"
        return $MOD_RC_FAIL
    fi
    mod_ok
}

mod_mgw_stop() { core_svc_stop "$MGW_UNIT" "osmo-mgw -s -c"; }

```

0.3.18 /opt/GSM/qosmo-overlay/run_modules/13-msc.sh

2516 octets, 65 lignes → 65 lignes

```

: "${MODDIR:=$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)}"
. "$MODDIR/_lib/core.sh"

MOD_REGISTER msc "Cœur - OsmoMSC (commutation CS)"
MOD_REQUIRED[msc]=1
MOD_DEPS[msc]="stp hlr mgw"
MOD_JOURNAL[msc]="osmo-msc"
MOD_TIMEOUT[msc]=40
MOD_ENABLED_IF[msc]='[ "${NO_OSMO_START:-0}" != 1 ]'

: "${MSC_UNIT:=osmo-msc}"
: "${MSC_VTY_PORT:=4254}"

_msc_cfg()      { core_cfg osmo-msc; }
_msc_asp_port() { core_cfg_field "$(_msc_cfg)" '^[:space:]*asp[:space:]+.*[:space:]*m3u'; }
_msc_mncc()     { core_cfg_field "$(_msc_cfg)" '^[:space:]*mncc[:space:]+external[:space:]*'; }

mod_msc_check() {
    core_bin osmo-msc >/dev/null || { mod_fail "binaire osmo-msc introuvable"; return $MOD_RC_FAIL; }
    [ -r "$(_msc_cfg)" ] || {
        mod_hint "déployez la configuration : cp configs/osmo-msc.cfg $OSMOCOM_CFG/"
        mod_fail "configuration illisible : $(_msc_cfg)"; return $MOD_RC_FAIL; }
    mod_ok
}

```

```

mod_msc_status() { core_unit_active "$MSC_UNIT" || core_vty_listen "$MSC_VTY_PORT"; }

mod_msc_start() {
    core_svc_start "$MSC_UNIT" "$(core_bin osmo-msc)" -c "$(_msc_cfg)" \
        || { mod_fail "systemctl start $MSC_UNIT a échoué"
            mod_hint "journalctl -u $MSC_UNIT -n 30"; return $MOD_RC_FAIL; }
    mod_ok
}

mod_msc_wait() {
    local to="${MOD_TIMEOUT[msc]}" asp mncc
    asp="$(_msc_asp_port)"; mncc="$(_msc_mncc)"

    if ! wait_until "$to" "VTY OsmoMSC ($MSC_VTY_PORT)" core_vty_listen "$MSC_VTY_PORT"; then
        mod_hint "journalctl -u $MSC_UNIT -n 30"
        return $MOD_RC_FAIL
    fi
    if [ -n "$asp" ]; then
        if ! wait_until "$to" "lien M3UA vers le STP (SCTP local :$asp)" core_sctp_estab "$asp"; then
            mod_hint "STP joignable ? ss -an --sctp | grep $asp ; vérifiez « asp ... m3ua »"
            return $MOD_RC_FAIL
        fi
    else
        mod_say "aucun ASP m3ua dans $(_msc_cfg) - barrière SS7 non applicable"
    fi
    if [ -n "$mncc" ]; then
        if ! wait_until "$to" "socket MNCC ($mncc)" core_unix_listen "$mncc"; then
            mod_hint "« mncc external $mncc » configuré mais la socket n'apparaît pas : ss -tln"
            return $MOD_RC_FAIL
        fi
    fi
    if core_restarted_since "$MSC_UNIT"; then
        mod_hint "journalctl -u $MSC_UNIT -n 50 : le service redémarre en boucle"
        mod_fail "OsmoMSC a redémarré depuis le lancement"
        return $MOD_RC_FAIL
    fi
    mod_ok
}

mod_msc_stop() { core_svc_stop "$MSC_UNIT" "osmo-msc -c"; }

```

0.3.19 /opt/GSM/qosmo-overlay/run_modules/13-sidecar-cfg.sh

5092 octets, 140 lignes → 140 lignes

```
. "$(dirname "${BASH_SOURCE[0]}")/_lib/radio.sh"
. "$(dirname "${BASH_SOURCE[0]}")/_lib/modem.sh"

MOD_REGISTER sidecar-cfg "Configurations du BTS#1 (side-car)"
MOD_REQUIRED[sidecar-cfg]=1
MOD_DEPS[sidecar-cfg]="config"
MOD_PURE[sidecar-cfg]=1
MOD_TIMEOUT[sidecar-cfg]=10

: "${BSC_CFG:=${OSMOCOM_CFG:-/etc/osmocom}/osmo-bsc.cfg}"

_sc_bsc_declare() { grep -qE "^[[:space:]]*ipa unit-id ${SC_UNIT_ID} 0" "$BSC_CFG" 2>/dev/nu.

mod_sidecar_cfg_check() {
    [ -r "$BSC_CFG" ] || {
        mod_hint "posez BSC_CFG, ou déployez /etc/osmocom/osmo-bsc.cfg"
        mod_fail "configuration BSC illisible : $BSC_CFG"; return $MOD_RC_FAIL; }
    mod_ok
}

mod_sidecar_cfg_status() {
    _sc_bsc_declare && [ -r "$SC_BTS_CFG" ] && [ -r "$SC_MOBILE_CFG" ]
}

# Le Gb du BTS#1 : meme bloc que le BTS principal (nsei 10, bvci 12, udp 23001),
# avec ses propres identifiants (_lib/modem.sh) pour que le SGSN y voie un second
# NSE. Sans lui, « gprs mode none » : pas de PDCH servi, le second PCU n'a rien.
_sc_gprs_block() {
    if ! modem_enabled; then printf ' gprs mode none\n'; return; fi
    cat <<GPRS
gprs mode egprs
gprs routing area 0
gprs network-control-order nc0
gprs power-control alpha 0
gprs cell bvci ${SC_GPRS_BVCI}
gprs cell timer blocking-timer 3
gprs cell timer blocking-retries 3
gprs cell timer unblocking-retries 3
gprs cell timer reset-timer 3
```

```

gprs cell timer reset-retries 3
gprs cell timer suspend-timer 10
gprs cell timer suspend-retries 3
gprs cell timer resume-timer 10
gprs cell timer resume-retries 3
gprs cell timer capability-update-timer 10
gprs cell timer capability-update-retries 3
gprs nsei ${SC_GPRS_NSEI}
gprs ns timer tns-block 3
gprs ns timer tns-block-retries 3
gprs ns timer tns-reset 3
gprs ns timer tns-reset-retries 3
gprs ns timer tns-test 30
gprs ns timer tns-alive 3
gprs ns timer tns-alive-retries 10
gprs nsvc 0 nsvci ${SC_GPRS_NSVC1}
gprs nsvc 0 local udp port ${SC_GPRS_NS_PORT}
gprs nsvc 0 remote ip ${SC_SGSN_IP}
gprs nsvc 0 remote udp port ${SC_SGSN_PORT}
gprs nsvc 1 nsvci 0
gprs nsvc 1 local udp port 0
gprs nsvc 1 remote ip 0.0.0.0
gprs nsvc 1 remote udp port 0
GPRS
}

```

```

_sc_bts_block() {
    cat <<BLOC
bts 1
    type osmo-bts
    band DCS1800
    cell_identity ${SC_UNIT_ID}
    location_area_code 0x0001
    base_station_id_code 8
    ms max power 15
    cell reselection hysteresis 4
    rxlev access min 0
    radio-link-timeout 32
    channel allocator mode chan-req descending
    channel allocator mode assignment ascending
    channel allocator mode handover ascending
    rach tx integer 9

```

```

rach max transmission 7
ipa unit-id ${SC_UNIT_ID} 0
oml ipa stream-id 255 line 0
neighbor-list mode automatic
codec-support fr
$_sc_gprs_block)
  trx 0
  rf_locked 0
  arfcn ${SC_ARFCN}
  nominal power 23
  max_power_red 0
  rsl e1 tei 0
$(for t in 0 1 2 3 4 5 6 7; do
  case "$t" in
    0) pchan='CCCH+SDCCH4' ;;
    1) pchan="${SC_TS1_PCHAN:-SDCCH8}" ;;
    5|6|7) pchan="${SC_TS_PDCH_PCHAN:-PDCH}" ;; # canaux paquet du modem GPRS (comme con
    *) pchan='TCH/F' ;;
  esac
  printf '    timeslot %s\n    phys_chan_config %s\n    hopping enabled 0\n' "$t" "$pchan"
done)
BLOC
}

mod_sidecar_cfg_start() {
  if _sc_bsc_declare; then
    mod_say "bloc « bts 1 » (unit-id $SC_UNIT_ID) déjà déclaré dans $BSC_CFG"
  else
    grep -qE '^msc 0' "$BSC_CFG" || {
      mod_hint "le bloc « bts 1 » s'insère avant « msc 0 » ; cette ancre est absente"
      mod_fail "structure inattendue dans $BSC_CFG"; return $MOD_RC_FAIL; }
    cp -f "$BSC_CFG" "${BSC_CFG}.avant-sidecar" 2>/dev/null
    local tmp; tmp="$(mktemp)"
    awk -v bloc="$_sc_bts_block" '
      /^msc 0/ && !fait { print bloc; fait = 1 }
      { print }
    ' "$BSC_CFG" > "$tmp" || { rm -f "$tmp"; mod_fail "réécriture de $BSC_CFG impossible"
    cat "$tmp" > "$BSC_CFG" && rm -f "$tmp"
    _sc_bsc_declare || { mod_fail "insertion du bloc « bts 1 » sans effet"; return $MOD_RC_FAIL; }
    mod_say "bloc « bts 1 » inséré (sauvegarde : ${BSC_CFG}.avant-sidecar)"
  fi
}

```

```

[ -r "$SC_BTS_CFG" ] || {
    mod_hint "attendu : $SC_BTS_CFG (VTY $SC_BTS_VTY_PORT, base-port remote $SC_TRX_PORT"
    mod_fail "configuration du BTS#1 absente"; return $MOD_RC_FAIL; }
local port; port="$(radio_cfg_val "$SC_BTS_CFG" "osmotrx base-port remote")"
[ "${port:-$SC_TRX_PORT}" = "$SC_TRX_PORT" ] && mod_say "BTS#1 : $SC_BTS_CFG (TRXD $SC_T"
    || mod_say " $SC_BTS_CFG annonce base-port remote $port, le side-car attend $SC_TRX"

if [ ! -r "$SC_MOBILE_CFG" ]; then
    mod_hint "attendu : $SC_MOBILE_CFG (layer2-socket $SC_L2_SOCKET, VTY $SC_MOBILE_VTY_PO"
    mod_fail "configuration du MS#2 absente"; return $MOD_RC_FAIL
fi
local imsi; imsi="$(awk '$1=="imsi"{print $2; exit}' "$SC_MOBILE_CFG" 2>/dev/null)"
mod_say "MS#2 : $SC_MOBILE_CFG (imsi ${imsi:-?}, socket $SC_L2_SOCKET)"
mod_ok
}

mod_sidecar_cfg_stop() { return 0; }

```

0.3.20 /opt/GSM/qosmo-overlay/run_modules/14-bsc.sh

2850 octets, 71 lignes → 71 lignes

```

: "${MODDIR:=$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)}"
. "$MODDIR/_lib/core.sh"

MOD_REGISTER bsc "Cœur - OsmoBSC (contrôleur BTS)"
MOD_REQUIRED[bsc]=1
MOD_DEPS[bsc]="stp mgw msc"
MOD_JOURNAL[bsc]="osmo-bsc"
MOD_TIMEOUT[bsc]=40
MOD_ENABLED_IF[bsc]='[ "${NO_OSMO_START:-0}" != 1 ]'

: "${BSC_UNIT:=osmo-bsc}"
: "${BSC_VTY_PORT:=4242}"
: "${BSC_OML_PORT:=3002}"
: "${BSC_RSL_PORT:=3003}"

_bsc_cfg()      { core_cfg osmo-bsc; }
_bsc_osp_port() { core_cfg_field "$(_bsc_cfg)" '^[[[:space:]]*asp[[[:space:]]+.*[[[:space:]]m3u

mod_bsc_check() {

```

```

core_bin osmo-bsc >/dev/null || { mod_fail "binaire osmo-bsc introuvable"; return $MOD_RC
[ -r "$(_bsc_cfg)" ] || {
    mod_hint "déployez la configuration : cp configs/osmo-bsc.cfg $OSMOCOM_CFG/"
    mod_fail "configuration illisible : $_bsc_cfg"; return $MOD_RC_FAIL; }
if grep -qa '__[A-Z_]\+__' "$(_bsc_cfg)" 2>/dev/null; then
    mod_hint "la configuration contient encore des jetons de gabarit - régénérez-la"
    mod_fail "$(_bsc_cfg) : gabarit non substitué"; return $MOD_RC_FAIL
fi
mod_ok
}

mod_bsc_status() { core_unit_active "$BSC_UNIT" || core_vty_listen "$BSC_VTY_PORT"; }

mod_bsc_start() {
    core_svc_start "$BSC_UNIT" "$(core_bin osmo-bsc)" -c "$(_bsc_cfg)" -s \
        || { mod_fail "systemctl start $BSC_UNIT a échoué"
            mod_hint "journalctl -u $BSC_UNIT -n 30"; return $MOD_RC_FAIL; }
    mod_ok
}

mod_bsc_wait() {
    local to="${MOD_TIMEOUT[bsc]}" asp="$_bsc_asp_port"

    if ! wait_until "$to" "VTY OsmoBSC ($BSC_VTY_PORT)" core_vty_listen "$BSC_VTY_PORT"; then
        mod_hint "journalctl -u $BSC_UNIT -n 30"
        return $MOD_RC_FAIL
    fi
    if ! wait_until "$to" "écoute A-bis OML ($BSC_OML_PORT)" core_tcp_listen "$BSC_OML_PORT"
        mod_hint "port $BSC_OML_PORT déjà pris par un autre BSC ? ss -ltn | grep $BSC_OML_PORT"
        return $MOD_RC_FAIL
    fi
    if ! wait_until "$to" "écoute A-bis RSL ($BSC_RSL_PORT)" core_tcp_listen "$BSC_RSL_PORT"
        mod_hint "port $BSC_RSL_PORT déjà pris ? ss -ltn | grep $BSC_RSL_PORT"
        return $MOD_RC_FAIL
    fi
    if [ -n "$asp" ]; then
        if ! wait_until "$to" "lien M3UA vers le STP (SCTP local :$asp)" core_sctp_estab "$asp"
            mod_hint "STP joignable ? ss -an --sctp | grep $asp ; vérifiez « asp ... m3ua »"
            return $MOD_RC_FAIL
        fi
    else
        mod_say "aucun ASP m3ua dans $_bsc_cfg - barrière SS7 non applicable"
    fi
}

```

```

    fi
    if core_restarted_since "$BSC_UNIT"; then
        mod_hint "journalctl -u $BSC_UNIT -n 50 : le service redémarre en boucle"
        mod_fail "OsmoBSC a redémarré depuis le lancement"
        return $MOD_RC_FAIL
    fi
    mod_ok
}

mod_bsc_stop() { core_svc_stop "$BSC_UNIT" "osmo-bsc -c"; }

```

0.3.21 /opt/GSM/qosmo-overlay/run_modules/15-apn0.sh

1833 octets, 48 lignes → 48 lignes

```

: "${MODDIR:=$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)}"
. "$MODDIR/_lib/core.sh"

MOD_REGISTER apn0 "Cœur - interface TUN de l'APN"
MOD_REQUIRED[apn0]=0
MOD_TIMEOUT[apn0]=15
MOD_ENABLED_IF[apn0]='[ "${NO_OSMO_START:-0}" != 1 ] && [ "${CORE_GPRS:-1}" = 1 ]'

: "${APN_DEV:=apn0}"
: "${APN_ADDR:=176.16.32.0/24}"

_apn_up() { ip -o link show "$APN_DEV" 2>/dev/null | grep -qE '<[^>]*(^|<|,)UP(,|>)' ; }
_apn_addr() { ip -o -4 addr show dev "$APN_DEV" 2>/dev/null | grep -q "${APN_ADDR%%/*}"; }

mod_apn0_check() {
    command -v ip >/dev/null 2>&1 || { mod_fail "commande ip absente (paquet iproute2)"; return 1; }
    [ -c /dev/net/tun ] || {
        mod_hint "conteneur sans TUN : ajoutez --device /dev/net/tun et --cap-add NET_ADMIN"
        mod_fail "/dev/net/tun absent - impossible de créer une interface TUN"
        return $MOD_RC_FAIL; }
    mod_ok
}

mod_apn0_status() { _apn_up && _apn_addr; }

mod_apn0_start() {

```

```

ip link show "$APN_DEV" >/dev/null 2>&1 && ip link del dev "$APN_DEV" 2>/dev/null
ip tuntap add dev "$APN_DEV" mode tun 2>&1 || {
    mod_hint "droits insuffisants : lancez le conteneur avec --cap-add NET_ADMIN"
    mod_fail "création de $APN_DEV refusée"; return $MOD_RC_FAIL; }
ip addr add "$APN_ADDR" dev "$APN_DEV" 2>&1
ip link set dev "$APN_DEV" up 2>&1
mod_ok
}

mod_apn0_wait() {
    if ! wait_until "${MOD_TIMEOUT[apn0]}" "interface $APN_DEV UP" _apn_up; then
        mod_hint "ip link show $APN_DEV"
        return $MOD_RC_FAIL
    fi
    if ! wait_until "${MOD_TIMEOUT[apn0]}" "adresse $APN_ADDR sur $APN_DEV" _apn_addr; then
        mod_hint "adresse déjà portée par une autre interface ? ip -4 addr | grep ${APN_ADDR}"
        return $MOD_RC_FAIL
    fi
    mod_ok
}

mod_apn0_stop() { ip link del dev "$APN_DEV" 2>/dev/null; return 0; }

```

0.3.22 /opt/GSM/qosmo-overlay/run_modules/16-ggsn.sh

2634 octets, 65 lignes → 65 lignes

```

: "${MODDIR:=$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)}"
. "$MODDIR/_lib/core.sh"

MOD_REGISTER ggsn "Cœur - OsmoGGSN (data GTP)"
MOD_REQUIRED[ggsn]=0
MOD_DEPS[ggsn]="apn0"
MOD_JOURNAL[ggsn]="osmo-ggsn"
MOD_TIMEOUT[ggsn]=25
MOD_ENABLED_IF[ggsn]='[ "${NO_OSMO_START:-0}" != 1 ] && [ "${CORE_GPRS:-1}" = 1 ]'

: "${GGSN_UNIT:=osmo-ggsn}"
: "${GGSN_VTY_PORT:=4260}"
: "${GGSN_GTPC_PORT:=2123}"
: "${GGSN_GTPU_PORT:=2152}"

```

```

_ggsn_cfg() { core_cfg osmo-ggsn; }
_ggsn_ip() { core_cfg_field "$(_ggsn_cfg)" '^[[[:space:]]*gtp[[[:space:]]+bind-ip[[[:space:]]]'

mod_ggsn_check() {
    core_bin osmo-ggsn >/dev/null || { mod_fail "binaire osmo-ggsn introuvable"; return $MOD_FAIL; }
    [ -r "$(_ggsn_cfg)" ] || {
        mod_hint "déployez la configuration : cp configs/osmo-ggsn.cfg $OSMOCOM_CFG/"
        mod_fail "configuration illisible : "$(_ggsn_cfg)"; return $MOD_RC_FAIL; }
    local ip; ip="$(_ggsn_ip)"
    [ -n "$ip" ] || { mod_fail "« gtp bind-ip » absent de "$(_ggsn_cfg)"; return $MOD_RC_FAIL; }
    core_ip_local "$ip" || {
        mod_hint "corrigez « gtp bind-ip » dans "$(_ggsn_cfg) : aucune interface ne porte $ip"
        mod_fail "adresse GTP $ip introuvable localement - le GGSN refusera de démarrer"
        return $MOD_RC_FAIL; }
    mod_ok
}

mod_ggsn_status() { core_unit_active "$GGSN_UNIT" || core_vty_listen "$GGSN_VTY_PORT"; }

mod_ggsn_start() {
    core_svc_start "$GGSN_UNIT" "$(core_bin osmo-ggsn)" -c "$(_ggsn_cfg)" \
        || { mod_fail "systemctl start $GGSN_UNIT a échoué"
            mod_hint "journalctl -u $GGSN_UNIT -n 30"; return $MOD_RC_FAIL; }
    mod_ok
}

mod_ggsn_wait() {
    local to="${MOD_TIMEOUT[ggsn]}" ip; ip="$(_ggsn_ip)"

    if ! wait_until "$to" "VTY OsmoGGSN ($GGSN_VTY_PORT)" core_vty_listen "$GGSN_VTY_PORT"; then
        mod_hint "journalctl -u $GGSN_UNIT -n 30"
        return $MOD_RC_FAIL
    fi
    if ! wait_until "$to" "GTP-C ($ip:$GGSN_GTPC_PORT)" core_udp_listen "$GGSN_GTPC_PORT" "$ip"; then
        mod_hint "l'interface TUN de l'APN est-elle montée ? ip link show ${APN_DEV:-apn0}"
        return $MOD_RC_FAIL
    fi
    if ! wait_until "$to" "GTP-U ($ip:$GGSN_GTPU_PORT)" core_udp_listen "$GGSN_GTPU_PORT" "$ip"; then
        mod_hint "port $GGSN_GTPU_PORT déjà pris ? ss -lun | grep $GGSN_GTPU_PORT"
        return $MOD_RC_FAIL
    fi
    if core_restarted_since "$GGSN_UNIT"; then

```

```

        mod_hint "journalctl -u $GGSN_UNIT -n 50 : le service redémarre en boucle"
        mod_fail "OsmoGGSN a redémarré depuis le lancement"
        return $MOD_RC_FAIL
    fi
    mod_ok
}

mod_ggsn_stop() { core_svc_stop "$GGSN_UNIT" "osmo-ggsn -c"; }

```

0.3.23 /opt/GSM/qosmo-overlay/run_modules/17-sgsn.sh

2351 octets, 60 lignes → 60 lignes

```

: "${MODDIR:=$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)}"
. "$MODDIR/_lib/core.sh"

MOD_REGISTER sgsn "Cœur - OsmoSGSN (rattachement paquet)"
MOD_REQUIRED[sgsn]=0
MOD_DEPS[sgsn]="hlr"
MOD_JOURNAL[sgsn]="osmo-sgsn"
MOD_TIMEOUT[sgsn]=25
MOD_ENABLED_IF[sgsn]='[ "${NO_OSMO_START:-0}" != 1 ] && [ "${CORE_GPRS:-1}" = 1 ]'

: "${SGSN_UNIT:=osmo-sgsn}"
: "${SGSN_VTY_PORT:=4245}"

_sgsn_cfg() { core_cfg osmo-sgsn; }
_sgsn_ns_ip() { core_cfg_field "$(_sgsn_cfg)" '^[:space:]*listen[:space:]+[0-9]' 2 ""; }
_sgsn_ns_port() { core_cfg_field "$(_sgsn_cfg)" '^[:space:]*listen[:space:]+[0-9]' 3 2300; }

mod_sgsn_check() {
    core_bin osmo-sgsn >/dev/null || { mod_fail "binaire osmo-sgsn introuvable"; return $MOD_RC_FAIL; }
    [ -r "$(_sgsn_cfg)" ] || {
        mod_hint "déployez la configuration : cp configs/osmo-sgsn.cfg $OSMOCOM_CFG/"
        mod_fail "configuration illisible : $(_sgsn_cfg)"; return $MOD_RC_FAIL; }
    local ip; ip="$(_sgsn_ns_ip)"
    if [ -n "$ip" ] && ! core_ip_local "$ip"; then
        mod_hint "corrigez « ns / bind udp local / listen » dans $(_sgsn_cfg)"
        mod_fail "adresse NS $ip portée par aucune interface"; return $MOD_RC_FAIL
    fi
    mod_ok
}

```

```

}

mod_sgsn_status() { core_unit_active "$SGSN_UNIT" || core_vty_listen "$SGSN_VTY_PORT"; }

mod_sgsn_start() {
    core_svc_start "$SGSN_UNIT" "$(core_bin osmo-sgsn)" -c "$(_sgsn_cfg)" \
        || { mod_fail "systemctl start $SGSN_UNIT a échoué"
            mod_hint "journalctl -u $SGSN_UNIT -n 30"; return $MOD_RC_FAIL; }
    mod_ok
}

mod_sgsn_wait() {
    local to="${MOD_TIMEOUT[sgsn]}" ip port
    ip="$(_sgsn_ns_ip)"; port="$(_sgsn_ns_port)"

    if ! wait_until "$to" "VTY OsmoSGSN ($SGSN_VTY_PORT)" core_vty_listen "$SGSN_VTY_PORT"; then
        mod_hint "journalctl -u $SGSN_UNIT -n 30"
        return $MOD_RC_FAIL
    fi
    if ! wait_until "$to" "NS UDP ($ip:$port)" core_udp_listen "$port" "$ip"; then
        mod_hint "adresse $ip absente ou port $port déjà pris : ss -lun | grep $port"
        return $MOD_RC_FAIL
    fi
    if core_restarted_since "$SGSN_UNIT"; then
        mod_hint "journalctl -u $SGSN_UNIT -n 50 : le service redémarre en boucle"
        mod_fail "OsmoSGSN a redémarré depuis le lancement"
        return $MOD_RC_FAIL
    fi
    mod_ok
}

mod_sgsn_stop() { core_svc_stop "$SGSN_UNIT" "osmo-sgsn -c"; }

```

0.3.24 /opt/GSM/qosmo-overlay/run_modules/18-pcu.sh

2008 octets, 53 lignes → 53 lignes

```

: "${MODDIR:=$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)}"
. "$MODDIR/_lib/core.sh"

MOD_REGISTER pcu "Cœur - OsmoPCU (ordonnanceur paquet)"

```

```

MOD_REQUIRED[pcu]=0
MOD_DEPS[pcu]="sgsn"
MOD_JOURNAL[pcu]="osmo-pcu"
MOD_TIMEOUT[pcu]=20
MOD_ENABLED_IF[pcu]='[ "${NO_OSMO_START:-0}" != 1 ] && [ "${CORE_GPRS:-1}" = 1 ]'

: "${PCU_UNIT:=osmo-pcu}"
: "${PCU_VTY_PORT:=4240}"
: "${PCU_SOCKET:=/tmp/pcu_bts}"

_pcu_cfg() { core_cfg osmo-pcu; }

mod_pcu_check() {
    core_bin osmo-pcu >/dev/null || { mod_fail "binaire osmo-pcu introuvable"; return $MOD_RC; }
    [ -r "$(_pcu_cfg)" ] || {
        mod_hint "déployez la configuration : cp configs/osmo-pcu.cfg $OSMOCOM_CFG/"
        mod_fail "configuration illisible : $(_pcu_cfg)"; return $MOD_RC_FAIL; }
    mod_ok
}

mod_pcu_status() { core_unit_active "$PCU_UNIT" || core_vty_listen "$PCU_VTY_PORT"; }

mod_pcu_start() {
    core_svc_start "$PCU_UNIT" "$(core_bin osmo-pcu)" -c "$(_pcu_cfg)" \
        || { mod_fail "systemctl start $PCU_UNIT a échoué"
            mod_hint "journalctl -u $PCU_UNIT -n 30"; return $MOD_RC_FAIL; }
    [ -e "$PCU_SOCKET" ] && chmod 777 "$PCU_SOCKET" 2>/dev/null
    mod_ok
}

mod_pcu_wait() {
    if ! core_alive "$PCU_UNIT"; then
        mod_hint "journalctl -u $PCU_UNIT -n 30"
        mod_fail "osmo-pcu n'est pas resté en vie après le lancement"
        return $MOD_RC_FAIL
    fi
    if ! wait_until "${MOD_TIMEOUT[pcu]}" "VTY OsmoPCU ($PCU_VTY_PORT)" core_vty_listen "$PCU_VTY_PORT"; then
        mod_say "VTY $PCU_VTY_PORT non observé - port non déclaré dans $(_pcu_cfg), repli sur le mode de secours"
    fi
    if core_restarted_since "$PCU_UNIT"; then
        mod_hint "journalctl -u $PCU_UNIT -n 50 : conf invalide ou socket PCU inaccessible"
        mod_fail "OsmoPCU redémarre en boucle"
    fi
}

```

```

        return $MOD_RC_FAIL
    fi
    core_unix_listen "$PCU_SOCKET" && mod_say "socket PCU $PCU_SOCKET présente (BTS déjà démarré)"
    mod_ok
}

mod_pcu_stop() { core_svc_stop "$PCU_UNIT" "osmo-pcu -c"; }

```

0.3.25 /opt/GSM/qosmo-overlay/run_modules/19-asterisk.sh

2241 octets, 66 lignes → 66 lignes

```

: "${MODDIR:=$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)}"
. "$MODDIR/_lib/core.sh"

MOD_REGISTER asterisk "Cœur - Asterisk (PBX SIP)"
MOD_REQUIRED[asterisk]=0
MOD_JOURNAL[asterisk]="asterisk"
MOD_TIMEOUT[asterisk]=45
MOD_ENABLED_IF[asterisk]='[ "${NO_OSMO_START:-0}" != 1 ] && [ "${CORE_VOICE:-1}" = 1 ]'

: "${ASTERISK_UNIT:=asterisk}"
: "${ASTERISK_CFG:=/etc/asterisk/asterisk.conf}"

_ast_cli() { asterisk -rx "$1" 2>/dev/null; }
_ast_ready() { _ast_cli "core show uptime" | grep -qi 'uptime'; }

mod_asterisk_check() {
    command -v asterisk >/dev/null 2>&1 || {
        mod_hint "installez asterisk, ou désactivez la voix : CORE_VOICE=0"
        mod_fail "binaire asterisk introuvable"; return $MOD_RC_FAIL; }
    [ -r "$ASTERISK_CFG" ] || {
        mod_fail "configuration illisible : $ASTERISK_CFG"; return $MOD_RC_FAIL; }
    mod_ok
}

mod_asterisk_status() { _ast_ready; }

mod_asterisk_start() {
    if _ast_ready; then mod_say "déjà actif - on ne relance pas"; mod_ok; return 0; fi
}

```

```

local bin pf log
bin="$(command -v asterisk)"
pf="$(core_pidfile "$ASTERISK_UNIT")"
log="${LOG_DIR}/${ASTERISK_UNIT}.log"
mkdir -p "$RUN_DIR" "$LOG_DIR" 2>/dev/null || true

mod_say "lancement direct : $bin -f -U asterisk"
setsid "$bin" -f -U asterisk >>"$log" 2>&1 </dev/null &
printf '%s\n' "$!" > "$pf"

sleep 1
if ! kill -0 "$(cat "$pf" 2>/dev/null)" 2>/dev/null; then
    mod_hint "tail -30 $log"
    mod_fail "asterisk est mort dans la seconde qui a suivi le lancement"
    return $MOD_RC_FAIL
fi
mod_ok
}

mod_asterisk_wait() {
    if ! wait_until "${MOD_TIMEOUT[asterisk]}" "console Asterisk" _ast_ready; then
        mod_hint "asterisk -rx 'core show uptime' ; journalctl -u $ASTERISK_UNIT -n 40"
        return $MOD_RC_FAIL
    fi
    local pf; pf="$(core_pidfile "$ASTERISK_UNIT")"
    if [ -f "$pf" ] && ! kill -0 "$(cat "$pf" 2>/dev/null)" 2>/dev/null; then
        mod_hint "tail -50 ${LOG_DIR}/${ASTERISK_UNIT}.log"
        mod_fail "le PID lancé n'existe plus : Asterisk est mort ou a été remplacé"
        return $MOD_RC_FAIL
    fi
    mod_ok
}

mod_asterisk_stop() {
    _ast_cli "core stop now" >/dev/null 2>&1
    core_svc_stop "$ASTERISK_UNIT" ""
}

```

0.3.26 /opt/GSM/qosmo-overlay/run_modules/20-mobile-cfg.sh

3917 octets, 89 lignes → 89 lignes

```

MOD_REGISTER mobile-cfg "Configuration du mobile"
MOD_REQUIRED[mobile-cfg]=0
MOD_DEPS[mobile-cfg]="config"
MOD_TIMEOUT[mobile-cfg]=10

_mcfg_imsi() { grep -oE '^[[:space:]]*imsi [0-9]{15}' "${MOBILE_CFG:-}" 2>/dev/null | head -n 1
_mcfg_ki() { sed -n 's/^[[:space:]]*ki comp128 \([0-9a-fA-F ]*\)\$/\1/p' "${MOBILE_CFG:-}" 2>/dev/null
_mcfg_stick(){ sed -n 's/^[[:space:]]*stick \([0-9]\+\)\.*/\1/p' "${MOBILE_CFG:-}" 2>/dev/null

mod_mobile_cfg_check() {
    if [ "${CALYPSO_SYNC_MOBILE_CFG:-1}" != "1" ]; then
        [ -r "${MOBILE_CFG:-}" ] && { mod_say "synchronisation désactivée, fichier existant"
        mod_hint "CALYPSO_SYNC_MOBILE_CFG=1 pour réinstaller la version du dépôt"
        mod_fail "synchronisation désactivée et ${MOBILE_CFG:-<non défini>} absent"
        return $MOD_RC_FAIL
        fi
    if [ ! -r "${MOBILE_CFG_SRC:-}" ]; then
        mod_hint "attendu : \${QEMU_TREE}/cfgs/mobile_group1.cfg - vérifiez QEMU_CFGS dans path
        mod_skip "configuration versionnée absente : ${MOBILE_CFG_SRC:-<non défini>}"
        return $MOD_RC_SKIP
        fi
    mod_ok
}

mod_mobile_cfg_status() {
    [ -r "${MOBILE_CFG:-}" ] || return 1
    [ "${CALYPSO_SYNC_MOBILE_CFG:-1}" = "1" ] || return 0
    cmp -s "${MOBILE_CFG_SRC:-}" "${MOBILE_CFG}" 2>/dev/null || return 1
    [ -z "${CALYPSO_STICK_ARFCN:-}" ] || [ "${CALYPSO_STICK_ARFCN}" = "0" ]
}

mod_mobile_cfg_start() {
    if [ "${CALYPSO_SYNC_MOBILE_CFG:-1}" = "1" ]; then
        mkdir -p "${(dirname "${MOBILE_CFG}")}" 2>/dev/null
        if ! cp "${MOBILE_CFG_SRC}" "${MOBILE_CFG}" 2>/dev/null; then
            mod_hint "vérifiez les droits sur ${dirname "${MOBILE_CFG}"}"
            mod_fail "copie impossible : ${MOBILE_CFG_SRC} -> ${MOBILE_CFG}"
            return $MOD_RC_FAIL
        fi
        mod_say "installée : ${MOBILE_CFG_SRC} -> ${MOBILE_CFG}"
    fi
}

```



```

    fi
    mod_ok
}

mod_mobile_cfg_stop() { return 0; }

```

0.3.27 /opt/GSM/qosmo-overlay/run_modules/20-sip-connector.sh

2710 octets, 64 lignes → 64 lignes

```

: "${MODDIR:=$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)}"
. "$MODDIR/_lib/core.sh"

MOD_REGISTER sip-connector "Cœur - passerelle MNCC SIP"
MOD_REQUIRED[sip-connector]=0
MOD_DEPS[sip-connector]="msc asterisk"
MOD_JOURNAL[sip-connector]="osmo-sip-connector"
MOD_TIMEOUT[sip-connector]=25
MOD_ENABLED_IF[sip-connector]='[ "${NO_OSMO_START:-0}" != 1 ] && [ "${CORE_VOICE:-1}" = 1 ]'

: "${SIPC_UNIT:=osmo-sip-connector}"

_sipc_cfg() { core_cfg osmo-sip-connector; }
_sipc_ip() { core_cfg_field "$(_sipc_cfg)" '^[[[:space:]]*local[[[:space:]]+[0-9]' 2 127.0.0.1; }
_sipc_port() { core_cfg_field "$(_sipc_cfg)" '^[[[:space:]]*local[[[:space:]]+[0-9]' 3 5060; }
_sipc_mncc() { core_cfg_field "$(_sipc_cfg)" '^[[[:space:]]*socket-path[[[:space:]]]' 2 /tmp/mscc

_sipc_mncc_connected() {
    local p; p="$(_sipc_mncc)"
    ss -xH 2>/dev/null | grep -F -- "$p" | grep -q 'ESTAB\|CONNECTED'
}

mod_sip_connector_check() {
    core_bin osmo-sip-connector >/dev/null || {
        mod_hint "installez osmo-sip-connector, ou désactivez la voix : CORE_VOICE=0"
        mod_fail "binaire osmo-sip-connector introuvable"; return $MOD_RC_FAIL; }
    [ -r "$(_sipc_cfg)" ] || {
        mod_fail "configuration illisible : $(_sipc_cfg)"; return $MOD_RC_FAIL; }
    core_ip_local "$(_sipc_ip)" || {
        mod_hint "corrigez « sip / local » dans $(_sipc_cfg)"
        mod_fail "adresse SIP $(_sipc_ip) portée par aucune interface"; return $MOD_RC_FAIL; }
}

```

```

    mod_ok
}

mod_sip_connector_status() { core_unit_active "$SIPC_UNIT" || have_proc "osmo-sip-connector"

mod_sip_connector_start() {
    core_svc_start "$SIPC_UNIT" "$(core_bin osmo-sip-connector)" -c "$(_sipc_cfg)" \
        || { mod_fail "systemctl start $SIPC_UNIT a échoué"
            mod_hint "journalctl -u $SIPC_UNIT -n 30"; return $MOD_RC_FAIL; }
    mod_ok
}

mod_sip_connector_wait() {
    local to="${MOD_TIMEOUT[sip-connector]}" ip port
    ip="$(_sipc_ip)"; port="$(_sipc_port)"

    if ! wait_until "$to" "bind SIP ($ip:$port)" core_udp_listen "$port" "$ip"; then
        mod_hint "port $port déjà pris (Asterisk ?) : ss -lun | grep $port"
        return $MOD_RC_FAIL
    fi
    if ! wait_until "$to" "connexion MNCC ($(_sipc_mncc))" _sipc_mncc_connected; then
        mod_hint "le MSC expose-t-il « mncc external $_sipc_mncc » ? ss -x | grep $_sipc_mncc"
        return $MOD_RC_FAIL
    fi
    if core_restarted_since "$SIPC_UNIT"; then
        mod_hint "journalctl -u $SIPC_UNIT -n 50 : le service redémarre en boucle"
        mod_fail "osmo-sip-connector a redémarré depuis le lancement"
        return $MOD_RC_FAIL
    fi
    mod_ok
}

mod_sip_connector_stop() { core_svc_stop "$SIPC_UNIT" "osmo-sip-connector -c"; }

```

0.3.28 /opt/GSM/qosmo-overlay/run_modules/21-abonnes-hlr.sh

4107 octets, 94 lignes → 94 lignes

```

: "${MODDIR:=$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)}"
. "$MODDIR/_lib/core.sh"

```

```

MOD_REGISTER abones-hlr "Cœur - abonnés provisionnés dans le HLR"
MOD_REQUIRED[abones-hlr]=0
MOD_DEPS[abones-hlr]="hls"
MOD_TIMEOUT[abones-hlr]=30
MOD_ENABLED_IF[abones-hlr]='[ "${NO_OSMO_START:-0}" != 1 ]'

: "${HLR_VTY_PORT:=4258}"
: "${HLR_DB:=/var/lib/osmocom/hlr.db}"
: "${OPERATOR_ID:=1}"
: "${N_MS:=1}"

_abo_mcc() { printf '%s\n' "${MCC:-$(core_cfg_field "$(core_cfg osmo-msc)" '^[:space:]*netw
_abo_mnc() { printf '%s\n' "${MNC:-$(core_cfg_field "$(core_cfg osmo-msc)" '^[:space:]*mob

_abo_imsi() { printf '%s%04d%06d\n' "$(_abo_mcc)" "$(_abo_mnc)" "$OPERATOR_ID" "$1"; }
_abo_ki() { printf '00112233445566778899aabbccdd%02x%02x\n' "$1" "$OPERATOR_ID"; }

_abo_node() {
    local n="${OSMO_WAN_NODE:-${WAN_NODE_ID:-}}"
    [ -n "$n" ] || n="$(sed -n 's/^PLAN_NODE=//p' "${OSMOCOM_CFG:-/etc/osmocom}/radio-plan.er
    case "$n" in [1-9]) ;; *) n=1 ;; esac
    printf '%s' "$n"
}

_abo_msisdn() { echo $(( $(_abo_node) * 100000 + OPERATOR_ID * 100 + $1 )); }

_abo_present() {
    local imsi="$1" n=""
    if [ -r "$HLR_DB" ] && command -v sqlite3 >/dev/null 2>&1; then
        n="$(sqlite3 -readonly "$HLR_DB" "select count(*) from subscriber where imsi='$imsi'
        case "$n" in ''|*[*!0-9]*) n="" ;; esac
        if [ -n "$n" ]; then [ "$n" -ge 1 ]; return $?; fi
    fi
    core_vty_ask "$HLR_VTY_PORT" "enable" "subscriber imsi $imsi show" 2>/dev/null | grep -q
}

mod_abones_hlr_check() {
    case "$N_MS" in ''|*[*!0-9]*) mod_fail "N_MS invalide : « $N_MS »"; return $MOD_RC_FAIL ;
    [ "$N_MS" -ge 1 ] || { mod_skip "N_MS=$N_MS : aucun abonné à créer"; return $MOD_RC_SKIP
    core_vty_listen "$HLR_VTY_PORT" || {
        mod_hint "le HLR doit être prêt : ./run.sh --only hlr"
        mod_fail "VTY HLR ($HLR_VTY_PORT) injoignable"; return $MOD_RC_FAIL; }
    command -v socat >/dev/null 2>&1 || command -v telnet >/dev/null 2>&1 || {

```

```

        mod_hint "installez socat (ou telnet) : le provisionnement passe par le VTY"
        mod_fail "ni socat ni telnet - impossible d'écrire dans le HLR"; return $MOD_RC_FAIL
    mod_ok
}

_abo_msisdn_now() {
    local imsi="$1" v=""
    if [ -r "$HLR_DB" ] && command -v sqlite3 >/dev/null 2>&1; then
        v="$(sqlite3 -readonly "$HLR_DB" "select msisdn from subscriber where imsi='$imsi';"
        [ -n "$v" ] && { printf '%s' "$v"; return 0; }
    fi
    core_vty_ask "$HLR_VTY_PORT" "enable" "subscriber imsi $imsi show" 2>/dev/null \
        | sed -n 's/.*MSISDN: *\([0-9][0-9]*\)*/\1/p' | head -1
}

mod_abonnes_hlr_status() {
    local imsi; imsi="$(_abo_imsi "$N_MS")"
    _abo_present "$imsi" || return 1
    [ "$(_abo_msisdn_now "$imsi")" = "$(_abo_msisdn "$N_MS")" ]
}

mod_abonnes_hlr_start() {
    local m cmds=() imsi msisdn
    cmds+=("enable")
    for m in $(seq 1 "$N_MS"); do
        imsi="$(_abo_imsi "$m")"
        msisdn="$(_abo_msisdn "$m")"
        cmds+=("subscriber imsi $imsi create")
        cmds+=("subscriber imsi $imsi update msisdn $msisdn")
        cmds+=("subscriber imsi $imsi update aud2g comp128v1 ki "$(_abo_ki "$m")")
    done
    cmds+=("exit")
    mod_say "provisionnement de $N_MS abonné(s), nœud "$_abo_node), opérateur $OPERATOR_ID, I
    core_vty_ask "$HLR_VTY_PORT" "${cmds[@]}" || {
        mod_hint "vérifiez le VTY : socat STDIO TCP:127.0.0.1:$HLR_VTY_PORT,crlf"
        mod_fail "le dialogue VTY avec le HLR n'a rien retourné"; return $MOD_RC_FAIL; }
    mod_ok
}

mod_abonnes_hlr_wait() {
    local imsi; imsi="$(_abo_imsi "$N_MS")"
    if ! wait_until "${MOD_TIMEOUT[abonnes-hlr]}" "abonné $imsi présent dans le HLR" _abo_pr

```

```

        mod_hint "essayez à la main : subscriber imsi $imsi show (VTY $HLR_VTY_PORT) ; base =
        return $MOD_RC_FAIL
    fi
    mod_ok
}

mod_abonnes_hlr_stop() { return 0; }

```

0.3.29 /opt/GSM/qosmo-overlay/run_modules/22-smsc.sh

2866 octets, 68 lignes → 68 lignes

```

: "${MODDIR:=$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)}"
. "$MODDIR/_lib/core.sh"

MOD_REGISTER smsc "Cœur - passerelle SMS (proto-SMSC)"
MOD_REQUIRED[smc]=0
MOD_DEPS[smc]="hlr"
MOD_TIMEOUT[smc]=60
MOD_ENABLED_IF[smc]='[ "${NO_OSMO_START:-0}" != 1 ] && [ "${CORE_SMS:-1}" = 1 ]'

: "${OPERATOR_ID:=1}"
: "${SMSC_SCRIPT:=/etc/osmocom/smsc-start.sh}"
: "${SMSC_RELAY_SCRIPT:=/etc/osmocom/sms-interop-relay.py}"
: "${SMSC_RELAY_PORT:=7890}"
: "${SMSC_SENDMT_SOCKET:=/tmp/sendmt_socket}"
: "${SMSC_HLR_IP:=127.0.0.2}"

mod_smsc_check() {
    command -v proto-smc-daemon >/dev/null 2>&1 || {
        mod_hint "sans proto-smc-daemon les SMS passent uniquement par le VTY du MSC ; CORE
        mod_skip "proto-smc-daemon absent"; return $MOD_RC_SKIP; }
    [ -x "$SMSC_SCRIPT" ] || {
        mod_hint "déployez les scripts : cp scripts/smsc-start.sh $OSMOCOM_CFG/"
        mod_fail "script de démarrage absent : $SMSC_SCRIPT"; return $MOD_RC_FAIL; }
    core_tcp_listen 4222 "$SMSC_HLR_IP" || {
        mod_hint "le HLR doit écouter en GSUP : ./run.sh --only hlr"
        mod_fail "GSUP HLR ($SMSC_HLR_IP:4222) injoignable"; return $MOD_RC_FAIL; }
    mod_ok
}

```

```

mod_smsc_status() { have_proc 'proto-smsc-daemon'; }

mod_smsc_start() {
    mkdir -p "$RUN_DIR" "$LOG_DIR" /var/log/osmocom 2>/dev/null || true
    pkill -f 'proto-smsc-daemon' 2>/dev/null
    local log="$LOG_DIR/smsc-op${OPERATOR_ID}.log"
    mod_say "journal : $log ; SMS MO reçus : /var/log/osmocom/mo-sms-op${OPERATOR_ID}.log"
    setsid env OPERATOR_ID="$OPERATOR_ID" HLR_IP="$SMSC_HLR_IP" RELAY_PORT="$SMSC_RELAY_PORT"
        stdbuf -oL -eL bash "$SMSC_SCRIPT" >"$log" 2>&1 </dev/null &
    printf '%s\n' "$!" > "$RUN_DIR/smsc-op${OPERATOR_ID}.pid"
    mod_ok
}

mod_smsc_wait() {
    local to="${MOD_TIMEOUT[smsc]}"

    if ! wait_until "$to" "socket d'envoi MT ($SMSC_SENDMT_SOCKET)" core_unix_listen "$SMSC_SENDMT_SOCKET"
        mod_hint "voir $LOG_DIR/smsc-op${OPERATOR_ID}.log : le daemon n'a pas pu se connecter"
        return $MOD_RC_FAIL
    fi
    if [ -f "$SMSC_RELAY_SCRIPT" ]; then
        if ! wait_until "$to" "relay SMS interop (TCP $SMSC_RELAY_PORT)" core_tcp_listen "$SMSC_RELAY_PORT"
            mod_hint "port $SMSC_RELAY_PORT déjà pris, ou sms-routing.conf illisible : voir $LOG_DIR/smsc-relay.log"
            return $MOD_RC_FAIL
        fi
    else
        mod_say "$SMSC_RELAY_SCRIPT absent - pas de relay inter-opérateurs, barrière non appliquée"
    fi
    mod_ok
}

mod_smsc_stop() {
    local pf="$RUN_DIR/smsc-op${OPERATOR_ID}.pid"
    [ -f "$pf" ] && { kill "$(cat "$pf" 2>/dev/null)" 2>/dev/null; rm -f "$pf"; }
    pkill -f 'proto-smsc-daemon' 2>/dev/null
    pkill -f 'sms-interop-relay.py' 2>/dev/null
    rm -f "$SMSC_SENDMT_SOCKET" 2>/dev/null
    return 0
}

```

0.3.30 /opt/GSM/qosmo-overlay/run_modules/25-audio.sh

2720 octets, 75 lignes → 75 lignes

```
MOD_REGISTER audio "Pont audio (PulseAudio + GAPK)"
MOD_REQUIRED[audio]=0
MOD_TIMEOUT[audio]=25

MOD_ENABLED_IF[audio]='[ "${AUDIO:-1}" = 1 ]'

: "${EGPRS_DIR:=${NITB_ROOT}}"
: "${AUDIO_LIB:=${EGPRS_DIR}/lib/audio.sh}"
: "${AUDIO_SINK:=gsm_audio}"
: "${AUDIO_TMUX_SESSION:=gapk}"

_audio_sink_present() {
    pactl list short sinks 2>/dev/null | grep -q "$AUDIO_SINK"
}

_audio_gapk_vivant() {
    pgrep --ns $$ --nslist mnt -f "[g]apk-start\.sh auto" >/dev/null 2>&1 \
        || tmux has-session -t "${AUDIO_TMUX:-gapk}" 2>/dev/null
}

mod_audio_check() {
    if [ ! -r "$AUDIO_LIB" ]; then
        mod_hint "la bibliothèque est l'extraction de start-direct.sh.legacy L566-722 ; elle
        mod_skip "bibliothèque audio absente ($AUDIO_LIB) : pont audio non monté"
        return $MOD_RC_SKIP
    fi
    local manquant=""
    command -v tmux >/dev/null 2>&1 || manquant="$manquant tmux"
    command -v osmo-gapk >/dev/null 2>&1 || manquant="$manquant osmo-gapk"
    if [ -n "$manquant" ]; then
        mod_hint "installez-les pour avoir le son :$manquant - ou AUDIO=0 pour ne plus l'ess
        mod_skip "outillage audio absent :$manquant"
        return $MOD_RC_SKIP
    fi
    mod_ok
}

mod_audio_status() { _audio_gapk_vivant && _audio_sink_present; }

mod_audio_start() {
```

```

local retour="$PWD" rc=0
cd "$EGPRS_DIR" || { mod_fail "impossible d'entrer dans $EGPRS_DIR"; return $MOD_RC_FAIL
HERE="$EGPRS_DIR"
. "$AUDIO_LIB" || { cd "$retour"; mod_fail "bibliothèque illisible : $AUDIO_LIB"; return

mod_say "sink visé : $AUDIO_SINK · session tmux : $AUDIO_TMUX_SESSION"
ensure_gapk || rc=$?
cd "$retour"
[ "$rc" -eq 0 ] || { mod_fail "ensure_gapk a rendu $rc"; return $MOD_RC_FAIL; }
mod_ok
}

mod_audio_wait() {
    wait_until "${MOD_TIMEOUT[audio]}" "démon PulseAudio" pactl info || {
        mod_hint "regardez $LOG_DIR/pulse-system.log : le démon système n'a pas démarré"
        mod_fail "PulseAudio ne répond pas"
        return $MOD_RC_FAIL
    }
    wait_until "${MOD_TIMEOUT[audio]}" "sink $AUDIO_SINK" _audio_sink_present || {
        mod_hint "pactl load-module module-null-sink sink_name=$AUDIO_SINK"
        mod_fail "le sink $AUDIO_SINK n'est pas enregistré"
        return $MOD_RC_FAIL
    }
    wait_until "${MOD_TIMEOUT[audio]}" "superviseur gapk" _audio_gapk_vivant || {
        mod_hint "regardez $LOG_DIR/gapk-auto.log : le superviseur s'est arrêté aussitôt lan
        mod_fail "le superviseur gapk ne tourne pas"
        return $MOD_RC_FAIL
    }
    mod_ok
}

mod_audio_stop() {
    tmux kill-session -t "$AUDIO_TMUX_SESSION" 2>/dev/null
    pkill -x osmo-gapk 2>/dev/null
    return 0
}

```

0.3.31 /opt/GSM/qosmo-overlay/run_modules/35-tmux.sh

2009 octets, 53 lignes → 53 lignes

```

MOD_REGISTER tmux "Session d'observation tmux"
MOD_REQUIRED[tmux]=0
MOD_DEPS[tmux]="logs"
MOD_TIMEOUT[tmux]=10
MOD_ENABLED_IF[tmux]='command -v tmux >/dev/null 2>&1'

: "${TMUX_SESSION:=calypso}"
: "${TMUX_MODULES_DIR:=$(cd "$(dirname "${BASH_SOURCE[0]}")/../../tmux_modules" 2>/dev/null &&

mod_tmux_check() {
    command -v tmux >/dev/null 2>&1 || {
        mod_hint "apt-get install tmux - ou lancez sans : les services tournent quand même"
        mod_fail "tmux introuvable"; return $MOD_RC_FAIL; }
    mod_ok
}

mod_tmux_status() { tmux has-session -t "$TMUX_SESSION" 2>/dev/null; }

_tmux_charger() {
    local d="${TMUX_MODULES_DIR:-}"
    [ -n "$d" ] && [ -r "$d/_commun.sh" ] && [ -r "$d/bench.sh" ] || {
        mod_hint "attendu : <dépôt>/tmux_modules/_commun.sh et bench.sh"
        mod_fail "disposition tmux introuvable (TMUX_MODULES_DIR=${d:-<vide>})"
        return $MOD_RC_FAIL; }
    . "$d/_commun.sh"
    . "$d/bench.sh"
    mod_say "disposition : bench.sh"
    return 0
}

mod_tmux_start() {
    _tmux_charger || return $MOD_RC_FAIL
    tmux new-session -d -s "$TMUX_SESSION" -n "${TMUX_FENETRE_PREMIERE:-radio}" \
        "${tmux_layout_premiere}" 2>/dev/null || {
        mod_hint "un serveur tmux périmé peut refuser la session : tmux kill-server puis rel
        mod_fail "création de la session « $TMUX_SESSION » refusée"; return $MOD_RC_FAIL; }
    _tmux_env_repropre
    tmux_layout
    _tmux_style
    tmux select-window -t "$TMUX_SESSION:${TMUX_FENETRE_PREMIERE:-radio}" 2>/dev/null
    mod_say "${TMUX_RESUME:-session créée}"
    mod_ok
}

```

```

}

mod_tmux_wait() {
    wait_until "${MOD_TIMEOUT[tmux]}" "session tmux < $TMUX_SESSION >" \
        tmux has-session -t "$TMUX_SESSION" || return $MOD_RC_FAIL
    [ "$(tmux list-windows -t "$TMUX_SESSION" 2>/dev/null | wc -l)" -ge 1 ] || {
        mod_fail "session créée mais aucune fenêtre"; return $MOD_RC_FAIL; }
    mod_ok
}

mod_tmux_stop() { tmux kill-session -t "$TMUX_SESSION" 2>/dev/null; return 0; }

```

0.3.32 /opt/GSM/qosmo-overlay/run_modules/38-bruit.sh

9884 octets, 201 lignes → 201 lignes

```

# 38-bruit.sh - injecteur de bruit sur le sens DESCENDANT. INACTIF PAR DEFAULT.
#
# [2026-10-03] Intercale tools/injecteur_bruit.py entre le pont et la couche 1
# du QEMU de CE plan, quand l'operateur pose BRUIT_MODE (ou BRUIT_SNR_DB seul).
# Numerote 38 : avant 40-qemu, parce que ce module pose dans l'environnement
# des variables que QEMU doit heriter (CALYPSO_BSP_PORT, CALYPSO_BSP_SNR_DB).
#
#   BRUIT_MODE      ber | souple | iq | relais (vide = module desactive)
#                   ber      bits inverses, taux BRUIT_BER (BRUIT_RAFALES=L :
#                   souple   AWGN a BRUIT_SNR_DB puis decision dure (le flux
#                   iq       EXPERIMENTAL : bits -> I/Q GMSK + AWGN (BRUIT_SNR_DB)
#                   relais   rien d'altere : verifie l'intercalation seule
#   BRUIT_BER       taux d'inversion, 0..1 (mode ber)
#   BRUIT_RAFALES   longueur moyenne d'une rafale, bits (mode ber, optionnel)
#   BRUIT_SNR_DB    SNR en dB, P/(2 sigma^2) par echantillon (souple, iq)
#   POSE SANS BRUIT_MODE : pas d'injecteur ; le module exporte
#   CALYPSO_BSP_SNR_DB (AWGN du coeur C54x, calypso_c54x.c
#   c54x_bsp_load), qui n'agit que sur un QEMU construit avec
#   --enable-l1-dsp (le DSP externe, c54x_exe, le recoit de
#   c54x_exe/run.sh).
#   BRUIT_PERTE    probabilite d'effacer un burst (dsp) / un bloc (grgsm)
#   BRUIT_TN       intervalles a degrader, ex. 0 ou 1,2 (defaut tous)
#   BRUIT_GRAINE   graine : meme graine + memes trames = memes erreurs ; sans

```

```

#          BRUIT_MODE, devient CALYPSO_BSP_BRUIT_GRAINE
# BRUIT_STATS    periode des statistiques dans bruit.log (10 s)
# BRUIT_OPTS     options supplementaires passees telles quelles
# BRUIT_CIBLE    auto (default) | dsp | grgsm : couche 1 du QEMU ; auto lit
#                build/meson-info (l1_dsp / l1_grgsm)
# BRUIT_PORT_GRGSM port ou l'injecteur ecoute en grgsm (14730)
# BRUIT_PORT_DSP   port du BSP deplace en dsp (16702)
# BRUIT_PY         chemin de l'injecteur (default <QEMU_TREE>/tools/injecteur_bruit.py)
#
# OU IL S'INTERCALE
# L1 gr-gsm (le build de ce banc) : QEMU ecoute GSMTAP sur 127.0.0.1:4730 en
#   dur (calypso_l1_grgsm.c). C'est le pont qui change de cible : pont/pont.py
#   vise PONT_GSMTAP_PORT, que lui-meme met a BRUIT_PORT_GRGSM (14730) quand
#   BRUIT_MODE vaut ber ou relais. Le pont est lance par start-native.sh, pas
#   par ce plan : BRUIT_MODE doit donc etre dans l'environnement de
#   start-native.sh (BRUIT_MODE=ber BRUIT_BER=0.01 ./start-native.sh --grgsm).
#   Modes ber et relais seulement (blocs L2 deja decodes, cf. injecteur).
# L1 DSP dans QEMU (--enable-l1-dsp) : le BSP de QEMU ecoute BRUIT_PORT_DSP
#   (CALYPSO_BSP_PORT, exporte ici avant 40-qemu), l'injecteur prend
#   127.0.0.1:6702 et y recoit ce que le pont envoie a --dsp-port 6702.
# Montage DSP de start-native.sh (--dsp) : qemu est retire de ce plan, le
#   module se saute ; c'est c54x_exe/run.sh qui intercale l'injecteur.
# Journal de l'injecteur : $LOG_DIR/bruit.log. Detail :
# osmo-operator/wiki/Injecteur-bruit.md.
MOD_REGISTER bruit "Injecteur de bruit (sens descendant)"
MOD_REQUIRED[bruit]=0
MOD_DEPS[bruit]="logs"
MOD_TIMEOUT[bruit]=10
MOD_ENABLED_IF[bruit]='[ -n "${BRUIT_MODE:-}${BRUIT_SNR_DB:-}" ] '

_TM[fr:bruit]="Injecteur de bruit (sens descendant)"
_TM[en:bruit]="Noise injector (downlink)"

: "${BRUIT_PY:=${QEMU_TREE:-/opt/GSM/qosmo}/tools/injecteur_bruit.py}"
: "${BRUIT_PORT_DSP:=16702}"
: "${BRUIT_PORT_GRGSM:=14730}"
: "${BRUIT_STATS:=10}"

_bruit_log() { printf '%s/bruit.log' "${LOG_DIR:-/tmp/calypso/logs}"; }
_bruit_pidf() { printf '%s/bruit.pid' "${RUN_DIR:-/tmp/calypso}"; }

# La couche 1 du QEMU de ce plan : BRUIT_CIBLE, sinon les options du build.

```

```

_bruit_l1() {
    case "${BRUIT_CIBLE:-auto}" in dsp|grgsm) printf '%s' "$BRUIT_CIBLE"; return 0 ;; esac
    local j
    j="$(dirname "${QEMU_BIN:-/nonexistent}"/meson-info/intro-buildoptions.json"
    python3 -c 'import json, sys
d = {o["name"]: o["value"] for o in json.load(open(sys.argv[1]))}'
print("dsp" if d.get("l1_dsp") == "enabled" else "grgsm", end="")' "$j" 2>/dev/null || print
}

# qemu est-il dans le plan joue ? (start-native.sh --dsp le retire : --skip qemu,...)
_bruit_qemu_au_plan() {
    local m
    for m in "${selected[@]:-}"; do [ "$m" = qemu ] && return 0; done
    return 1
}

_bruit_args() {
    BRUIT_ARGS=(--mode "$BRUIT_MODE" --etiquette qosmo --stats "$BRUIT_STATS")
    if [ "${_bruit_l1}" = dsp ]; then
        BRUIT_ARGS+=(--cible dsp --ecoute 127.0.0.1:6702 --vers "127.0.0.1:$BRUIT_PORT_DSP")
    else
        BRUIT_ARGS+=(--cible grgsm --ecoute "127.0.0.1:${PONT_GSM_TAP_PORT:-$BRUIT_PORT_GRGSM}"
        --vers "127.0.0.1:${PORT_GSM_TAP_SI:-4730}")
    fi
    [ -n "${BRUIT_BER:-}" ] && BRUIT_ARGS+=(--ber "$BRUIT_BER")
    [ -n "${BRUIT_RAFALES:-}" ] && BRUIT_ARGS+=(--rafales "$BRUIT_RAFALES")
    [ -n "${BRUIT_SNR_DB:-}" ] && BRUIT_ARGS+=(--snr-db "$BRUIT_SNR_DB")
    [ -n "${BRUIT_PERTE:-}" ] && BRUIT_ARGS+=(--perte "$BRUIT_PERTE")
    [ -n "${BRUIT_TN:-}" ] && BRUIT_ARGS+=(--tn "$BRUIT_TN")
    [ -n "${BRUIT_GRAINE:-}" ] && BRUIT_ARGS+=(--graine "$BRUIT_GRAINE")
    # shellcheck disable=SC2206 # BRUIT_OPTS : decoupage en mots voulu
    [ -n "${BRUIT_OPTS:-}" ] && BRUIT_ARGS+=($BRUIT_OPTS)
    return 0
}

_bruit_vivant() {
    [ -n "${BRUIT_MODE:-}" ] || return 1
    local pid; pid="$(cat "$_bruit_pidf" 2>/dev/null || echo 0)"
    [ "$pid" != 0 ] && kill -0 "$pid" 2>/dev/null \
        && tr '\0' ' ' < "/proc/$pid/cmdline" 2>/dev/null | grep -q 'injecteur_bruit\.py'
}

```

```

mod_bruit_check() {
    if ! _bruit_qemu_au_plan; then
        mod_skip "qemu hors du plan (montage DSP de start-native.sh) : c54x_exe/run.sh inter
        return $MOD_RC_SKIP
    fi
    [ -n "${BRUIT_MODE:-}" ] || { mod_ok; return $MOD_RC_OK; }      # BRUIT_SNR_DB seul : ri
    command -v python3 >/dev/null 2>&1 || { mod_fail "python3 introuvable"; return $MOD_RC_F
    [ -r "$BRUIT_PY" ] || {
        mod_hint "posez BRUIT_PY=<chemin de injecteur_bruit.py>"
        mod_fail "injecteur introuvable : $BRUIT_PY"; return $MOD_RC_FAIL; }
    local l1; l1="$_bruit_l1"
    if [ "$l1" = grgsm ]; then
        case "$BRUIT_MODE" in
            ber|relais) ;;
            *) mod_hint "en gr-gsm, QEMU recoit des blocs L2 deja decodes : BRUIT_MODE=ber (
               mod_fail "BRUIT_MODE=$BRUIT_MODE impossible avec la L1 gr-gsm (le pont n'est p
               return $MOD_RC_FAIL ;;
        esac
    if [ "${PONT_GSM_TAP_PORT:-$BRUIT_PORT_GRGSM}" = "${PORT_GSM_TAP_SI:-4730}" ]; then
        mod_hint "laissez PONT_GSM_TAP_PORT vide (14730 par default) ou posez BRUIT_PORT_G
        mod_fail "l'injecteur ecouterait le port de QEMU (${PORT_GSM_TAP_SI:-4730})"
        return $MOD_RC_FAIL
    fi
fi
bruit_args
mod_say "verification : python3 $BRUIT_PY ${BRUIT_ARGS[*]} --verifier"
local out
if ! out="$(python3 "$BRUIT_PY" "${BRUIT_ARGS[@]}" --verifier 2>&1)"; then
    mod_say "$out"
    mod_fail "$(printf '%s\n' "$out" | tail -1)"
    return $MOD_RC_FAIL
fi
mod_say "$out"
mod_ok
}

mod_bruit_status() { _bruit_vivant; }

mod_bruit_start() {
    local l1; l1="$_bruit_l1"
    if [ -z "${BRUIT_MODE:-}" ]; then
        # BRUIT_SNR_DB seul : le bruit « physique » du coeur C54x, en C.

```

```

export CALYPSO_BSP_SNR_DB="${CALYPSO_BSP_SNR_DB:-$BRUIT_SNR_DB}"
[ -n "${BRUIT_GRAINE:-}" ] && export CALYPSO_BSP_BRUIT_GRAINE="${CALYPSO_BSP_BRUIT_GRAINE:-$BRUIT_GRAINE}"
if [ "$1" = dsp ]; then
    mod_say "CALYPSO_BSP_SNR_DB=$CALYPSO_BSP_SNR_DB exporte pour QEMU (L1 DSP) : AWG"
else
    mod_say "CALYPSO_BSP_SNR_DB=$CALYPSO_BSP_SNR_DB exporte, mais la L1 de ce QEMU est"
    mod_say " (bruit physique = coeur C54x : montage DSP, c54x_exe/run.sh ; ici : BL"
fi
mod_ok; return $MOD_RC_OK
fi
local log; log="$(_bruit_log)"
mkdir -p "$(dirname "$log")" "$(dirname "$(_bruit_pidf)")" 2>/dev/null
: > "$log"
_bruit_args
if [ "$1" = dsp ]; then
    export CALYPSO_BSP_PORT="$BRUIT_PORT_DSP"
    mod_say "CALYPSO_BSP_PORT=$CALYPSO_BSP_PORT exporte : le BSP de QEMU y ecouterait, l'injecteur"
else
    mod_say "L1 gr-gsm : pont -> 127.0.0.1:${PONT_GSM_TAP_PORT:-$BRUIT_PORT_GRGSM} (injecteur)"
    mod_say " le pont doit viser ce port : BRUIT_MODE dans l'environnement de start-nat"
fi
mod_say "commande : python3 -u $BRUIT_PY ${BRUIT_ARGS[*]}"
mod_say "journal : $log"
setsid python3 -u "$BRUIT_PY" "${BRUIT_ARGS[@]}" >>"$log" 2>&1 </dev/null &
printf '%s\n' "$!" > "$(_bruit_pidf)"
mod_ok
}

mod_bruit_wait() {
    [ -n "${BRUIT_MODE:-}" ] || { mod_ok; return $MOD_RC_OK; }
    local log; log="$(_bruit_log)"
    wait_until "${MOD_TIMEOUT[bruit]}" "injecteur pret" log_has "$log" "PRET ecoute=" || {
        modb_tail "$log" 15
        mod_hint "lisez $log"
        return $MOD_RC_FAIL; }
    _bruit_vivant || {
        modb_tail "$log" 15
        mod_fail "l'injecteur s'est arrete aussitot lance"
        return $MOD_RC_FAIL; }
    mod_say "$(grep -a 'PRET ecoute=' "$log" | tail -1)"
    mod_ok
}

```

```

mod_bruit_stop() {
    local pid; pid="$(cat "$(_bruit_pidf)" 2>/dev/null || echo 0)"
    [ "$pid" != 0 ] && kill "$pid" 2>/dev/null
    pkill -f -- "injecteur_bruit\py.*--etiquette qosmo" 2>/dev/null
    rm -f "$(_bruit_pidf)"
    return 0
}

```

0.3.33 /opt/GSM/qosmo-overlay/run_modules/40-qemu.sh

6932 octets, 144 lignes → 144 lignes

```

MOD_REGISTER qemu "Émulateur Calypso (QEMU)"
MOD_REQUIRED[qemu]=1
MOD_DEPS[qemu]="prereqs"
MOD_TIMEOUT[qemu]=30

: "${QEMU_LOG_MAX:=${((64 * 1024 * 1024))}}"
: "${QEMU_LOG_HEAD:=${((8 * 1024 * 1024))}}"

mod_qemu_check() {
    [ -x "${QEMU_BIN:-}" ] || { mod_fail "binaire QEMU absent : ${QEMU_BIN:-<non défini>}"
                                mod_hint "compilez-le : ./configure --target-list=arm-softmmu"
                                return $MOD_RC_FAIL; }
    [ -r "${FIRMWARE_ELF:-}" ] || { mod_fail "firmware ARM introuvable : ${FIRMWARE_ELF:-<non défini>}"
    mod_ok
}

mod_qemu_status() { have_proc "qemu-system-arm.*calypso"; }

_qemu_save_head() {
    local src="$1" dst="$2" taille="$3" i=0
    while [ "$i" -lt 120 ]; do
        if [ -s "$src" ] && [ "$(stat -c %s "$src" 2>/dev/null || echo 0)" -ge "$taille" ]; then
            head -c "$taille" "$src" > "$dst" 2>/dev/null
            return 0
        fi
        sleep 1; i=$(( i + 1 ))
    done
    [ -s "$src" ] && head -c "$taille" "$src" > "$dst" 2>/dev/null
    return 0
}

```

```

}

_qemu_log_guard() {
    local f="$1" max="$2" qpid="$3" n=0
    while kill -0 "$qpid" 2>/dev/null; do
        if [ -f "$f" ] && [ "$(stat -c %s "$f" 2>/dev/null || echo 0)" -gt "$max" ]; then
            n=$((n+1))
            printf '\n--- journal remis à zéro (%d fois) : plafond de %s octets atteint.\n--'
                "$n" "$max" > "$f"
        fi
        sleep 5
    done
}

mod_qemu_start() {
    local qlog="${LOG_DIR}/qemu.log" qpid
    mod_say "machine : calypso"
    mod_say "journal : $qlog (plafond ${QEMU_LOG_MAX} o)"
    # [2026-09-03] LANCEUR C `qosmo-grgsm` (tools/qosmo-launch, installe dans
    # /usr/local/bin par `make install`). Meme ligne de commande que ci-dessous
    # (-M calypso, -cpu arm946, -serial pty x2, -monitor, -kernel), lecture de
    # lls/last_rach dans l'ELF, relais de stdout+stderr de QEMU dans qemu.log
    # (41-pty continue de lire « redirected to /dev/pts/N »), et deux liens
    # stables : $RUN_DIR/modem.ptty (serial0, celui d'osmocon) et $RUN_DIR/irda.ptty.
    # Il transmet SIGTERM a QEMU et meurt avec lui. Sans lanceur : ligne historique.
    # [2026-09-03] gdbstub ARM ACTIF ici aussi (tcp::$CALYPSO_GDB_PORT, 1234) :
    # QEMU ne s'arrete que si un client s'y connecte, donc aucun cout tant qu'on
    # ne s'en sert pas. C'est ce que 44-gdb-telnet.sh attache a la demande
    # (telnet localhost 44444, cible en marche). CALYPSO_GDB_PORT=off pour couper.
    : "${CALYPSO_GDB_PORT:=1234}"
    # [2026-09-05] -display none -parallel none, ICI COMME DANS LE LANCEUR.
    # La machine calypso n'a pas d'ecran. Sans -display, QEMU prend son defaut
    # graphique (SDL est compile) et cable ses peripheriques par defaut sur des
    # consoles virtuelles ; les deux serie etant prises (-serial pty) et le
    # moniteur aussi, il ne restait que le port parallele : une fenetre SDL vide
    # intitulee « parallel0 », qui capte le clavier, pendant que le boot se passe
    # sur les pty. -parallel none supprime en plus le chardev (`info chardev` le
    # montrait encore en `vc` avec le seul -display none).
    # icount / trace in_asm, désactivés par défaut (./run.sh --icount / --asm) :
    # QEMU_ICOUNT=auto|shift=N,... -> -icount
    # QEMU_ASM=1|FICHIER -> -d in_asm -D FICHIER (défaut $LOG_DIR/qemu-asm.log)
    # QEMU_ASM_FLAGS=... remplace in_asm (in_asm,exec,nochain : chaque execution

```

```

# QEMU_ASM_FILTRE=...          ajoute -dfilter
# Passés après `--` au lanceur : marche aussi avec un lanceur installé plus ancien.
local -a qx=(
case "${QEMU_ICOUNT:-}" in ''|0|off) ;; *) qx+=(-icount "$QEMU_ICOUNT") ;; esac
case "${QEMU_ASM:-}" in
    ''|0|off) ;;
    1) qx+=(-d "${QEMU_ASM_FLAGS:-in_asm}" -D "${LOG_DIR}/qemu-asm.log") ;;
    *) qx+=(-d "${QEMU_ASM_FLAGS:-in_asm}" -D "$QEMU_ASM") ;;
esac
[ -n "${QEMU_ASM:-}" ] && [ -n "${QEMU_ASM_FILTRE:-}" ] && qx+=(-dfilter "$QEMU_ASM_FILTRE")
[ "${#qx[@]}" -gt 0 ] && mod_say "extra      : ${qx[*]}"
local launcher="${QOSMO_LAUNCHER:-/usr/local/bin/qosmo}"
if [ -x "$launcher" ]; then
    mod_say "lanceur   : $launcher (QOSMO_LAUNCHER) ; gdbstub ARM : $CALYPSO_GDB_PORT"
    setsid "$launcher" --qemu "$QEMU_BIN" -k "$FIRMWARE_ELF" --bin "$FIRMWARE_BIN" \
        --cpu arm946 --gdb "$CALYPSO_GDB_PORT" --rundir "$RUN_DIR" \
        --monitor "${RUN_DIR}/qemu-monitor.sock" -- "${qx[@]}" >>"$qlog" 2>&1 </dev/null &
    qpid=$!
else
    mod_say "lanceur   : absent ($launcher) - ligne qemu-system-arm directe ; installez-le"
    setsid "$QEMU_BIN" -M calypso -cpu arm946 \
        -display none -parallel none \
        -serial pty -serial pty \
        -monitor "unix:${RUN_DIR}/qemu-monitor.sock,server,nowait" \
        -kernel "$FIRMWARE_ELF" "${qx[@]}" >>"$qlog" 2>&1 </dev/null &
    qpid=$!
fi
printf '%s\n' "$qpid" > "${RUN_DIR}/qemu.pid"
( trap '' HUP; _qemu_save_head "$qlog" "${LOG_DIR}/qemu-tete.log" "${QEMU_LOG_HEAD}" ) >
( trap '' HUP; _qemu_log_guard "$qlog" "$QEMU_LOG_MAX" "$qpid" ) >/dev/null 2>&1 </dev/null
mod_say "tête       : ${LOG_DIR}/qemu-tete.log"
mod_ok
}

_qemu_running() {
    local sock="$1"
    printf 'info status\n' \
        | timeout 3 socat -t 1 - "UNIX-CONNECT:$sock" 2>/dev/null \
        | grep -q 'VM status: running'
}

mod_qemu_wait() {

```

```

local sock="${RUN_DIR}/qemu-monitor.sock" qlog="${LOG_DIR}/qemu.log"
wait_until "${MOD_TIMEOUT[qemu]}" "socket du moniteur QEMU" have_unix "$sock" || return 1
local qpid; qpid="$(cat "${RUN_DIR}/qemu.pid" 2>/dev/null || echo 0)"
if ! kill -0 "$qpid" 2>/dev/null; then
    modb_tail "$qlog" 20
    mod_hint "cause typique : machine type ou firmware invalide"
    mod_fail "QEMU a démarré puis s'est arrêté"
    return $MOD_RC_FAIL
fi
if command -v socat >/dev/null 2>&1; then
    wait_until "${MOD_TIMEOUT[qemu]}" "machine virtuelle en marche" _qemu_running "$sock"
    modb_tail "$qlog" 20
    mod_hint "socat - UNIX-CONNECT:$sock puis « info status » pour voir l'état de la machine"
    return $MOD_RC_FAIL
}
else
    wait_until "${MOD_TIMEOUT[qemu]}" "PTY série alloués" grep -aq 'label serial1' "$qlog"
    modb_tail "$qlog" 20
    return $MOD_RC_FAIL
}
fi
mod_ok
}

mod_qemu_stop() {
    local qpid; qpid="$(cat "${RUN_DIR}/qemu.pid" 2>/dev/null || echo 0)"
    [ "$qpid" != 0 ] && kill "$qpid" 2>/dev/null
    pkill -f "qemu-system-arm.*calypso" 2>/dev/null
    rm -f "${RUN_DIR}/qemu.pid" "${RUN_DIR}/qemu-monitor.sock"
    return 0
}

```

0.3.34 /opt/GSM/qosmo-overlay/run_modules/41-pty.sh

2592 octets, 75 lignes → 75 lignes

```

MOD_REGISTER pty "PTY série de l'émulateur"
MOD_REQUIRED[pty]=1
MOD_DEPS[pty]="qemu"
MOD_TIMEOUT[pty]=60

```

```

: "${QEMU_MON_SOCKET:=${RUN_DIR:-/tmp/calypso}/qemu-monitor.sock}"

_pty_from_monitor() {
    local sock="$1" label="$2"
    [ -S "$sock" ] || return 1
    command -v socat >/dev/null 2>&1 || return 1
    printf 'info chardev\n' \
        | timeout 3 socat -t 1 - "UNIX-CONNECT:$sock" 2>/dev/null \
        | grep -E "^${label}:" \
        | grep -oE '/dev/pts/[0-9]+' \
        | head -1
}

_pty_from_log() {
    local log="$1" label="$2"
    [ -r "$log" ] || return 1
    grep -a "redirected to /dev/pts/. (label ${label})" "$log" 2>/dev/null \
        | grep -oE '/dev/pts/[0-9]+' \
        | head -1
}

_pty_resolve() {
    local log="${LOG_DIR}/qemu.log" p
    if [ -z "${OSMOCON_PTY:-}" ] || [ ! -c "${OSMOCON_PTY:-/nonexistent}" ]; then
        p="$_pty_from_monitor "$QEMU_MON_SOCKET" serial0)"
        [ -n "$p" ] || p="$_pty_from_log "$log" serial0)"
        [ -n "$p" ] && OSMOCON_PTY="$p"
    fi
    [ -n "${OSMOCON_PTY:-}" ] && [ -c "$OSMOCON_PTY" ]
}

mod_pty_check() {
    if ! command -v socat >/dev/null 2>&1; then
        mod_say "socat absent : lecture du moniteur impossible, secours par le journal seulement"
    fi
    [ -S "$QEMU_MON_SOCKET" ] || [ -r "${LOG_DIR}/qemu.log" ] || {
        mod_hint "le module qemu doit avoir démarré : ./run.sh --only qemu"
        mod_fail "ni moniteur ($QEMU_MON_SOCKET) ni journal QEMU : rien à interroger"
        return $MOD_RC_FAIL
    }
    mod_ok
}

```

```

mod_pty_status() { return 1; }
mod_pty_start() { _pty_resolve; mod_ok; }

mod_pty_wait() {
    local deadline=$(( SECONDS + ${MOD_TIMEOUT[pty]} ))
    local qpid; qpid="$(cat "${RUN_DIR:-/tmp/calypso}/qemu.pid" 2>/dev/null || echo 0)"
    while (( SECONDS < deadline )); do
        if _pty_resolve; then
            export OSMOCON_PTY
            mod_say "serial0=$OSMOCON_PTY"
            mod_ok
            return $MOD_RC_OK
        fi
        if [ "$qpid" != 0 ] && ! kill -0 "$qpid" 2>/dev/null; then
            modb_tail "${LOG_DIR}/qemu.log" 20
            mod_hint "cause typique : machine type ou firmware invalide"
            mod_fail "QEMU (pid $qpid) s'est arrêté avant d'allouer son PTY"
            return $MOD_RC_FAIL
        fi
        sleep 0.5
    done
    mod_hint "vérifiez que QEMU est lancé avec « -serial pty » ; sinon : socat - UNIX-CONNECT"
    mod_fail "aucun PTY serial0 après ${MOD_TIMEOUT[pty]}s (moniteur $QEMU_MON_SOCKET et journal)"
    return $MOD_RC_FAIL
}

mod_pty_stop() { return 0; }

```

0.3.35 /opt/GSM/qosmo-overlay/run_modules/44-gdb-telnet.sh

3805 octets, 71 lignes → 71 lignes

```

# =====
# 44-gdb-telnet - console gdb de l'ARM, servie en telnet, non bloquante
# =====
# RÔLE      Tenir un serveur (tools/gdb-telnet.py) qui, a chaque connexion
#            `telnet localhost 44444`, attache un gdb-multiarch au gdbstub de
#            QEMU (-gdb tcp::$CALYPSO_GDB_PORT) avec la cible EN MARCHE
#            (`continue &`). Ctrl-C arrete l'ARM, `continue &` repart, quit
#            ferme la session : gdb est tue et QEMU reprend l'ARM.
#            Tant que personne n'est connecte, rien ne touche a la cible :

```

```

#          on s'en sert quand on en a envie.
# PRÉREQUIS QEMU lance avec son gdbstub (module qemu ; le lanceur qosmo-dsp
#          exporte CALYPSO_GDB_PORT et CALYPSO_GDB_ELF).
# SUCCÈS    Le port telnet ecoute.
# JOURNAL   $LOG_DIR/gdb-telnet.log
# RÉGLAGES  CALYPSO_GDB_TELNET_PORT (default 44444, 0 = pas de serveur)
#          CALYPSO_GDB_TELNET_BIND (default 127.0.0.1 : une console gdb
#          controle la machine, on n'expose pas ca au reseau sans le vouloir)
# -----
MOD_REGISTER gdbtelnet "Console gdb ARM en telnet (non bloquante)"
MOD_REQUIRED[gdbtelnet]=0
MOD_DEPS[gdbtelnet]="qemu"
MOD_TIMEOUT[gdbtelnet]=10
MOD_ENABLED_IF[gdbtelnet]='[ "${CALYPSO_GDB_TELNET_PORT:-44444}" != 0 ] && [ -n "${CALYPSO_G'

# Le gdbstub QEMU est lui-meme non bloquant : QEMU tourne normalement tant que
# personne n'y est connecte. 1234 = le port que le lanceur (qosmo-dsp/qosmo-grgsm)
# passe a -gdb ; le poser ici garantit que 40-qemu et ce module parlent du meme.
: "${CALYPSO_GDB_PORT:=1234}"
: "${CALYPSO_GDB_TELNET_PORT:=44444}"
: "${CALYPSO_GDB_TELNET_BIND:=127.0.0.1}"
: "${GDB_TELNET_PY:=${QEMU_TREE:-.}/tools/gdb-telnet.py}"

_gdbtel_log() { printf '%s\n' "${LOG_DIR:-/root/calypso/logs}/gdb-telnet.log"; }

mod_gdbtelnet_check() {
    [ -r "$GDB_TELNET_PY" ] || { mod_skip "gdb-telnet.py absent ($GDB_TELNET_PY)"; return $MO
    command -v gdb-multiarch >/dev/null 2>&1 || {
        mod_hint "apt-get install gdb-multiarch - ou CALYPSO_GDB_TELNET_PORT=0"
        mod_fail "gdb-multiarch introuvable"; return $MOD_RC_FAIL; }
    command -v python3 >/dev/null 2>&1 || { mod_fail "python3 introuvable"; return $MOD_RC_F
    mod_ok
}

mod_gdbtelnet_status() { have_proc "gdb-telnet.py" && have_port "$CALYPSO_GDB_TELNET_PORT"; }

mod_gdbtelnet_start() {
    local log; log="$_gdbtel_log"
    mod_say "telnet" : $CALYPSO_GDB_TELNET_BIND:$CALYPSO_GDB_TELNET_PORT (telnet localhost)
    mod_say "gdbstub" : 127.0.0.1:$CALYPSO_GDB_PORT
    mod_say "elf" : ${CALYPSO_GDB_ELF:-${FIRMWARE_ELF:-}}
    setsid python3 "$GDB_TELNET_PY" --port "$CALYPSO_GDB_TELNET_PORT" --bind "$CALYPSO_GDB_T

```

```

        --stub "$CALYPSO_GDB_PORT" --elf "${CALYPSO_GDB_ELF:-${FIRMWARE_ELF:-}}" \
        </dev/null >>"$log" 2>&1 &
    printf '%s\n' "$!" > "${RUN_DIR:-/tmp/calypso}/gdb-telnet.pid"
    mod_ok
}

mod_gdbtelnet_wait() {
    wait_until "${MOD_TIMEOUT[gdbtelnet]}" "port telnet $CALYPSO_GDB_TELNET_PORT have_port"
    mod_hint "regardez $( _gdbtel_log ) - port deja pris ? CALYPSO_GDB_TELNET_PORT=autre"
    mod_fail "le serveur gdb-telnet n'ecoute pas"; return $MOD_RC_FAIL; }
    mod_ok
}

mod_gdbtelnet_stop() {
    local pid; pid="$(cat "${RUN_DIR:-/tmp/calypso}/gdb-telnet.pid" 2>/dev/null || echo 0)"
    [ "$pid" != 0 ] && kill "$pid" 2>/dev/null
    for p in $(pgrep -f 'tools/gdb-telnet.py' 2>/dev/null); do [ "$p" = "$$" ] || kill "$p" 2>/dev/null
    rm -f "${RUN_DIR:-/tmp/calypso}/gdb-telnet.pid"
    return 0
}

```

0.3.36 /opt/GSM/qosmo-overlay/run_modules/50-osmocon.sh

3011 octets, 77 lignes → 77 lignes

```

MOD_REGISTER osmocon "Passerelle osmocon"
MOD_REQUIRED[osmocon]=1
MOD_DEPS[osmocon]="qemu pty"
MOD_TIMEOUT[osmocon]=30

: "${OSMOCON_MODEL:=romload}"
: "${OSMOCON_INTER_BYTE_DELAY:=100}"
: "${OSMOCON_DEBUG:=tr}"
: "${L1CTL_SOCK_PATH:=/tmp/osmocom_12}"

_osmocon_log() { printf '%s\n' "${LOG_DIR}/osmocon.log"; }
_osmocon_vivant() { pgrep -f "osmocon .*-p " >/dev/null 2>&1; }

mod_osmocon_check() {
    [ -x "${OSMOCON:-}" ] || {
        mod_hint "r glez OSMOCOM_BB dans environnement/bench.env, ou compilez osmocon"
    }
}

```

```

    mod_fail "osmocon introuvable : ${OSMOCON:-<non défini>}"; return $MOD_RC_FAIL; }
[ -r "${FIRMWARE_BIN:-}" ] || {
    mod_hint "réglez FIRMWARE_BIN dans environnement/bench.env"
    mod_fail "image firmware illisible : ${FIRMWARE_BIN:-<non défini>}"; return $MOD_RC_FAIL; }
[ -n "${OSMOCON_PTY:-}" ] && [ -c "$OSMOCON_PTY" ] || {
    mod_hint "le module « pty » doit avoir résolu le port série de QEMU ; regardez son journal"
    mod_fail "PTY série indisponible : ${OSMOCON_PTY:-<non résolu>}"; return $MOD_RC_FAIL; }
mod_ok
}

mod_osmocon_status() { _osmocon_vivant && have_unix "$L1CTL SOCK_PATH"; }

mod_osmocon_start() {
    local log; log="$_osmocon_log"
    rm -f "$L1CTL SOCK_PATH" 2>/dev/null
    mod_say "PTY      : $OSMOCON_PTY"
    mod_say "firmware : $FIRMWARE_BIN"
    mod_say "modèle   : $OSMOCON_MODEL"
    mod_say "L1CTL    : $L1CTL SOCK_PATH"
    setuid stdbuf -oL -eL "$OSMOCON" \
        -m "$OSMOCON_MODEL" \
        -i "$OSMOCON_INTER_BYTE_DELAY" \
        -p "$OSMOCON_PTY" \
        -s "$L1CTL SOCK_PATH" \
        "$FIRMWARE_BIN" \
        -d "$OSMOCON_DEBUG" >>"$log" 2>&1 </dev/null &
    printf '%s\n' "$!" > "${RUN_DIR:-/tmp/calypso}/osmocon.pid"
    mod_ok
}

mod_osmocon_wait() {
    local log; log="$_osmocon_log"
    wait_until 5 "processus osmocon" _osmocon_vivant || {
        modb_tail "$log" 15
        mod_hint "PTY $OSMOCON_PTY refusé ? regardez $log"
        mod_fail "osmocon s'est arrêté aussitôt lancé"
        return $MOD_RC_FAIL
    }
    if ! wait_until "${MOD_TIMEOUT[osmocon]}" "socket L1CTL $L1CTL SOCK_PATH" \
        have_unix "$L1CTL SOCK_PATH"; then
        modb_tail "$log" 15
        if _osmocon_vivant; then

```

```

        mod_hint "osmocon vit mais n'a pas fini son romload : modèle inadapté (OSMOCON_M
        mod_fail "aucune socket L1CTL : le firmware n'a pas été chargé"
    else
        mod_hint "regardez $log"
        mod_fail "osmocon est mort pendant le chargement"
    fi
    return $MOD_RC_FAIL
fi
mod_say "firmware chargé, socket L1CTL ouverte"
mod_ok
}

mod_osmocon_stop() {
    local pid; pid="$(cat "${RUN_DIR:-/tmp/calypso}/osmocon.pid" 2>/dev/null || echo 0)"
    [ "$pid" != 0 ] && kill "$pid" 2>/dev/null
    pkill -f "osmocon .*-p " 2>/dev/null
    rm -f "${RUN_DIR:-/tmp/calypso}/osmocon.pid" "$L1CTL_SOCK_PATH"
    return 0
}

```

0.3.37 /opt/GSM/qosmo-overlay/run_modules/58-sidecar-faketrx.sh

2714 octets, 64 lignes → 64 lignes

```

. "$(dirname "${BASH_SOURCE[0]}")/_lib/radio.sh"
. "$(dirname "${BASH_SOURCE[0]}")/_lib/modem.sh"

MOD_REGISTER sidecar-faketrx "Transceiver simulé du BTS#1"
MOD_REQUIRED[sidecar-faketrx]=1
MOD_DEPS[sidecar-faketrx]="sidecar-cfg"
MOD_TIMEOUT[sidecar-faketrx]=20

: "${FAKETRX_PY:=${OSMOCOM_BB:-${GSM_ROOT}/osmocom-bb}/src/target/trx_toolkit/fake_trx.py}"
: "${FAKETRX_PYTHON:=python3}"

mod_sidecar_faketrx_check() {
    command -v "$FAKETRX_PYTHON" >/dev/null 2>&1 || {
        mod_fail "interpréteur introuvable : $FAKETRX_PYTHON"; return $MOD_RC_FAIL; }
    [ -r "$FAKETRX_PY" ] || {
        mod_hint "posez FAKETRX_PY (trx_toolkit/fake_trx.py d'osmocom-bb)"
        mod_fail "fake_trx.py illisible : $FAKETRX_PY"; return $MOD_RC_FAIL; }
}

```

```

local p ports="$SC_TRX_PORT $SC_BB_PORT"
modem_enabled && ports="$ports $MODEM_BB_PORT"
for p in $ports; do
    if radio_udp_bound "$p" && ! radio_alive sidecar-faketrx; then
        mod_hint "ss -uap | grep :$p - arrêtez le tenant, ou décalez SC_TRX_PORT/SC_BB_PORT"
        mod_fail "port UDP $p déjà pris par un autre processus"; return $MOD_RC_FAIL
    fi
done
mod_ok
}

mod_sidecar_faketrx_status() { radio_alive sidecar-faketrx; }

mod_sidecar_faketrx_start() {
    radio_dirs
    local log; log="$(radio_log sidecar-faketrx)"
    mod_say "fake_trx -P $SC_TRX_PORT (BTS#1) -p $SC_BB_PORT (bande de base)"
    # Second MS sur le meme BTS#1 : le modem GPRS. fake_trx multiplexe lui-meme
    # (--trx NOM@ADDR:PORT, transceiver independant) ; trxcon du modem sur ce port.
    local -a extra=()
    if modem_enabled; then
        extra=(--trx "MODEM@${TRX_BB_IP}:${MODEM_BB_PORT}")
        mod_say "second MS (modem GPRS) : --trx ${TRX_BB_IP}:${MODEM_BB_PORT}"
    fi
    mod_say "journal : $log"
    setsid stdbuf -oL -eL "$FAKETRX_PYTHON" "$FAKETRX_PY" \
        -b "$TRX_BIND_IP" -R "$TRX_BTS_IP" -r "$TRX_BB_IP" \
        -P "$SC_TRX_PORT" -p "$SC_BB_PORT" "${extra[@]}" >>"$log" 2>&1 </dev/null &
    radio_save_pid sidecar-faketrx $!
    mod_ok
}

mod_sidecar_faketrx_wait() {
    local log; log="$(radio_log sidecar-faketrx)"
    wait_until "${MOD_TIMEOUT[sidecar-faketrx]}" "port UDP $SC_TRX_PORT (côté BTS#1)" \
        radio_udp_bound "$SC_TRX_PORT" || {
        modb_tail "$log" 20
        mod_hint "fake_trx a-t-il démarré ? $log"
        mod_fail "port $SC_TRX_PORT jamais lié"; return $MOD_RC_FAIL; }
    wait_until 10 "port UDP $SC_BB_PORT (côté bande de base)" \
        radio_udp_bound "$SC_BB_PORT" || {
        mod_fail "côté BTS lié mais pas le côté bande de base ($SC_BB_PORT)"
    }
}

```

```

        return $MOD_RC_FAIL; }
    mod_ok
}

mod_sidecar_faketrx_stop() { radio_kill sidecar-faketrx; return 0; }

```

0.3.38 /opt/GSM/qosmo-overlay/run_modules/59-sidecar-bts.sh

2446 octets, 66 lignes → 66 lignes

```

. "$(dirname "${BASH_SOURCE[0]}")/_lib/radio.sh"
. "$(dirname "${BASH_SOURCE[0]}")/_lib/temps_reel.sh"

MOD_REGISTER sidecar-bts "Station de base BTS#1 (side-car)"
MOD_REQUIRED[sidecar-bts]=1
MOD_DEPS[sidecar-bts]="sidecar-faketrx bsc"
MOD_TIMEOUT[sidecar-bts]=30

: "${SC_BTS_BIN:=osmo-bts-trx}"
: "${SC_BTS_STAB_SECS:=8}"
: "${SC_BTS_RETRIES:=3}"

_sc_bts_mort() {
    grep -qE "No clock since TRX|BTS_SHUTDOWN.*Shutting down" "$(radio_log sidecar-bts)" 2>/dev/null || {
        :
    }
}

mod_sidecar_bts_check() {
    command -v "$SC_BTS_BIN" >/dev/null 2>&1 || {
        mod_fail "binaire introuvable : $SC_BTS_BIN"; return $MOD_RC_FAIL; }
    [ -r "$SC_BTS_CFG" ] || {
        mod_hint "le module « sidecar-cfg » doit avoir posé ce fichier"
        mod_fail "configuration illisible : $SC_BTS_CFG"; return $MOD_RC_FAIL; }
    mod_ok
}

mod_sidecar_bts_status() { radio_alive sidecar-bts; }

mod_sidecar_bts_start() {
    radio_dirs
    local log; log="$(radio_log sidecar-bts)"
    local essai=0 reste

```

```

while ;; do
    essai=$(( essai + 1 ))
    : > "$log"
    setsid stdbuf -oL -eL $(rt_prefixe) "$SC_BTS_BIN" -c "$SC_BTS_CFG" >>"$log" 2>&1 </d
    radio_save_pid sidecar-bts $!
    reste="$SC_BTS_STAB_SECS"
    while [ "$reste" -gt 0 ]; do
        radio_alive sidecar-bts || break
        _sc_bts_mort && break
        sleep 1; reste=$(( reste - 1 ))
    done
    if [ "$reste" -eq 0 ] && radio_alive sidecar-bts && ! _sc_bts_mort; then
        mod_say "BTS#1 stable après ${SC_BTS_STAB_SECS}s (essai $essai, $(rt_etat "$(rad
        mod_ok; return $MOD_RC_OK
    fi
    radio_kill sidecar-bts
    if [ "$essai" -ge "$SC_BTS_RETRIES" ]; then
        modb_tail "$log" 20
        mod_hint "fake_trx publie-t-il son horloge ? voir $(radio_log sidecar-faketrx)"
        mod_fail "BTS#1 instable après $essai tentatives (course « No clock »)"
        return $MOD_RC_FAIL
    fi
    mod_say "BTS#1 tombé (« No clock ») - nouvelle tentative $essai/$SC_BTS_RETRIES"
done
}

mod_sidecar_bts_wait() {
    wait_until "${MOD_TIMEOUT[sidecar-bts]}" "VTY du BTS#1 ($SC_BTS_VTY_PORT)" \
        have_port "$SC_BTS_VTY_PORT" || {
            modb_tail "$(radio_log sidecar-bts)" 20
            mod_fail "VTY $SC_BTS_VTY_PORT jamais en écoute"; return $MOD_RC_FAIL; }
    mod_ok
}

mod_sidecar_bts_stop() { radio_kill sidecar-bts; return 0; }

```

0.3.39 /opt/GSM/qosmo-overlay/run_modules/60-bts.sh

4975 octets, 114 lignes → 114 lignes

```

. "$(dirname "${BASH_SOURCE[0]}")/_lib/temps_reel.sh"

MOD_REGISTER bts "Station de base BTS#0 (osmo-bts-trx, pont)"
MOD_REQUIRED[bts]=0
MOD_DEPS[bts]="bsc"
MOD_TIMEOUT[bts]=30

: "${OSMO_BTS_TRX:=}"
: "${BTS_CFG:=${OSMOCOM_CFG:-/etc/osmocom}/osmo-bts-trx.cfg}"
: "${BTS_LOG:=${LOG_DIR}/bts.log}"
: "${BTS_VTY_PORT:=4241}"

mod_bts_check() {
    if [ -z "$OSMO_BTS_TRX" ] || [ ! -x "$OSMO_BTS_TRX" ]; then
        OSMO_BTS_TRX="$(command -v osmo-bts-trx 2>/dev/null)"
    fi
    [ -n "$OSMO_BTS_TRX" ] && [ -x "$OSMO_BTS_TRX" ] || {
        mod_hint "installez osmo-bts (cible trx) ou posez OSMO_BTS_TRX=<chemin>"
        mod_fail "osmo-bts-trx introuvable"
        return $MOD_RC_FAIL
    }
    [ -r "$BTS_CFG" ] || {
        mod_hint "attendu : <OSMOCOM_CFG>/osmo-bts-trx.cfg - ou posez BTS_CFG=<chemin>"
        mod_fail "configuration de la BTS illisible : $BTS_CFG"
        return $MOD_RC_FAIL
    }
    have_port "${BSC_VTY_PORT:-4242}" || mod_say "AVERTISSEMENT : le VTY du BSC (${BSC_VTY_PORT}) n'est pas ouvert"
    mod_say "binaire=$OSMO_BTS_TRX cfg=$BTS_CFG"
    mod_ok
}

_bts_est_le_notre() {
    tr '\0' ' ' < "/proc/$1/cmdline" 2>/dev/null | grep -qF "$BTS_CFG"
}

mod_bts_status() {
    local pid; pid="$(cat "${RUN_DIR}/bts.pid" 2>/dev/null || echo 0)"
    if [ "$pid" != 0 ] && kill -0 "$pid" 2>/dev/null && _bts_est_le_notre "$pid"; then
        return 0
    fi
    local p
    for p in $(pgrep -x osmo-bts-trx 2>/dev/null); do

```

```

        _bts_est_le_notre "$p" && return 0
done
return 1
}

mod_bts_start() {
    mkdir -p "${RUN_DIR:-/tmp/calypso}" "${(dirname "$BTS_LOG")}" 2>/dev/null || true
    : > "$BTS_LOG" 2>/dev/null || true
    # [2026-10-23] GARDIEN (BTS_GARDIEN=1 par default) : osmo-bts-trx SORT (exit 1) sur « PC
    # quand l'horloge du pont, cadencee par le DSP, cale plus de 50 trames (run du 23/10 17:
    # une liberation SDCCH, BTSO partie, plus aucune voie descendante, reseau perdu jusqu'a
    # La boucle la relance apres BTS_GARDIEN_PAUSE s ; elle attend si une autre BTS de la mer
    # deja (c54x_exe/run.sh la relance aussi en --bts). Le pid du fichier est celui de la bo
    # tue la boucle d'abord, puis la BTS.
    if [ "${BTS_GARDIEN:-1}" = 1 ]; then
        setsid bash -c '
            bin="$1" cfg="$2" log="$3" pause="$4"; shift 4
            while ;; do
                if pgrep -f -- "osmo-bts-trx -c $cfg\$" >/dev/null 2>&1; then sleep 2; contin
                stdbuf -oL -eL "$@" "$bin" -c "$cfg" </dev/null; rc=$?
                printf "\n[gardien bts] %s : osmo-bts-trx sorti (code %s), relance dans %s s"
                sleep "$pause"
            done' gardien-bts "$OSMO_BTS_TRX" "$BTS_CFG" "$BTS_LOG" "${BTS_GARDIEN_PAUSE:-3}
            >>"$BTS_LOG" 2>&1 </dev/null &
    else
        setsid stdbuf -oL -eL $(rt_prefixe) "$OSMO_BTS_TRX" -c "$BTS_CFG" >>"$BTS_LOG" 2>&1 &
    fi
    printf '%s\n' "$!" > "${RUN_DIR:-/tmp/calypso}/bts.pid"
    mod_ok
}

mod_bts_wait() {
    local pid; pid="$(cat "${RUN_DIR:-/tmp/calypso}/bts.pid" 2>/dev/null || echo 0)"
    if ! wait_until "${MOD_TIMEOUT[bts]}" "VTY de la BTS ($BTS_VTY_PORT)" have_port "$BTS_VTY
        modb_tail "$BTS_LOG" 25
        if [ "$pid" != 0 ] && ! kill -0 "$pid" 2>/dev/null; then
            mod_hint "causes fréquentes : phy/instance absente de $BTS_CFG, ou unit-id déjà p
            mod_fail "osmo-bts-trx s'est arrêté pendant son initialisation"
        else
            mod_hint "détail : $BTS_LOG"
            mod_fail "osmo-bts-trx tourne mais n'expose pas son VTY après ${MOD_TIMEOUT[bts]}
        fi
}

```

```

        return $MOD_RC_FAIL
    fi
    if [ "$pid" != 0 ] && ! kill -0 "$pid" 2>/dev/null; then
        modb_tail "$BTS_LOG" 25
        mod_hint "une BTS résiduelle (unité systemd ?) tient le VTY : systemctl stop osmo-bt
        mod_fail "le VTY $BTS_VTY_PORT répond, mais ce n'est PAS notre processus (il est mort
        return $MOD_RC_FAIL
    fi
    if grep -qi 'rsl' "$BTS_LOG" 2>/dev/null; then
        mod_say "RSL mentionné dans $BTS_LOG (lien vers le BSC en cours d'établissement)"
    else
        mod_say "aucune mention de RSL dans $BTS_LOG pour l'instant - surveillez l'OML côté
    fi
    mod_say "ordonnancement : $(rt_etat "$pid")"
    mod_ok
}

mod_bts_stop() {
    local pidf="${RUN_DIR:-/tmp/calypso}/bts.pid" pid _p
    pid="$(cat "$pidf" 2>/dev/null || echo 0)"
    [ "$pid" != 0 ] && kill "$pid" 2>/dev/null
    for _p in $(pgrep -f '^bash -c .*gardien-bts' 2>/dev/null) $(pgrep -f "gardien-bts .*$BTS
        _bts_est_le_notre "$_p" && kill "$_p" 2>/dev/null
    done
    for _p in $(pgrep -x osmo-bts-trx 2>/dev/null); do
        _bts_est_le_notre "$_p" && kill "$_p" 2>/dev/null
    done
    rm -f "$pidf"
    return 0
}

```

0.3.40 /opt/GSM/qosmo-overlay/run_modules/61-sidecar-pcu.sh

3243 octets, 62 lignes → 62 lignes

```

: "${MODDIR:=$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)}"
. "$MODDIR/_lib/core.sh"
. "$MODDIR/_lib/radio.sh"
. "$MODDIR/_lib/modem.sh"

MOD_REGISTER sidecar-pcu "Second OsmoPCU (BTS#1, GPRS du modem)"

```

```

MOD_REQUIRED[sidecar-pcu]=0
MOD_DEPS[sidecar-pcu]="sgsn sidecar-bts"
MOD_TIMEOUT[sidecar-pcu]=30
MOD_ENABLED_IF[sidecar-pcu]='[ "${NO_OSMO_START:-0}" != 1 ] && [ "${CORE_GPRS:-1}" = 1 ] &&

# Un osmo-pcu ne sert qu'UN socket BTS. Le premier (18-pcu.sh) est sur
# /tmp/pcu_bts, celui du BTS principal ; celui-ci prend /tmp/pcu_bts2, le socket
# que le BTS#1 declare (osmo-bts-trx-bts1.cfg). Sa config est derivee de celle
# du premier : meme politique, autre socket, autre journal, VTY sur 127.0.0.2.
_scpcu_gen_cfg() {
    local src; src="$(core_cfg osmo-pcu)"
    [ -r "$src" ] || return 1
    {
        sed -e "s|^log file .*|log file /var/log/osmocom/${SC_PCU_UNIT}.log|" \
            -e "/^ *pcu-socket /d" "$src"
        printf ' pcu-socket %s\n' "$SC_PCU_SOCKET"
        printf 'line vty\n no login\n bind %s\n' "$SC_PCU_VTY_HOST"
    } > "$SC_PCU_CFG.tmp" && mv -f "$SC_PCU_CFG.tmp" "$SC_PCU_CFG"
}

mod_sidecar_pcu_check() {
    core_bin osmo-pcu >/dev/null || { mod_fail "binaire osmo-pcu introuvable"; return $MOD_RC_FAIL; }
    [ -r "$(core_cfg osmo-pcu)" ] || { mod_fail "configuration du premier PCU illisible : $(core_cfg osmo-pcu)"; return $MOD_RC_FAIL; }
    mod_ok
}

mod_sidecar_pcu_status() { core_alive "$SC_PCU_UNIT"; }

mod_sidecar_pcu_start() {
    _scpcu_gen_cfg || { mod_fail "generation de $SC_PCU_CFG impossible"; return $MOD_RC_FAIL; }
    # Le bloc « pcu » doit finir par pcu-socket : sed l'a retire s'il y etait,
    # et on l'a reecrit en fin du bloc pcu (dernier noeud du fichier source).
    grep -q "^ pcu-socket $SC_PCU_SOCKET" "$SC_PCU_CFG" || {
        mod_fail "pcu-socket absent de $SC_PCU_CFG"; return $MOD_RC_FAIL; }
    mod_say "osmo-pcu -c $SC_PCU_CFG (socket $SC_PCU_SOCKET, VTY $SC_PCU_VTY_HOST:$SC_PCU_VTY_PORT)"
    core_svc_start "$SC_PCU_UNIT" "$(core_bin osmo-pcu)" -c "$SC_PCU_CFG" \
        || { mod_fail "lancement de $SC_PCU_UNIT impossible"; return $MOD_RC_FAIL; }
    [ -e "$SC_PCU_SOCKET" ] && chmod 777 "$SC_PCU_SOCKET" 2>/dev/null
    mod_ok
}

mod_sidecar_pcu_wait() {

```

```

wait_until 5 "processus $SC_PCU_UNIT" core_alive "$SC_PCU_UNIT" || {
    modb_tail "$LOG_DIR/$SC_PCU_UNIT.log" 20
    mod_fail "le second osmo-pcu s'est arrete aussitot lance"; return $MOD_RC_FAIL; }
wait_until "${MOD_TIMEOUT[sidecar-pcu]}" "liaison PCU <-> BTS#1 ($SC_PCU_SOCKET)" \
    grep -aq "PCU socket $SC_PCU_SOCKET has been connected" "/var/log/osmocom/${SC_PCU_UNIT}.log" 15
modb_tail "/var/log/osmocom/${SC_PCU_UNIT}.log" 15
mod_hint "le BTS#1 cree $SC_PCU_SOCKET (osmo-bts-trx-bts1.cfg : pcu-socket) ; bts 1 c
mod_fail "le PCU ne s'est pas raccorde au BTS#1"; return $MOD_RC_FAIL; }
mod_say "pret - Gb : nsei $SC_GPRS_NSEI, bvci $SC_GPRS_BVCI, udp $SC_GPRS_NS_PORT -> $SC
mod_ok
}

# Par pid seulement : « osmo-pcu -c » en motif tuerait aussi le premier PCU.
mod_sidecar_pcu_stop() { core_svc_stop "$SC_PCU_UNIT" "osmo-pcu -c $SC_PCU_CFG"; }

```

0.3.41 /opt/GSM/qosmo-overlay/run_modules/67-sidecar-trxcon.sh

1680 octets, 44 lignes → 44 lignes

```

. "$(dirname "${BASH_SOURCE[0]}")/_lib/radio.sh"

MOD_REGISTER sidecar-trxcon "Couche 1 du MS#2 (trxcon)"
MOD_REQUIRED[sidecar-trxcon]=1
MOD_DEPS[sidecar-trxcon]="sidecar-faketrx"
MOD_TIMEOUT[sidecar-trxcon]=20

: "${SC_TRXCON_BIN:=trxcon}"
: "${SC_TRXCON_FN_ADVANCE:=100}"

mod_sidecar_trxcon_check() {
    command -v "$SC_TRXCON_BIN" >/dev/null 2>&1 || {
        mod_fail "binaire introuvable : $SC_TRXCON_BIN"; return $MOD_RC_FAIL; }
    mod_ok
}

mod_sidecar_trxcon_status() { radio_alive sidecar-trxcon && have_unix "$SC_L2_SOCKET"; }

mod_sidecar_trxcon_start() {
    radio_dirs
    local log; log="$(radio_log sidecar-trxcon)"
    rm -f "$SC_L2_SOCKET" 2>/dev/null
}

```

```

    mod_say "trxcon -p $SC_BB_PORT -s $SC_L2_SOCKET (avance de trame $SC_TRXCON_FN_ADVANCE)"
    mod_say "journal : $log"
    setsid stdbuf -oL -eL "$SC_TRXCON_BIN" -i "$TRX_BIND_IP" -b "$TRX_BB_IP" \
        -p "$SC_BB_PORT" -s "$SC_L2_SOCKET" -C 1 -F "$SC_TRXCON_FN_ADVANCE" >>"$log" 2>&1 </dev/null &
    radio_save_pid sidecar-trxcon $!
    mod_ok
}

mod_sidecar_trxcon_wait() {
    local log; log="$(radio_log sidecar-trxcon)"
    wait_until 5 "processus trxcon" radio_alive sidecar-trxcon || {
        modb_tail "$log" 20
        mod_fail "trxcon s'est arrêté aussitôt lancé"; return $MOD_RC_FAIL; }
    wait_until "${MOD_TIMEOUT[sidecar-trxcon]}" "socket L1CTL $SC_L2_SOCKET" \
        have_unix "$SC_L2_SOCKET" || {
        modb_tail "$log" 20
        mod_hint "fake_trx répond-il sur $SC_BB_PORT ? voir $(radio_log sidecar-faketrx)"
        mod_fail "socket L1CTL jamais créée"; return $MOD_RC_FAIL; }
    mod_ok
}

mod_sidecar_trxcon_stop() { radio_kill sidecar-trxcon; rm -f "$SC_L2_SOCKET"; return 0; }

```

0.3.42 /opt/GSM/qosmo-overlay/run_modules/68-sidecar-mobile.sh

2124 octets, 53 lignes → 53 lignes

```

. "$(dirname "${BASH_SOURCE[0]}")/_lib/radio.sh"

MOD_REGISTER sidecar-mobile "Second abonné (MS#2)"
MOD_REQUIRED[sidecar-mobile]=0
MOD_DEPS[sidecar-mobile]="sidecar-trxcon"
MOD_TIMEOUT[sidecar-mobile]=30

: "${SC_MOBILE_BIN:=mobile}"
: "${SC_MOBILE_DELAY:=3}"
: "${CALYPSO_PULSE_LATENCY_MSEC:=320}"

mod_sidecar_mobile_check() {
    command -v "$SC_MOBILE_BIN" >/dev/null 2>&1 || {
        mod_fail "binaire introuvable : $SC_MOBILE_BIN"; return $MOD_RC_FAIL; }
}

```

```

[ -r "$SC_MOBILE_CFG" ] || {
    mod_hint "le module « sidecar-cfg » doit avoir posé ce fichier"
    mod_fail "configuration illisible : $SC_MOBILE_CFG"; return $MOD_RC_FAIL; }
mod_ok
}

mod_sidecar_mobile_status() { radio_alive sidecar-mobile; }

mod_sidecar_mobile_start() {
    radio_dirs
    local log; log="$(radio_log sidecar-mobile)"
    wait_until 30 "socket L1CTL $SC_L2_SOCK" have_unix "$SC_L2_SOCK" || {
        mod_fail "socket L1CTL absente : $SC_L2_SOCK"; return $MOD_RC_FAIL; }
    if [ "${SC_MOBILE_DELAY}" != 0 ]; then
        modSay "attente de ${SC_MOBILE_DELAY}s avant lancement"
        sleep "$SC_MOBILE_DELAY"
    fi
    modSay "mobile -c $SC_MOBILE_CFG (VTY $SC_MOBILE_VTY_PORT)"
    modSay "journal : $log"
    modSay "latence : PULSE_LATENCY_MSEC=${CALYPSO_PULSE_LATENCY_MSEC:-<defaut greffon>}"
    modSay "debug      : $CALYPSO_MOBILE_DEBUG"
    PULSE_LATENCY_MSEC="$CALYPSO_PULSE_LATENCY_MSEC" \
        setsid stdbuf -oL -eL "$SC_MOBILE_BIN" -c "$SC_MOBILE_CFG" \
        -d "$CALYPSO_MOBILE_DEBUG" >>"$log" 2>&1 </dev/null &
    radio_save_pid sidecar-mobile $!
    mod_ok
}

mod_sidecar_mobile_wait() {
    wait_until "${MOD_TIMEOUT[sidecar-mobile]}" "VTY du MS#2 ($SC_MOBILE_VTY_PORT)" \
        have_port "$SC_MOBILE_VTY_PORT" || {
        modb_tail "$(radio_log sidecar-mobile)" 20
        modHint "port déjà pris ? ss -tlnp | grep :$SC_MOBILE_VTY_PORT"
        mod_fail "VTY $SC_MOBILE_VTY_PORT jamais en écoute"; return $MOD_RC_FAIL; }
    modSay "prêt - telnet 127.0.0.1 $SC_MOBILE_VTY_PORT puis « show ms »"
    mod_ok
}

mod_sidecar_mobile_stop() { radio_kill sidecar-mobile; return 0; }

```

0.3.43 /opt/GSM/qosmo-overlay/run_modules/69-modem-trxcon.sh

2606 octets, 56 lignes → 56 lignes

```
. "$(dirname "${BASH_SOURCE[0]}")/_lib/radio.sh"
. "$(dirname "${BASH_SOURCE[0]}")/_lib/modem.sh"

MOD_REGISTER modem-trxcon "Couche 1 du MS GPRS (trxcon, second MS du fake_trx)"
MOD_REQUIRED[modem-trxcon]=0
MOD_DEPS[modem-trxcon]="sidecar-faketrx"
MOD_TIMEOUT[modem-trxcon]=20
MOD_ENABLED_IF[modem-trxcon]='[ "${MODEM_GPRS:-1}" = 1 ]'

: "${SC_TRXCON_BIN:=trxcon}"

mod_modem_trxcon_check() {
    command -v "$SC_TRXCON_BIN" >/dev/null 2>&1 || {
        mod_fail "binaire introuvable : $SC_TRXCON_BIN"; return $MOD_RC_FAIL; }
    mod_ok
}

mod_modem_trxcon_status() { radio_alive modem-trxcon && have_unix "$MODEM_L2_SOCKET"; }

# Meme lancement que le trxcon du MS#2 (67-sidecar-trxcon.sh), sur la base
# TRXD que fake_trx a ouverte pour ce second MS (--trx ...:MODEM_BB_PORT).
mod_modem_trxcon_start() {
    radio_dirs
    local log; log="$(radio_log modem-trxcon)"
    rm -f "$MODEM_L2_SOCKET" 2>/dev/null
    # -f 0 (avance de tirage des rafales montantes) : sur un PDCH, trxcon
    # demande au modem le bloc montant X au DECODAGE du bloc descendant
    # (rafale X-1, trxcon_fsm.c TRXCON_EV_RX_DATA_IND) mais, avec l'avance par
    # default (2), il TIRE ce bloc a l'arrivee de la rafale X-2 : la primitive
    # arrive toujours apres le tirage et part a la poubelle (« dropping stale
    # Tx prim », un bloc de retard a chaque fois ; N3101 cote PCU). A 0, le
    # tirage a lieu a la rafale X : une trame pour repondre. MS#2 (voix) garde
    # son propre trxcon et son avance.
    mod_say "trxcon -p $MODEM_BB_PORT -s $MODEM_L2_SOCKET (fbsb +$MODEM_TRXCON_FN_ADVANCE, avan
    mod_say "journal : $log"
    setsid stdbuf -oL -eL "$SC_TRXCON_BIN" -i "$TRX_BIND_IP" -b "$TRX_BB_IP" \
        -p "$MODEM_BB_PORT" -s "$MODEM_L2_SOCKET" -C 1 -F "$MODEM_TRXCON_FN_ADVANCE" \
        -f "$MODEM_TRXCON_TRX_ADVANCE" >>"$log" 2>&1 </dev/null &
    radio_save_pid modem-trxcon $!
```

```

    mod_ok
}

mod_modem_trxcon_wait() {
    local log; log="$(radio_log modem-trxcon)"
    wait_until 5 "processus trxcon (modem)" radio_alive modem-trxcon || {
        modb_tail "$log" 20
        mod_fail "trxcon (modem) s'est arrete aussitot lance"; return $MOD_RC_FAIL; }
    wait_until "${MOD_TIMEOUT[modem-trxcon]}" "socket L1CTL $MODEM_L2_SOCKET" \
        have_unix "$MODEM_L2_SOCKET" || {
        modb_tail "$log" 20
        mod_hint "fake_trx a-t-il ete lance avec --trx ...:$MODEM_BB_PORT ? voir $(radio_log
        mod_fail "socket L1CTL jamais creee"; return $MOD_RC_FAIL; }
    mod_ok
}

mod_modem_trxcon_stop() { radio_kill modem-trxcon; rm -f "$MODEM_L2_SOCKET"; return 0; }

```

0.3.44 /opt/GSM/qosmo-overlay/run_modules/70-l2.sh

4027 octets, 97 lignes → 97 lignes

```

MOD_REGISTER 12 "Client de couche 2 (mobile, MS#1)"
MOD_REQUIRED[12]=0
MOD_DEPS[12]="osmocon"
MOD_TIMEOUT[12]=30

: "${L1CTL_SOCKET_PATH:=/tmp/osmocom_12}"
: "${CALYPSO_MOBILE_DEBUG:=DCS:DNB:DPLMN:DRR:DMM:DSIM:DCC:DMNCC:DSS:DLSMS:DPAG:DSUM:DSAP:DGP}
: "${CALYPSO_PULSE_LATENCY_MSEC:=320}"
: "${CALYPSO_L1_READY_MARKER:=your code is running now}"
: "${CALYPSO_L1_READY_TIMEOUT:=20}"

_l2_log() { printf '%s' "${LOG_DIR}/mobile.log"; }
_l2_cfg() {
    if [ -n "${MOBILE_CFG:-}" ]; then printf '%s' "$MOBILE_CFG"
    elif [ -r "${OSMOCOM_HOME:-$HOME/.osmocom}/bb/mobile_group1.cfg" ]; then
        printf '%s' "${OSMOCOM_HOME:-$HOME/.osmocom}/bb/mobile_group1.cfg"
    else printf '%s' "${QEMU_CFGS:-${QEMU_TREE}/cfgs}/mobile_group1.cfg"; fi
}
_l2_pat() { printf '%s' "mobile -c $_l2_cfg"; }

```

```

mod_l2_check() {
    command -v mobile >/dev/null 2>&1 || {
        mod_hint "compilez osmocom-bb (layer23) ou ajoutez son répertoire au PATH"
        mod_fail "client L2 introuvable dans le PATH : mobile"
        return $MOD_RC_FAIL; }
    local cfg; cfg="$(_l2_cfg)"
    [ -r "$cfg" ] || {
        mod_hint "le module mobile-cfg doit avoir copié cfgs/mobile_group1.cfg vers \${OSMOCON}"
        mod_fail "configuration du mobile illisible : $cfg"
        return $MOD_RC_FAIL; }
    if [ "${DRY:-0}" = 1 ]; then
        mod_say "simulation : attente de la socket L1CTL et du marqueur L1 non jouées"
        mod_ok; return $MOD_RC_OK
    fi
    wait_until "${MOD_TIMEOUT[l2]}" "socket L1CTL $L1CTL SOCK_PATH" \
        have_unix "$L1CTL SOCK_PATH" || {
        mod_hint "c'est osmocon qui crée cette socket (option -s) : regardez le journal du m"
        mod_fail "socket L1CTL absente : $L1CTL SOCK_PATH"
        return $MOD_RC_FAIL; }
    if ! wait_until "$CALYPSO_L1_READY_TIMEOUT" "L1 prête" \
        log_has "${LOG_DIR}/osmocon.log" "$CALYPSO_L1_READY_MARKER"; then
        mod_say "AVERTISSEMENT : marqueur « $CALYPSO_L1_READY_MARKER » absent d'osmocon.log"
    fi
    mod_ok
}

mod_l2_status() {
    local pid; pid="$(cat "${RUN_DIR}/l2.pid" 2>/dev/null || echo 0)"
    if [ "$pid" != 0 ] && kill -0 "$pid" 2>/dev/null; then
        tr '\0' ' ' < "/proc/$pid/cmdline" 2>/dev/null | grep -qF "$(_l2_cfg)" && return 0
    fi
    have_proc "$(_l2_pat)"
}

mod_l2_start() {
    local log cfg
    log="$(_l2_log)"; cfg="$(_l2_cfg)"
    mkdir -p "$LOG_DIR" "$RUN_DIR" 2>/dev/null
    : > "$log"
    mod_say "journal    : $log"
    mod_say "cfg        : $cfg"
    mod_say "debug       : $CALYPSO_MOBILE_DEBUG"
}

```

```

mod_say "latence : PULSE_LATENCY_MSEC=$CALYPSO_PULSE_LATENCY_MSEC"
PULSE_LATENCY_MSEC="$CALYPSO_PULSE_LATENCY_MSEC" \
    setsid mobile -c "$cfg" -d "$CALYPSO_MOBILE_DEBUG" >>"$log" 2>&1 </dev/null &
printf '%s\n' "$!" > "${RUN_DIR}/l2.pid"
mod_ok
}

mod_l2_wait() {
    local pid log; pid="$(cat "${RUN_DIR}/l2.pid" 2>/dev/null || echo 0)"
    log="$_l2_log"
    wait_until "${MOD_TIMEOUT[l2]}" "première trace du mobile" test -s "$log" || {
        mod_hint "lisez $log ; « same layer2-socket » = un autre client tient déjà $L1CTL_SOCKET"
        mod_fail "mobile n'a produit aucune trace"
        return $MOD_RC_FAIL; }
    kill -0 "$pid" 2>/dev/null || {
        mod_hint "lisez la fin de $log : la cause est presque toujours dans les 10 dernières lignes"
        mod_fail "mobile a démarré puis s'est arrêté"
        return $MOD_RC_FAIL; }
    if command -v ss >/dev/null 2>&1; then
        if ss -x 2>/dev/null | grep -q "$L1CTL_SOCKET_PATH"; then
            mod_say "socket L1CTL $L1CTL_SOCKET_PATH : connexion observée"
        else
            mod_say "socket L1CTL $L1CTL_SOCKET_PATH : pas de connexion visible via ss (indice 1)"
        fi
    fi
    mod_ok
}

mod_l2_stop() {
    local pid; pid="$(cat "${RUN_DIR}/l2.pid" 2>/dev/null || echo 0)"
    [ "$pid" != 0 ] && kill "$pid" 2>/dev/null
    pkill -f "$_l2_pat" 2>/dev/null
    rm -f "${RUN_DIR}/l2.pid"
    return 0
}

```

0.3.45 /opt/GSM/qosmo-overlay/run_modules/71-modem-gprs.sh

5888 octets, 135 lignes → 135 lignes

```

: "${MODDIR:=$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)}"
. "$MODDIR/_lib/core.sh"
. "$MODDIR/_lib/radio.sh"
. "$MODDIR/_lib/modem.sh"

MOD_REGISTER modem-gprs "MS GPRS (modem osmocom-bb : attache, PDP, tun)"
MOD_REQUIRED[modem-gprs]=0
MOD_DEPS[modem-gprs]="modem-trxcon sidecar-pcu abones-hlr"
MOD_TIMEOUT[modem-gprs]=90
MOD_ENABLED_IF[modem-gprs]='[ "${MODEM_GPRS:-1}" = 1 ]'

: "${HLR_VTY_PORT:=4258}"
: "${OPERATOR_ID:=1}"
: "${BSC_CFG:=${OSMOCOM_CFG:-/etc/osmocom}/osmo-bsc.cfg}"

# Memes formules que 21-abones-hlr.sh, pour le 3e abonne (MODEM_MS_IDX).
_mg_mcc() { printf '%s\n' "${MCC:-$(core_cfg_field "$(core_cfg osmo-msc)" '^[:space:]]*netw"}"
_mg_mnc() { printf '%s\n' "${MNC:-$(core_cfg_field "$(core_cfg osmo-msc)" '^[:space:]]*mobi"}"
_mg_imsi() { printf '%s%s%04d%06d\n' "$(_mg_mcc)" "$(_mg_mnc)" "$OPERATOR_ID" "$MODEM_MS_IDX"
_mg_ki() { printf '00112233445566778899aabbccdd%02x%02x\n' "$MODEM_MS_IDX" "$OPERATOR_ID"; }
_mg_ki_sp() { _mg_ki | sed 's/./& /g; s/ $//'; }
_mg_node() {
    local n="${OSMO_WAN_NODE:-${WAN_NODE_ID:-}}"
    [ -n "$n" ] || n="$(sed -n 's/^PLAN_NODE=//p' "${OSMOCOM_CFG:-/etc/osmocom}/radio-plan.en
    case "$n" in [1-9]) ;; *) n=1 ;; esac
    printf '%s' "$n"
}
_mg_msisdn() { echo $(( $(_mg_node) * 100000 + OPERATOR_ID * 100 + MODEM_MS_IDX )); }
# L'ARFCN du BTS#1 tel que le BSC le declare (le radio-plan peut l'avoir change).
_mg_arfcn() {
    local a; a="$(awk '/^ bts 1/{f=1} f && /^ *arfcn/{print $2; exit}' "$BSC_CFG" 2>/dev/nu
    printf '%s' "${a:-$SC_ARFCN}"
}

_mg_write_cfg() {
    cat > "$MODEM_CFG.tmp" <<EOF
! modem-gprs.cfg - MS GPRS du banc (osmocom-bb modem), ecrit par 71-modem-gprs.sh
! IMSI $(_mg_imsi) · MSISDN $(_mg_msisdn) · APN $MODEM_APN -> $MODEM_TUN · ARFCN $(_mg_arfcn)
line vty
no login
bind 127.0.0.1 $MODEM_VTY_PORT
log stderr

```

```

logging filter all 1
logging color 0
logging timestamp 1
logging print category 1
logging print level 1
logging level all notice
ms 1
layer2-socket $MODEM_L2_SOCKET
sim test
imei 35892500865948$MODEM_MS_IDX 0
imei-fixed
test-sim
    imsi $_mg_imsi
    ki comp128 $_mg_ki_sp
    no barred-access
    rplmn $_mg_mcc $_mg_mnc
apn $MODEM_APN
tun-device $MODEM_TUN
type-support v4
no shutdown
no shutdown
EOF
    mv -f "$MODEM_CFG.tmp" "$MODEM_CFG"
}

mod_modem_gprs_check() {
    [ -x "$MODEM_BIN" ] || {
        mod_hint "compilez layer23 d'osmocom-bb (src/host/layer23/src/modem/modem), ou posez
        mod_fail "binaire modem introuvable : $MODEM_BIN"; return $MOD_RC_FAIL; }
    mod_ok
}

mod_modem_gprs_status() { radio_alive modem-gprs; }

mod_modem_gprs_start() {
    radio_dirs
    local log imsi; log="$(radio_log modem-gprs)"; imsi="$_mg_imsi"

    # 1. L'abonne dans le HLR (idempotent : « create » d'un IMSI existant est refuse sans ef
    core_vty_ask "$HLR_VTY_PORT" "enable" \
        "subscriber imsi $imsi create" \
        "subscriber imsi $imsi update msisdn $_mg_msisdn" \

```

```

    "subscriber imsi $imsi update aud2g comp128v1 ki ${_mg_ki}" "exit" >/dev/null 2>&1
# Presence : la base d'abord (comme 21-abonnes-hlr.sh), le VTY en repli -
# le dialogue VTY peut ne rien rendre (socat absent, delais) alors que
# l'abonne est bien la.
_mg_present() {
    local n=""
    if command -v sqlite3 >/dev/null 2>&1; then
        n="$(sqlite3 -readonly "${HLR_DB:-/var/lib/osmocom/hlr.db}" "select count(*) from
        [ "${n:-0}" -ge 1 ] 2>/dev/null && return 0
    fi
    core_vty_ask "$HLR_VTY_PORT" "enable" "subscriber imsi $1 show" "exit" 2>/dev/null |
}
_mg_present "$imsi" \
    || { mod_fail "abonne $imsi absent du HLR apres provisionnement (VTY $HLR_VTY_PORT)"
mod_say "abonne GPRS : IMSI $imsi, MSISDN ${_mg_msisdn}"

# 2. La config du modem, reecrite a chaque lancement (elle suit le plan radio).
_mg_write_cfg || { mod_fail "ecriture de $MODEM_CFG impossible"; return $MOD_RC_FAIL; }

# 3. Le modem. Il cree lui-meme $MODEM_TUN (root) une fois le PDP actif.
wait_until 30 "socket L1CTL $MODEM_L2_SOCK" have_unix "$MODEM_L2_SOCK" || {
    mod_fail "socket L1CTL absente : $MODEM_L2_SOCK"; return $MOD_RC_FAIL; }
ip link del "$MODEM_TUN" 2>/dev/null
mod_say "modem -c $MODEM_CFG -a ${_mg_arfcn} (VTY $MODEM_VTY_PORT, tun $MODEM_TUN)"
mod_say "journal : $log"
setuid stdbuf -oL -eL "$MODEM_BIN" -c "$MODEM_CFG" -s "$MODEM_L2_SOCK" -a "${_mg_arfcn}"
radio_save_pid modem-gprs $!
mod_ok
}

mod_modem_gprs_wait() {
    local log; log="$(radio_log modem-gprs)"
    wait_until 5 "processus modem" radio_alive modem-gprs || {
        modb_tail "$log" 20
        mod_fail "le modem s'est arrete aussitot lance"; return $MOD_RC_FAIL; }
    wait_until 20 "VTY du modem ($MODEM_VTY_PORT)" have_port "$MODEM_VTY_PORT" || {
        modb_tail "$log" 20
        mod_fail "VTY $MODEM_VTY_PORT jamais en ecoute"; return $MOD_RC_FAIL; }
# L'attache GPRS puis le PDP : visibles quand le tun porte une adresse (celle du GGSN, 1
if wait_until "${MOD_TIMEOUT[modem-gprs]}" "PDP actif ($MODEM_TUN avec adresse)" \
    sh -c "ip -4 -o addr show '$MODEM_TUN' 2>/dev/null | grep -q inet"; then
    mod_say "pret - $(ip -4 -o addr show "$MODEM_TUN" | awk '{print $4}' | head -1) sur

```

```

else
    modb_tail "$log" 25
    mod_hint "attache GMM / PDP non abouti : journal du modem, puis osmo-sgsn.log (attache)"
    mod_say "modem lance mais pas de PDP dans le delai - la voix n'est pas concernee"
fi
mod_ok
}

mod_modem_gprs_stop() { radio_kill modem-gprs; ip link del "$MODEM_TUN" 2>/dev/null; return 0; }

```

0.3.46 /opt/GSM/qosmo-overlay/run_modules/75-gsmtap.sh

2319 octets, 63 lignes → 63 lignes

```

MOD_REGISTER gsmtap "Capture GSMTAP (pcap)"
MOD_REQUIRED[gsmtap]=0
MOD_DEPS[gsmtap]="l2"
MOD_TIMEOUT[gsmtap]=15

: "${PORT_GSMTAP:=4729}"
: "${CALYPSO_GSMTAP_IFACE:=any}"
: "${CAPTURE_DIR:=${RUN_DIR:-/tmp/calypso}/captures}"

_gsmtap_pat() { printf '%s' "tcpdump .*udp port ${PORT_GSMTAP}"; }
_gsmtap_file() { cat "${RUN_DIR}/gsmtap.pcap.path" 2>/dev/null; }

mod_gsmtap_check() {
    command -v tcpdump >/dev/null 2>&1 || {
        mod_hint "installez-le vous-même : apt-get install tcpdump - ou lancez avec --skip g"
        mod_fail "tcpdump absent : aucune capture GSMTAP possible"
        return $MOD_RC_FAIL; }
    local d="$CAPTURE_DIR"
    [ -d "$d" ] || d="$(dirname "$CAPTURE_DIR")"
    [ -w "$d" ] || {
        mod_hint "posez CAPTURE_DIR vers un répertoire inscriptible"
        mod_fail "répertoire de capture non inscriptible : $d"
        return $MOD_RC_FAIL; }
    mod_ok
}

mod_gsmtap_status() { have_proc "$(_gsmtap_pat)"; }

```

```

mod_gsmtap_start() {
    local pcap log
    pcap="$CAPTURE_DIR/mobile-gsmtap-$(date +%Y%m%d_%H%M%S).pcap"
    log="${LOG_DIR}/gsmtap.log"
    mkdir -p "$LOG_DIR" "$RUN_DIR" "$CAPTURE_DIR" 2>/dev/null
    printf '%s\n' "$pcap" > "${RUN_DIR}/gsmtap.pcap.path"
    mod_say "capture : $pcap"
    mod_say "filtre : udp port $PORT_GSM TAP sur $CALYPSO_GSM TAP_IFACE"
    setsid tcpdump -i "$CALYPSO_GSM TAP_IFACE" -U \
        -w "$pcap" "udp port $PORT_GSM TAP" >>"$log" 2>&1 </dev/null &
    printf '%s\n' "$!" > "${RUN_DIR}/gsmtap.pid"
    mod_ok
}

mod_gsmtap_wait() {
    local pid pcap; pid="$(cat "${RUN_DIR}/gsmtap.pid" 2>/dev/null || echo 0)"
    pcap="$(_gsmtap_file)"
    wait_until "${MOD_TIMEOUT[gsmtap]}" "création du pcap" test -s "$pcap" || {
        mod_hint "lisez ${LOG_DIR}/gsmtap.log ; « permission denied » = capture sans droits"
        mod_fail "tcpdump n'a pas créé $pcap"
        return $MOD_RC_FAIL; }
    kill -0 "$pid" 2>/dev/null || {
        mod_fail "tcpdump a démarré puis s'est arrêté"
        mod_hint "lisez ${LOG_DIR}/gsmtap.log"
        return $MOD_RC_FAIL; }
    mod_ok
}

mod_gsmtap_stop() {
    local pid; pid="$(cat "${RUN_DIR}/gsmtap.pid" 2>/dev/null || echo 0)"
    [ "$pid" != 0 ] && kill "$pid" 2>/dev/null
    pkill -f "$(_gsmtap_pat)" 2>/dev/null
    rm -f "${RUN_DIR}/gsmtap.pid"
    return 0
}

```

0.3.47 /opt/GSM/qosmo-overlay/run_modules/80-panes.sh

2649 octets, 67 lignes → 67 lignes

```

MOD_REGISTER panes "Vue de supervision (tmux)"
MOD_REQUIRED[panes]=0
MOD_DEPS[panes]="tmux"
MOD_TIMEOUT[panes]=15

: "${TMUX_SESSION:=calypso}"
: "${CALYPSO_GSM_SNIFF:=${QEMU_TREE}/opt-gsm-scripts/gsm_sniff.py}"

_panes_count() { tmux list-panes -t "$TMUX_SESSION:all" 2>/dev/null | wc -l; }
_panes_enough() { [ "$(_panes_count)" -ge 2 ]; }

mod_panes_check() {
    command -v tmux >/dev/null 2>&1 || { mod_skip "tmux absent - pas de vue de supervision";
    if [ "${DRY:-0}" = 1 ]; then
        mod_say "simulation : existence de la session tmux non vérifiée"
        mod_ok; return $MOD_RC_OK
    fi
    tmux has-session -t "$TMUX_SESSION" 2>/dev/null || {
        mod_hint "le module tmux crée la session ; ./run.sh --only tmux"
        mod_fail "session tmux « $TMUX_SESSION » inexistante"
        return $MOD_RC_FAIL; }
    mod_ok
}

mod_panes_status() { tmux list-windows -t "$TMUX_SESSION" 2>/dev/null | grep -q '^[0-9]*: al

mod_panes_start() {
    tmux new-window -t "$TMUX_SESSION" -n all \
        "clear; echo '=== osmocon ==='; tail -F ${LOG_DIR}/osmocon.log" || {
        mod_fail "tmux a refusé de créer la fenêtre « all »"
        return $MOD_RC_FAIL; }
    local specs=() spec name what cmd added=0
    specs+=("qemu|${LOG_DIR}/qemu.log")
    [ -r "$CALYPSO_GSM_SNIFF" ] && specs+=("burst|__BURST__")
    specs+=("${MS1_LOG_NOM:-mobile}|${MS1_LOG:-${LOG_DIR}/mobile.log}") # [2026-10-24] rvin
    specs+=("pont|/dev/shm/pont.log")
    for spec in "${specs[@]}"; do
        name="${spec%|*}"; what="${spec##*|}"
        case "$what" in
            __BURST__)
                cmd="clear; echo '=== BURST decode (4731 + 5700-5702) ==='; python3 -u $CALY
            *) cmd="clear; echo '=== $name ==='; tail -F $what" ;;

```

```

    esac
    if tmux split-window -t "$TMUX_SESSION:all" "$cmd" 2>/dev/null; then
        added=$((added + 1))
    else
        mod_say "pane « $name » refusé par tmux (place insuffisante)"
    fi
    tmux select-layout -t "$TMUX_SESSION:all" tiled >/dev/null 2>&1
done
mod_say "fenêtre « all » : $((added + 1)) panes demandés"
mod_ok
}

mod_panes_wait() {
    wait_until "${MOD_TIMEOUT[panes]}" "panes de la fenêtre « all »" _panes_enough || {
        mod_hint "tmux list-panes -t $TMUX_SESSION:all pour voir ce qui a survécu"
        mod_fail "la fenêtre « all » n'a pas gardé ses panes"
        return $MOD_RC_FAIL; }
    mod_say "panes vivants : $_panes_count)"
    mod_ok
}

mod_panes_stop() {
    tmux kill-window -t "$TMUX_SESSION:all" 2>/dev/null
    return 0
}

```

0.3.48 /opt/GSM/qosmo-overlay/run_modules/95-summary.sh

1284 octets, 25 lignes → 25 lignes

```

MOD_REGISTER summary "Identité du build et résumé"
MOD_REQUIRED[summary]=0
MOD_PURE[summary]=1
MOD_TIMEOUT[summary]=10

mod_summary_check() { mod_ok; }
mod_summary_status() { return $MOD_RC_FAIL; }

mod_summary_start() {
    mod_say "==== IDENTITÉ DU BUILD ====="
    if command -v git >/dev/null 2>&1 && [ -d "${QEMU_TREE}/.git" ]; then

```

```

    mod_say "  révision git      : $(git -C "$QEMU_TREE" rev-parse --short HEAD 2>/dev/null)
    mod_say "  arbre modifié    : $(git -C "$QEMU_TREE" diff --stat 2>/dev/null | tail -1)
fi
mod_say "  binaire QEMU       : ${QEMU_BIN:-<non défini>}"
mod_say "  date du binaire    : $(stat -c %y "${QEMU_BIN:-/nonexistent}" 2>/dev/null | cut -d. -f1)
mod_say "  firmware         : ${FIRMWARE_ELF:-<non défini>}"
mod_say "==== RÉSUMÉ D'ENVIRONNEMENT ====="
mod_say "  couche 1                : gr-gsm (GSMTAP ${PORT_GSMTAP_SI:-4730}, SCH ${PORT_GSMTAP_SCH:-})"
mod_say "  cellule du pont         : ARFCN ${PONT_ARFCN:-?} BSIC ${PONT_BSIC:-?}"
mod_say "  chiffrement             : ${ENCRYPTION:-?}"
mod_say "  MS#1 / MS#2             : ${MOBILE_CFG:-?} / ${SC_MOBILE_CFG:-?}"
mod_say "  LOG_DIR / RUN_DIR       : ${LOG_DIR:-} / ${RUN_DIR:-}"
mod_ok
}

```

0.3.49 /opt/GSM/qosmo-overlay/run_modules/99-attach.sh

1931 octets, 50 lignes → 50 lignes

```

MOD_REGISTER attach "Reprise en main de la session tmux"
MOD_REQUIRED[attach]=0
MOD_DEPS[attach]="tmux"
MOD_TIMEOUT[attach]=10
MOD_ENABLED_IF[attach]='[ "${CALYPSO_NO_ATTACH:-0}" != 1 ] && [ -t 1 ]'

: "${TMUX_SESSION:=calypso}"
: "${CALYPSO_DEFAULT_WINDOW:=all}"

_attach_current_window() {
    tmux display-message -p -t "$TMUX_SESSION" '#W' 2>/dev/null
}

mod_attach_check() {
    command -v tmux >/dev/null 2>&1 || { mod_skip "tmux absent - rien à sélectionner"; return 1; }
    tmux has-session -t "$TMUX_SESSION" 2>/dev/null || {
        mod_hint "le module tmux (35) crée la session ; sans elle la pile tourne quand même,"
        mod_skip "session tmux « $TMUX_SESSION » inexistante"
        return $MOD_RC_SKIP; }
    mod_ok
}

```

```

mod_attach_status() { [ "$(_attach_current_window)" = "$CALYPSO_DEFAULT_WINDOW" ]; }

mod_attach_start() {
    mkdir -p "$RUN_DIR" 2>/dev/null
    tmux select-window -t "$TMUX_SESSION:$CALYPSO_DEFAULT_WINDOW" 2>/dev/null \
        || tmux select-window -t "$TMUX_SESSION:qemu" 2>/dev/null \
        || mod_say "aucune des fenêtres « $CALYPSO_DEFAULT_WINDOW » / « qemu » n'existe"

    printf 'tmux attach -t %s\n' "$TMUX_SESSION" > "${RUN_DIR}/attach.cmd"
    mod_say "reprise en main : tmux attach -t $TMUX_SESSION"
    mod_say "fin de session : tmux kill-session -t $TMUX_SESSION"
    mod_ok
}

mod_attach_wait() {
    wait_until "${MOD_TIMEOUT[attach]}" "session tmux « $TMUX_SESSION »" \
        tmux has-session -t "$TMUX_SESSION" || {
        mod_fail "la session tmux a disparu"
        return $MOD_RC_FAIL; }
    local w; w="$(_attach_current_window)"
    [ -n "$w" ] || { mod_fail "aucune fenêtre sélectionnable dans « $TMUX_SESSION »"
        mod_hint "tmux list-windows -t $TMUX_SESSION"
        return $MOD_RC_FAIL; }
    mod_say "fenêtre courante : $w"
    mod_ok
}

mod_attach_stop() { return 0; }

```

0.3.50 /opt/GSM/qosmo-overlay/run_modules/_lib/core.sh

4542 octets, 131 lignes → 131 lignes

```

[ -n "${CORE_LIB_LOADED:-}" ] && return 0
CORE_LIB_LOADED=1

: "${OSMOCOM_CFG:=/etc/osmocom}"
: "${RUN_DIR:=/tmp/calypso}"
: "${LOG_DIR:=$RUN_DIR/logs}"
: "${CORE_VTY_HOST:=127.0.0.1}"

```

```

core_bin() {
    local n="$1" p
    for p in "/usr/bin/$n" "/usr/local/bin/$n"; do
        [ -x "$p" ] && { printf '%s\n' "$p"; return 0; }
    done
    p="$(command -v "$n" 2>/dev/null)" || return 1
    [ -n "$p" ] && { printf '%s\n' "$p"; return 0; }
    return 1
}

core_cfg() { printf '%s/%s.cfg\n' "$OSMOCOM_CFG" "$1"; }

core_cfg_field() {
    local f="$1" re="$2" n="$3" def="${4:-}" v=""
    if [ -r "$f" ]; then
        v="$(grep -aE "$re" "$f" 2>/dev/null | head -n1 | awk -v k="$n" '{print $k}')"
    fi
    printf '%s\n' "${v:-$def}"
}

core_has_systemd() { [ -d /run/systemd/system ] && command -v systemctl >/dev/null 2>&1; }
core_unit_exists() { core_has_systemd && systemctl cat "$1" >/dev/null 2>&1; }
core_unit_active() { core_has_systemd && systemctl is-active --quiet "$1"; }
core_nrestarts() { core_has_systemd && systemctl show -p NRestarts --value "$1" 2>/dev/null; }
core_pidfile() { printf '%s/%s.pid\n' "$RUN_DIR" "$1"; }

core_svc_start() {
    local unit="$1" bin="$2"; shift 2
    mkdir -p "$RUN_DIR" "$LOG_DIR" 2>/dev/null || true
    core_nrestarts "$unit" > "$RUN_DIR/$unit.nrestarts" 2>/dev/null || true
    if core_unit_exists "$unit"; then
        mod_say "systemctl start $unit"
        systemctl start "$unit" || return 1
        return 0
    fi
    mod_say "systemd indisponible - lancement direct : $bin $*"
    setsid "$bin" "$@" >>"$LOG_DIR/$unit.log" 2>&1 </dev/null &
    printf '%s\n' "$!" > "$(core_pidfile "$unit")"
    return 0
}

core_svc_stop() {

```

```

local unit="$1" pat="${2:-}" pf
core_unit_exists "$unit" && systemctl stop "$unit" 2>/dev/null
pf="$(core_pidfile "$unit")"
if [ -f "$pf" ]; then
    kill "$(cat "$pf" 2>/dev/null)" 2>/dev/null
    rm -f "$pf"
fi
[ -n "$pat" ] && pkill -f "$pat" 2>/dev/null
rm -f "$RUN_DIR/$unit.nrestarts" 2>/dev/null
return 0
}

core_alive() {
    local unit="$1" pf
    if core_unit_exists "$unit"; then core_unit_active "$unit"; return $?; fi
    pf="$(core_pidfile "$unit")"
    [ -f "$pf" ] && kill -0 "$(cat "$pf" 2>/dev/null)" 2>/dev/null
}

core_restarted_since() {
    local unit="$1"
    local f="$RUN_DIR/$unit.nrestarts"
    local before now
    [ -f "$f" ] || return 1
    before="$(cat "$f" 2>/dev/null || echo 0)"
    now="$(core_nrestarts "$unit")"
    [ "${now:-0}" -gt "${before:-0}" ] 2>/dev/null
}

core_healthy() { core_alive "$1" && ! core_restarted_since "$1"; }

_core_listen() {
    local opts="$1" port="$2" ip="${3:-}"
    ss -H $opts 2>/dev/null | awk -v want="$port" -v wip="$ip" '
    { a = $4
      if (a == "") next
      if (match(a, /[0-9]+$/) == 0) next
      p = substr(a, RSTART + 1); h = substr(a, 1, RSTART - 1)
      if (p != want) next
      if (wip == "" || h == wip || h == "0.0.0.0" || h == "*" || h == "[::]" || h == ":::
    }
    END { exit !f }'

```

```

}
core_tcp_listen() { _core_listen "-ltn" "$@"; }
core_udp_listen() { _core_listen "-lun" "$@"; }
core_sctp_listen() { _core_listen "-ln --sctp" "$@"; }
core_vty_listen() { core_tcp_listen "$1" "$CORE_VTY_HOST"; }

core_sctp_estab() {
    ss -nH --sctp 2>/dev/null | awk -v want="$1" '
        $1 == "ESTAB" { a = $4
            if (match(a, /[0-9]+$/)) { p = substr(a, RSTART + 1); if (p == want) { f = 1; ex
            END { exit !f }'
}

core_unix_listen() {
    local p="$1"
    ss -x1H 2>/dev/null | awk -v w="$p" '{ for (i = 1; i <= NF; i++) if ($i == w) { f = 1; ex
        || [ -S "$p" ]
}

core_ip_local() {
    local w="$1"
    case "$w" in ""|0.0.0.0|127.*) return 0 ;; esac
    ip -o -4 addr show 2>/dev/null | awk -v w="$w" '{ split($4, a, "/"); if (a[1] == w) { f =
}

core_vty_ask() {
    local port="$1"; shift
    local body="" c t="${CORE_VTY_TIMEOUT:-10}"
    for c in "$@"; do body+="$c"$'\n'; done
    if command -v socat >/dev/null 2>&1; then
        { sleep "${CORE_VTY_OPEN:-1}"; printf '%s' "$body"; sleep "${CORE_VTY_READ:-2}"; } \
            | timeout "$t" socat -T"$t" STDIO "TCP:${CORE_VTY_HOST}:${port},crlf" 2>/dev/nul
    elif command -v telnet >/dev/null 2>&1; then
        { sleep "${CORE_VTY_OPEN:-1}"; printf '%s' "$body"; sleep "${CORE_VTY_READ:-2}"; } \
            | timeout "$t" telnet "$CORE_VTY_HOST" "$port" 2>/dev/null
    else
        return 1
    fi
}

```

0.3.51 /opt/GSM/qosmo-overlay/run_modules/_lib/i18n.sh

27847 octets, 630 lignes → 630 lignes

```
: "${CALYPSO_LANG:=en}"

declare -gA _T _TM

_T[fr:plan]="Plan - %d modules"
_T[en:plan]="Plan - %d modules"
_T[es:plan]="Plan - %d módulos"
_T[pt:plan]="Plano - %d módulos"
_T[it:plan]="Piano - %d moduli"
_T[hi:plan]="      - %d      "
_T[zh:plan]="      - %d      "
_T[ar:plan]="      - %d      "
_T[fr:ok]="OK"
_T[en:ok]="OK"
_T[es:ok]="OK"
_T[pt:ok]="OK"
_T[it:ok]="OK"
_T[hi:ok]="  "
_T[zh:ok]="  "
_T[ar:ok]="  "
_T[fr:fail]="ÉCHC"
_T[en:fail]="FAIL"
_T[es:fail]="FALL"
_T[pt:fail]="FALH"
_T[it:fail]="ERRO"
_T[hi:fail]="  "
_T[zh:fail]="  "
_T[ar:fail]="  "
_T[fr:skip]="SAUT"
_T[en:skip]="SKIP"
_T[es:skip]="OMIT"
_T[pt:skip]="OMIT"
_T[it:skip]="SALT"
_T[hi:skip]="  "
_T[zh:skip]="  "
_T[ar:skip]="  "
_T[fr:dry]="SIMU"
_T[en:dry]="DRY  "
_T[es:dry]="SIMU"
```

```

_T[pt:dry]="SIMU"
_T[it:dry]="SIMU"
_T[hi:dry]="  "
_T[zh:dry]="  "
_T[ar:dry]="    "
_T[fr:simule]="simulé"
_T[en:simule]="simulated"
_T[es:simule]="simulado"
_T[pt:simule]="simulado"
_T[it:simule]="simulato"
_T[hi:simule]="    "
_T[zh:simule]="  "
_T[ar:simule]="    "
_T[fr:deja]="déjà démarré"
_T[en:deja]="already running"
_T[es:deja]="ya iniciado"
_T[pt:deja]="já iniciado"
_T[it:deja]="già avviato"
_T[hi:deja]="      "
_T[zh:deja]="  "
_T[ar:deja]="    "
_T[fr:desactive]="désactivé par %s"
_T[en:desactive]="disabled by %s"
_T[es:desactive]="desactivado por %s"
_T[pt:desactive]="desativado por %s"
_T[it:desactive]="disattivato da %s"
_T[hi:desactive]="%s      "
_T[zh:desactive]="  %s  "
_T[ar:desactive]="    %s"
_T[fr:interrompu]="séquence interrompue (module obligatoire)"
_T[en:interrompu]="sequence aborted (required module)"
_T[es:interrompu]="secuencia interrumpida (módulo obligatorio)"
_T[pt:interrompu]="sequência interrompida (módulo obrigatório)"
_T[it:interrompu]="sequenza interrotta (modulo obbligatorio)"
_T[hi:interrompu]="      (      )"
_T[zh:interrompu]="      "
_T[ar:interrompu]="    ) "(
_T[fr:journal]="journal : %s"
_T[en:journal]="log : %s"
_T[es:journal]="registro : %s"
_T[pt:journal]="registro : %s"
_T[it:journal]="registro : %s"

```

```

_T[hi:journal]="      : %s"
_T[zh:journal]="      : %s"
_T[ar:journal]= "      : %s"
_T[fr:pas_pret]="démarré mais jamais prêt : %s"
_T[en:pas_pret]="started but never ready: %s"
_T[es:pas_pret]="iniciado pero nunca listo: %s"
_T[pt:pas_pret]="iniciado mas nunca pronto: %s"
_T[it:pas_pret]="avviato ma mai pronto: %s"
_T[hi:pas_pret]="              : %s"
_T[zh:pas_pret]="              %s"
_T[ar:pas_pret]= "              : %s"
_T[fr:sans_raison]="raison non fournie par le module"
_T[en:sans_raison]="no reason given by the module"
_T[es:sans_raison]="el módulo no indicó la causa"
_T[pt:sans_raison]="o módulo não indicou a causa"
_T[it:sans_raison]="il modulo non ha indicato la causa"
_T[hi:sans_raison]="              "
_T[zh:sans_raison]="              "
_T[ar:sans_raison]= "              "
_T[fr:prereq_sim]="prérequis non vérifiable en simulation : %s"
_T[en:prereq_sim]="prerequisite not checkable in simulation: %s"
_T[es:prereq_sim]="requisito no verificable en simulación: %s"
_T[pt:prereq_sim]="requisito não verificável em simulação: %s"
_T[it:prereq_sim]="prerequisito non verificabile in simulazione: %s"
_T[hi:prereq_sim]="              : %s"
_T[zh:prereq_sim]="              %s"
_T[ar:prereq_sim]= "              : %s"
_T[fr:bilan]="%d ok · %d ignorés · %d échecs"
_T[en:bilan]="%d ok · %d skipped · %d failed"
_T[es:bilan]="%d ok · %d omitidos · %d fallos"
_T[pt:bilan]="%d ok · %d omitidos · %d falhas"
_T[it:bilan]="%d ok · %d saltati · %d errori"
_T[hi:bilan]="%d      · %d      · %d      "
_T[zh:bilan]="%d      · %d      · %d      "
_T[ar:bilan]="%d      · %d      · %d      "
_T[fr:opt_inconnue]="option inconnue : %s"
_T[en:opt_inconnue]="unknown option: %s"
_T[es:opt_inconnue]="opción desconocida: %s"
_T[pt:opt_inconnue]="opção desconhecida: %s"
_T[it:opt_inconnue]="opzione sconosciuta: %s"
_T[hi:opt_inconnue]="              : %s"
_T[zh:opt_inconnue]="              %s"

```

```

_TM[ar:opt_inconnue]= " : %s"

_TM[fr:abonnes-hlr]="Cœur - abonnés provisionnés dans le HLR"
_TM[en:abonnes-hlr]="Core - subscribers provisioned in the HLR"
_TM[es:abonnes-hlr]="Núcleo - abonados aprovisionados en el HLR"
_TM[pt:abonnes-hlr]="Núcleo - assinantes provisionados no HLR"
_TM[it:abonnes-hlr]="Core - abbonati registrati nell'HLR"
_TM[hi:abonnes-hlr]=" - HLR "
_TM[zh:abonnes-hlr]=" - HLR "
_TM[ar:abonnes-hlr]= " - HLR"
_TM[fr:apn0]="Cœur - interface TUN de l'APN"
_TM[en:apn0]="Core - APN TUN interface"
_TM[es:apn0]="Núcleo - interfaz TUN del APN"
_TM[pt:apn0]="Núcleo - interface TUN do APN"
_TM[it:apn0]="Core - interfaccia TUN dell'APN"
_TM[hi:apn0]=" - APN TUN "
_TM[zh:apn0]=" - APN TUN "
_TM[ar:apn0]= " - TUN APN"
_TM[fr:asterisk]="Cœur - Asterisk (PBX SIP)"
_TM[en:asterisk]="Core - Asterisk (SIP PBX)"
_TM[es:asterisk]="Núcleo - Asterisk (PBX SIP)"
_TM[pt:asterisk]="Núcleo - Asterisk (PBX SIP)"
_TM[it:asterisk]="Core - Asterisk (PBX SIP)"
_TM[hi:asterisk]=" - Asterisk (SIP PBX)"
_TM[zh:asterisk]=" - Asterisk SIP "
_TM[ar:asterisk]= " - Asterisk ) SIP)"
_TM[fr:attach]="Reprise en main de la session tmux"
_TM[en:attach]="Attaching to the tmux session"
_TM[es:attach]="Reconexión a la sesión tmux"
_TM[pt:attach]="Reconexão à sessão tmux"
_TM[it:attach]="Ripresa della sessione tmux"
_TM[hi:attach]="tmux "
_TM[zh:attach]=" tmux "
_TM[ar:attach]= " tmux"
_TM[fr:audio]="Pont audio (PulseAudio + GAPK)"
_TM[en:audio]="Audio bridge (PulseAudio + GAPK)"
_TM[es:audio]="Puente de audio (PulseAudio + GAPK)"
_TM[pt:audio]="Ponte de áudio (PulseAudio + GAPK)"
_TM[it:audio]="Ponte audio (PulseAudio + GAPK)"
_TM[hi:audio]=" (PulseAudio + GAPK)"
_TM[zh:audio]=" PulseAudio + GAPK "
_TM[ar:audio]= " (PulseAudio + GAPK)"

```

```

_TM[fr:bsc]="Cœur - OsmoBSC (contrôleur BTS)"
_TM[en:bsc]="Core - OsmoBSC (BTS controller)"
_TM[es:bsc]="Núcleo - OsmoBSC (controlador BTS)"
_TM[pt:bsc]="Núcleo - OsmoBSC (controlador BTS)"
_TM[it:bsc]="Core - OsmoBSC (controllore BTS)"
_TM[hi:bsc]="      - OsmoBSC (BTS      )"
_TM[zh:bsc]="      - OsmoBSC      "
_TM[ar:bsc]="      - OsmoBSC      ) BTS)"
_TM[fr:bts]="Station de base osmo-bts-trx"
_TM[en:bts]="Base station osmo-bts-trx"
_TM[es:bts]="Estación base osmo-bts-trx"
_TM[pt:bts]="Estação base osmo-bts-trx"
_TM[it:bts]="Stazione base osmo-bts-trx"
_TM[hi:bts]="      osmo-bts-trx"
_TM[zh:bts]="      osmo-bts-trx"
_TM[ar:bts]="      osmo-bts-trx"
_TM[fr:config]="Résolution des chemins et des ports"
_TM[en:config]="Resolving paths and ports"
_TM[es:config]="Resolución de rutas y puertos"
_TM[pt:config]="Resolução de caminhos e portas"
_TM[it:config]="Risoluzione di percorsi e porte"
_TM[hi:config]="      "
_TM[zh:config]="      "
_TM[ar:config]="      "
_TM[fr:gabarits]="Gabarits de configuration du cœur"
_TM[en:gabarits]="Core configuration templates"
_TM[es:gabarits]="Plantillas de configuración del núcleo"
_TM[pt:gabarits]="Modelos de configuração do núcleo"
_TM[it:gabarits]="Modelli di configurazione del core"
_TM[hi:gabarits]="      "
_TM[zh:gabarits]="      "
_TM[ar:gabarits]="      "
_TM[fr:ggsn]="Cœur - OsmoGGSN (data GTP)"
_TM[en:ggsn]="Core - OsmoGGSN (GTP data)"
_TM[es:ggsn]="Núcleo - OsmoGGSN (datos GTP)"
_TM[pt:ggsn]="Núcleo - OsmoGGSN (dados GTP)"
_TM[it:ggsn]="Core - OsmoGGSN (dati GTP)"
_TM[hi:ggsn]="      - OsmoGGSN (GTP      )"
_TM[zh:ggsn]="      - OsmoGGSN GTP      "
_TM[ar:ggsn]="      - OsmoGGSN      ) GTP)"
_TM[fr:gsmtap]="Capture GSMTAP (pcap)"
_TM[en:gsmtap]="GSMTAP capture (pcap)"

```

```

_TM[es:gsmtap]="Captura GSMTAP (pcap)"
_TM[pt:gsmtap]="Captura GSMTAP (pcap)"
_TM[it:gsmtap]="Cattura GSMTAP (pcap)"
_TM[hi:gsmtap]="GSMTAP      (pcap)"
_TM[zh:gsmtap]="GSMTAP  pcap "
_TM[ar:gsmtap]="      GSMTAP (pcap)"
_TM[fr:hlr]="Cœur - OsmoHLR (registre GSUP)"
_TM[en:hlr]="Core - OsmoHLR (GSUP registry)"
_TM[es:hlr]="Núcleo - OsmoHLR (registro GSUP)"
_TM[pt:hlr]="Núcleo - OsmoHLR (registro GSUP)"
_TM[it:hlr]="Core - OsmoHLR (registro GSUP)"
_TM[hi:hlr]="      - OsmoHLR (GSUP      )"
_TM[zh:hlr]="      - OsmoHLR GSUP      "
_TM[ar:hlr]="      - OsmoHLR      ) GSUP)"
_TM[fr:l2]="Client de couche 2"
_TM[en:l2]="Layer 2 client"
_TM[es:l2]="Cliente de capa 2"
_TM[pt:l2]="Cliente de camada 2"
_TM[it:l2]="Client di livello 2"
_TM[hi:l2]="      2      "
_TM[zh:l2]="      "
_TM[ar:l2]="      "
_TM[fr:logs]="Journaux et espace de travail"
_TM[en:logs]="Logs and working area"
_TM[es:logs]="Registros y espacio de trabajo"
_TM[pt:logs]="Registos e área de trabalho"
_TM[it:logs]="Registri e area di lavoro"
_TM[hi:logs]="      "
_TM[zh:logs]="      "
_TM[ar:logs]="      "
_TM[fr:mgw]="Cœur - OsmoMGW (média MGCP/RTP)"
_TM[en:mgw]="Core - OsmoMGW (MGCP/RTP media)"
_TM[es:mgw]="Núcleo - OsmoMGW (medios MGCP/RTP)"
_TM[pt:mgw]="Núcleo - OsmoMGW (média MGCP/RTP)"
_TM[it:mgw]="Core - OsmoMGW (media MGCP/RTP)"
_TM[hi:mgw]="      - OsmoMGW (MGCP/RTP      )"
_TM[zh:mgw]="      - OsmoMGW MGCP/RTP      "
_TM[ar:mgw]="      - OsmoMGW      ) MGCP/RTP)"
_TM[fr:mobile-cfg]="Configuration du mobile"
_TM[en:mobile-cfg]="Mobile configuration"
_TM[es:mobile-cfg]="Configuración del móvil"
_TM[pt:mobile-cfg]="Configuração do móvel"

```

```

_TM[it:mobile-cfg]="Configurazione del mobile"
_TM[hi:mobile-cfg]="      "
_TM[zh:mobile-cfg]="      "
_TM[ar:mobile-cfg]="      "
_TM[fr:msc]="Cœur - OsmoMSC (commutation CS)"
_TM[en:msc]="Core - OsmoMSC (CS switching)"
_TM[es:msc]="Núcleo - OsmoMSC (conmutación CS)"
_TM[pt:msc]="Núcleo - OsmoMSC (comutação CS)"
_TM[it:msc]="Core - OsmoMSC (commutazione CS)"
_TM[hi:msc]="      - OsmoMSC (CS      )"
_TM[zh:msc]="      - OsmoMSC      "
_TM[ar:msc]="      - OsmoMSC      ) "(
_TM[fr:osmocon]="Passerelle osmocon"
_TM[en:osmocon]="osmocon gateway"
_TM[es:osmocon]="Pasarela osmocon"
_TM[pt:osmocon]="Gateway osmocon"
_TM[it:osmocon]="Gateway osmocon"
_TM[hi:osmocon]="osmocon      "
_TM[zh:osmocon]="osmocon      "
_TM[ar:osmocon]="      osmocon"
_TM[fr:outils]="Outillage de l'hôte"
_TM[en:outils]="Host tooling"
_TM[es:outils]="Herramientas del anfitrión"
_TM[pt:outils]="Ferramentas do anfitrião"
_TM[it:outils]="Strumenti dell'host"
_TM[hi:outils]="      "
_TM[zh:outils]="      "
_TM[ar:outils]="      "
_TM[fr:panes]="Vue de supervision (tmux)"
_TM[en:panes]="Supervision view (tmux)"
_TM[es:panes]="Vista de supervisión (tmux)"
_TM[pt:panes]="Vista de supervisão (tmux)"
_TM[it:panes]="Vista di supervisione (tmux)"
_TM[hi:panes]="      (tmux)"
_TM[zh:panes]="      tmux "
_TM[ar:panes]="      (tmux)"
_TM[fr:pcu]="Cœur - OsmoPCU (ordonnanceur paquet)"
_TM[en:pcu]="Core - OsmoPCU (packet scheduler)"
_TM[es:pcu]="Núcleo - OsmoPCU (planificador de paquetes)"
_TM[pt:pcu]="Núcleo - OsmoPCU (escalonador de pacotes)"
_TM[it:pcu]="Core - OsmoPCU (scheduler pacchetti)"
_TM[hi:pcu]="      - OsmoPCU (      )"

```

```

_TM[zh:pcu]=" - OsmoPCU      "
_TM[ar:pcu]= " - OsmoPCU      ) "(
_TM[fr:prereqs]="Vérification des prérequis"
_TM[en:prereqs]="Prerequisite checks"
_TM[es:prereqs]="Comprobación de requisitos"
_TM[pt:prereqs]="Verificação de requisitos"
_TM[it:prereqs]="Verifica dei prerequisiti"
_TM[hi:prereqs]="          "
_TM[zh:prereqs]="          "
_TM[ar:prereqs]= "          "
_TM[fr:pty]="PTY série de l'émulateur"
_TM[en:pty]="Emulator serial PTY"
_TM[es:pty]="PTY serie del emulador"
_TM[pt:pty]="PTY série do emulador"
_TM[it:pty]="PTY seriale dell'emulatore"
_TM[hi:pty]="          PTY"
_TM[zh:pty]="          PTY"
_TM[ar:pty]= "          PTY          "
_TM[fr:qemu]="Émulateur Calypso (QEMU)"
_TM[en:qemu]="Calypso emulator (QEMU)"
_TM[es:qemu]="Emulador Calypso (QEMU)"
_TM[pt:qemu]="Emulador Calypso (QEMU)"
_TM[it:qemu]="Emulatore Calypso (QEMU)"
_TM[hi:qemu]="Calypso          (QEMU)"
_TM[zh:qemu]="Calypso          QEMU "
_TM[ar:qemu]= "          Calypso (QEMU)"
_TM[fr:reseau]="Interfaces et plan d'adressage"
_TM[en:reseau]="Interfaces and addressing plan"
_TM[es:reseau]="Interfaces y plan de direccionamiento"
_TM[pt:reseau]="Interfaces e plano de endereçamento"
_TM[it:reseau]="Interfacce e piano di indirizzamento"
_TM[hi:reseau]="          -          "
_TM[zh:reseau]="          "
_TM[ar:reseau]= "          "
_TM[fr:restes]="Extinction des processus enregistrés"
_TM[en:restes]="Shutting down recorded processes"
_TM[es:restes]="Cierre de los procesos registrados"
_TM[pt:restes]="Encerramento dos processos registados"
_TM[it:restes]="Arresto dei processi registrati"
_TM[hi:restes]="          "
_TM[zh:restes]="          "
_TM[ar:restes]= "          "

```

```

_TM[fr:rundir]="Répertoire d'exécution"
_TM[en:rundir]="Run directory"
_TM[es:rundir]="Directorio de ejecución"
_TM[pt:rundir]="Diretório de execução"
_TM[it:rundir]="Directory di esecuzione"
_TM[hi:rundir]="      "
_TM[zh:rundir]="  "
_TM[ar:rundir]="  "
_TM[fr:sgsn]="Cœur - OsmoSGSN (rattachement paquet)"
_TM[en:sgsn]="Core - OsmoSGSN (packet attach)"
_TM[es:sgsn]="Núcleo - OsmoSGSN (conexión de paquetes)"
_TM[pt:sgsn]="Núcleo - OsmoSGSN (ligação de pacotes)"
_TM[it:sgsn]="Core - OsmoSGSN (attach pacchetto)"
_TM[hi:sgsn]="  - OsmoSGSN (  )"
_TM[zh:sgsn]="  - OsmoSGSN  "
_TM[ar:sgsn]="  - OsmoSGSN  ) "(
_TM[fr:sidecar-bts]="Station de base BTS#1 (side-car)"
_TM[en:sidecar-bts]="Base station BTS#1 (side-car)"
_TM[es:sidecar-bts]="Estación base BTS#1 (side-car)"
_TM[pt:sidecar-bts]="Estação base BTS#1 (side-car)"
_TM[it:sidecar-bts]="Stazione base BTS#1 (side-car)"
_TM[hi:sidecar-bts]="      BTS#1 (  - )"
_TM[zh:sidecar-bts]="  BTS#1  "
_TM[ar:sidecar-bts]="  BTS#1 "(  )"
_TM[fr:sidecar-cfg]="Configurations du BTS#1 (side-car)"
_TM[en:sidecar-cfg]="BTS#1 configurations (side-car)"
_TM[es:sidecar-cfg]="Configuraciones del BTS#1 (side-car)"
_TM[pt:sidecar-cfg]="Configurações do BTS#1 (side-car)"
_TM[it:sidecar-cfg]="Configurazioni del BTS#1 (side-car)"
_TM[hi:sidecar-cfg]="BTS#1      (  - )"
_TM[zh:sidecar-cfg]="BTS#1      "
_TM[ar:sidecar-cfg]="  BTS#1 "(  )"
_TM[fr:sidecar-faketrx]="Transceiver simulé du BTS#1"
_TM[en:sidecar-faketrx]="BTS#1 simulated transceiver"
_TM[es:sidecar-faketrx]="Transceptor simulado del BTS#1"
_TM[pt:sidecar-faketrx]="Transcetor simulado do BTS#1"
_TM[it:sidecar-faketrx]="Ricetrasmittitore simulato del BTS#1"
_TM[hi:sidecar-faketrx]="BTS#1      "
_TM[zh:sidecar-faketrx]="BTS#1      "
_TM[ar:sidecar-faketrx]="  BTS#1"
_TM[fr:sidecar-mobile]="Second abonné (MS#2)"
_TM[en:sidecar-mobile]="Second subscriber (MS#2)"

```

```

_TM[es:sidecar-mobile]="Segundo abonado (MS#2)"
_TM[pt:sidecar-mobile]="Segundo assinante (MS#2)"
_TM[it:sidecar-mobile]="Secondo abbonato (MS#2)"
_TM[hi:sidecar-mobile]="      (MS#2)"
_TM[zh:sidecar-mobile]="      MS#2 "
_TM[ar:sidecar-mobile]="      (MS#2)"
_TM[fr:sidecar-trxcon]="Couche 1 du MS#2 (trxcon)"
_TM[en:sidecar-trxcon]="MS#2 layer 1 (trxcon)"
_TM[es:sidecar-trxcon]="Capa 1 del MS#2 (trxcon)"
_TM[pt:sidecar-trxcon]="Camada 1 do MS#2 (trxcon)"
_TM[it:sidecar-trxcon]="Livello 1 del MS#2 (trxcon)"
_TM[hi:sidecar-trxcon]="MS#2      1 (trxcon)"
_TM[zh:sidecar-trxcon]="MS#2      trxcon "
_TM[ar:sidecar-trxcon]="      MS#2 (trxcon)"
_TM[fr:sip-connector]="Cœur - passerelle MNCC SIP"
_TM[en:sip-connector]="Core - MNCC SIP gateway"
_TM[es:sip-connector]="Núcleo - pasarela MNCC SIP"
_TM[pt:sip-connector]="Núcleo - gateway MNCC SIP"
_TM[it:sip-connector]="Core - gateway MNCC SIP"
_TM[hi:sip-connector]="      - MNCC SIP      "
_TM[zh:sip-connector]="      - MNCC SIP      "
_TM[ar:sip-connector]="      -      MNCC SIP"
_TM[fr:smc]="Cœur - passerelle SMS (proto-SMSC)"
_TM[en:smc]="Core - SMS gateway (proto-SMSC)"
_TM[es:smc]="Núcleo - pasarela SMS (proto-SMSC)"
_TM[pt:smc]="Núcleo - gateway SMS (proto-SMSC)"
_TM[it:smc]="Core - gateway SMS (proto-SMSC)"
_TM[hi:smc]="      - SMS      (proto-SMSC)"
_TM[zh:smc]="      -      proto-SMSC "
_TM[ar:smc]="      -      (proto-SMSC)"
_TM[fr:stp]="Cœur - OsmoSTP (SS7/M3UA)"
_TM[en:stp]="Core - OsmoSTP (SS7/M3UA)"
_TM[es:stp]="Núcleo - OsmoSTP (SS7/M3UA)"
_TM[pt:stp]="Núcleo - OsmoSTP (SS7/M3UA)"
_TM[it:stp]="Core - OsmoSTP (SS7/M3UA)"
_TM[hi:stp]="      - OsmoSTP (SS7/M3UA)"
_TM[zh:stp]="      - OsmoSTP SS7/M3UA "
_TM[ar:stp]="      - OsmoSTP (SS7/M3UA)"
_TM[fr:summary]="Identité du build et résumé"
_TM[en:summary]="Build identity and summary"
_TM[es:summary]="Identidad de la compilación y resumen"
_TM[pt:summary]="Identidade da compilação e resumo"

```

```

_TM[it:summary]="Identità della build e riepilogo"
_TM[hi:summary]="          "
_TM[zh:summary]="          "
_TM[ar:summary]="          "
_TM[fr:teardown]="Nettoyage du run précédent"
_TM[en:teardown]="Cleaning up the previous run"
_TM[es:teardown]="Limpieza de la ejecución anterior"
_TM[pt:teardown]="Limpeza da execução anterior"
_TM[it:teardown]="Pulizia dell'esecuzione precedente"
_TM[hi:teardown]="          "
_TM[zh:teardown]="          "
_TM[ar:teardown]="          "
_TM[fr:tmux]="Session d'observation tmux"
_TM[en:tmux]="tmux observation session"
_TM[es:tmux]="Sesión de observación tmux"
_TM[pt:tmux]="Sessão de observação tmux"
_TM[it:tmux]="Sessione di osservazione tmux"
_TM[hi:tmux]="tmux          "
_TM[zh:tmux]="tmux          "
_TM[ar:tmux]="          tmux"

_T[fr:arret]="Arrêt de %s"
_T[en:arret]="Stopping %s"
_T[es:arret]="Deteniendo %s"
_T[pt:arret]="A parar %s"
_T[it:arret]="Arresto di %s"
_T[hi:arret]="%s          "
_T[zh:arret]="          %s"
_T[ar:arret]="          %s"

_T[fr:dep_ko]="dépendance non satisfaite : %s"
_T[en:dep_ko]="unmet dependency: %s"
_T[es:dep_ko]="dependencia no satisfecha: %s"
_T[pt:dep_ko]="dependência não satisfeita: %s"
_T[it:dep_ko]="dipendenza non soddisfatta: %s"
_T[hi:dep_ko]="          : %s"
_T[zh:dep_ko]="          %s"
_T[ar:dep_ko]="          :          %s"

_T[fr:obligatoire]="obligatoire"
_T[en:obligatoire]="required"
_T[es:obligatoire]="obligatorio"
_T[pt:obligatoire]="obrigatório"

```

```

_T[it:obligatoire]="obbligatorio"
_T[hi:obligatoire]=" "
_T[zh:obligatoire]=" "
_T[ar:obligatoire]=" "
_T[fr:optionnel]="optionnel"
_T[en:optionnel]="optional"
_T[es:optionnel]="opcional"
_T[pt:optionnel]="opcional"
_T[it:optionnel]="opzionale"
_T[hi:optionnel]=" "
_T[zh:optionnel]=" "
_T[ar:optionnel]=" "
_T[fr:apres]="après: %s"
_T[en:apres]="after: %s"
_T[es:apres]="tras: %s"
_T[pt:apres]="após: %s"
_T[it:apres]="dopo: %s"
_T[hi:apres]=" : %s"
_T[zh:apres]=" %s"
_T[ar:apres]=": " %s"

_T[de:plan]="Plan - %d Module"
_T[de:ok]="OK"
_T[de:fail]="FEHL"
_T[de:skip]="ÜBER"
_T[de:dry]="PROB"
_T[de:simule]="simuliert"
_T[de:deja]="läuft bereits"
_T[de:desactive]="deaktiviert durch %s"
_T[de:interrompu]="Ablauf abgebrochen (Pflichtmodul)"
_T[de:journal]="Protokoll: %s"
_T[de:pas_pret]="gestartet, aber nie bereit: %s"
_T[de:sans_raison]="kein Grund vom Modul angegeben"
_T[de:prereq_sim]="Voraussetzung im Probelauf nicht prüfbar: %s"
_T[de:bilan]="%d ok · %d übersprungen · %d Fehler"
_T[de:opt_inconnue]="unbekannte Option: %s"
_T[de:arret]="%s wird beendet"
_T[de:dep_ko]="Abhängigkeit nicht erfüllt: %s"
_T[de:obligatoire]="Pflicht"
_T[de:optionnel]="optional"
_T[de:apres]="nach: %s"
_TM[de:abonnes-hlr]="Kern - im HLR eingerichtete Teilnehmer"

```

```

_TM[de:apn0]="Kern - TUN-Schnittstelle des APN"
_TM[de:asterisk]="Kern - Asterisk (SIP-Telefonanlage)"
_TM[de:attach]="Rückkehr in die tmux-Sitzung"
_TM[de:audio]="Audiobrücke (PulseAudio + GAPK)"
_TM[de:bsc]="Kern - OsmoBSC (BTS-Steuerung)"
_TM[de:bts]="Basisstation osmo-bts-trx"
_TM[de:config]="Auflösung von Pfaden und Ports"
_TM[de:gabarits]="Konfigurationsvorlagen des Kerns"
_TM[de:ggsn]="Kern - OsmoGGSN (GTP-Daten)"
_TM[de:gsmtap]="GSMTAP-Mitschnitt (pcap)"
_TM[de:hlr]="Kern - OsmoHLR (GSUP-Register)"
_TM[de:l2]="Schicht-2-Client"
_TM[de:logs]="Protokolle und Arbeitsbereich"
_TM[de:mgw]="Kern - OsmoMGW (MGCP/RTP-Medien)"
_TM[de:mobile-cfg]="Konfiguration des Mobilteils"
_TM[de:msc]="Kern - OsmoMSC (leitungsvermittelt)"
_TM[de:osmocon]="osmocon-Übergang"
_TM[de:outils]="Werkzeuge des Wirtssystems"
_TM[de:panes]="Überwachungsansicht (tmux)"
_TM[de:pcu]="Kern - OsmoPCU (Paketzuteilung)"
_TM[de:prereqs]="Prüfung der Voraussetzungen"
_TM[de:pty]="Seriell PTY des Emulators"
_TM[de:qemu]="Calypso-Emulator (QEMU)"
_TM[de:reseau]="Schnittstellen und Adressplan"
_TM[de:restes]="Beenden der erfassten Prozesse"
_TM[de:rundir]="Ausführungsverzeichnis"
_TM[de:sgsn]="Kern - OsmoSGSN (Paketanbindung)"
_TM[de:sidecar-bts]="Basisstation BTS#1 (Beiwagen)"
_TM[de:sidecar-cfg]="Konfigurationen der BTS#1 (Beiwagen)"
_TM[de:sidecar-faketrx]="Simulierter Transceiver der BTS#1"
_TM[de:sidecar-mobile]="Zweiter Teilnehmer (MS#2)"
_TM[de:sidecar-trxcon]="Schicht 1 des MS#2 (trxcon)"
_TM[de:sip-connector]="Kern - MNCC SIP-Übergang"
_TM[de:smsc]="Kern - SMS-Übergang (proto-SMSC)"
_TM[de:stp]="Kern - OsmoSTP (SS7/M3UA)"
_TM[de:summary]="Build-Kennung und Zusammenfassung"
_TM[de:teardown]="Aufräumen des vorherigen Laufs"
_TM[de:tmux]="tmux-Beobachtungssitzung"

_T[fr:up]="ACTIF"
_T[de:up]="AKTIV"
_T[en:up]="UP"

```

```

_T[es:up]="ACTIVO"
_T[pt:up]="ATIVO"
_T[it:up]="ATTIVO"
_T[hi:up]="  "
_T[zh:up]="  "
_T[ar:up]="  "
_T[fr:down]="ARRÊT"
_T[de:down]="AUS"
_T[en:down]="DOWN"
_T[es:down]="PARADO"
_T[pt:down]="PARADO"
_T[it:down]="FERMO"
_T[hi:down]="  "
_T[zh:down]="  "
_T[ar:down]="  "
_T[fr:connecter]="se connecter"
_T[de:connecter]="verbinden"
_T[en:connecter]="attach"
_T[es:connecter]="conectar"
_T[pt:connecter]="ligar"
_T[it:connecter]="collegarsi"
_T[hi:connecter]="  "
_T[zh:connecter]="  "
_T[ar:connecter]="  "
_T[fr:pas_session]="(aucune session tmux « %s » - la pile tourne en arrière-plan)"
_T[de:pas_session]="(keine tmux-Sitzung „%s“ - der Stapel läuft im Hintergrund)"
_T[en:pas_session]="(no tmux session \"%s\" - the stack runs in the background)"
_T[es:pas_session]="(sin sesión tmux «%s» - la pila corre en segundo plano)"
_T[pt:pas_session]="(sem sessão tmux «%s» - a pilha corre em segundo plano)"
_T[it:pas_session]="(nessuna sessione tmux «%s» - lo stack gira in background)"
_T[hi:pas_session]="(  tmux  «%s»  -  )"
_T[zh:pas_session]="(  tmux  %s -  )"
_T[ar:pas_session]=" )"      tmux «%s» -  "(
_T[fr:journals]="journaux"
_T[de:journals]="Protokolle"
_T[en:journals]="logs"
_T[es:journals]="registros"
_T[pt:journals]="registros"
_T[it:journals]="registri"
_T[hi:journals]="  "
_T[zh:journals]="  "
_T[ar:journals]="  "

```

```

_T[fr:etat]="état"
_T[de:etat]="Status"
_T[en:etat]="status"
_T[es:etat]="estado"
_T[pt:etat]="estado"
_T[it:etat]="stato"
_T[hi:etat]=" "
_T[zh:etat]=" "
_T[ar:etat]=" "
_T[fr:arreter]="arrêter"
_T[de:arreter]="beenden"
_T[en:arreter]="stop"
_T[es:arreter]="detener"
_T[pt:arreter]="parar"
_T[it:arreter]="fermare"
_T[hi:arreter]=" "
_T[zh:arreter]=" "
_T[ar:arreter]=" "
_T[fr:aucun_mod]="aucun module dans %s - rien à faire"
_T[de:aucun_mod]="kein Modul in %s - nichts zu tun"
_T[en:aucun_mod]="no module in %s - nothing to do"
_T[es:aucun_mod]="ningún módulo en %s - nada que hacer"
_T[pt:aucun_mod]="nenhum módulo em %s - nada a fazer"
_T[it:aucun_mod]="nessun modulo in %s - niente da fare"
_T[hi:aucun_mod]="%s - "
_T[zh:aucun_mod]="%s - "
_T[ar:aucun_mod]=" %s - "

t() {
    local cle="$1"; shift
    local fmt="${_T[${CALYPSO_LANG}]:$cle}:-${_T[en:$cle]}:-${_T[fr:$cle]}:-$cle}"
    printf "$fmt" "$@"
}

t_mod() {
    printf '%s' "${_TM[${CALYPSO_LANG}]:$1}:-${_TM[en:$1]}:-${_TM[fr:$1]}:-${MOD_DESC[$1]}:-$1}"
}

case " fr de en es pt it hi zh ar " in
    *) $CALYPSO_LANG *) ;;
    *) printf 'CALYPSO_LANG=%s inconnue - repli sur « en » (fr de en es pt it hi zh ar)\n' \
        "$CALYPSO_LANG" >&2

```

```
CALYPSO_LANG=en ;;  
esac
```

0.3.52 /opt/GSM/qosmo-overlay/run_modules/_lib/mod.sh

2363 octets, 80 lignes → 80 lignes

```
. "$(dirname "${BASH_SOURCE[0]}")/i18n.sh"  
  
readonly MOD_RC_OK=0  
readonly MOD_RC_FAIL=1  
readonly MOD_RC_UNKNOWN=2  
readonly MOD_RC_ALREADY=3  
readonly MOD_RC_SKIP=4  
  
declare -a MOD_ORDER=()  
declare -A MOD_DESC=()  
declare -A MOD_DEPS=()  
declare -A MOD_REQUIRED=()  
declare -A MOD_LOG=()  
declare -A MOD_JOURNAL=()  
declare -A MOD_PURE=()  
declare -A MOD_ENABLED_IF=()  
declare -A MOD_TIMEOUT=()  
  
MOD_REGISTER() {  
    local slug="$1" desc="$2"  
    MOD_ORDER+=("$slug")  
    MOD_DESC[$slug]="$desc"  
    MOD_DEPS[$slug]=""  
    MOD_REQUIRED[$slug]=1  
    MOD_LOG[$slug]=""  
    MOD_JOURNAL[$slug]=""  
    MOD_PURE[$slug]=0  
    MOD_ENABLED_IF[$slug]='true'  
    MOD_TIMEOUT[$slug]=30  
}  
  
mod_prefix() { printf 'mod_%s' "${1//-/}_"; }  
  
_MOD_REASON=""
```

```

_MOD_HINT=""

mod_ok()      { _MOD_REASON="" ; return $MOD_RC_OK; }
mod_fail()    { _MOD_REASON="$*" ; return $MOD_RC_FAIL; }
mod_already() { _MOD_REASON="$*" ; return $MOD_RC_ALREADY; }
mod_skip()    { _MOD_REASON="$*" ; return $MOD_RC_SKIP; }
mod_hint()    { _MOD_HINT="$*"; }
mod_say()     { printf '%s\n' "$*"; }

wait_until() {
    local timeout="$1" what="$2"; shift 2
    local deadline=$(( SECONDS + timeout ))
    while (( SECONDS < deadline )); do
        if "$@" >/dev/null 2>&1; then return $MOD_RC_OK; fi
        sleep 0.2
    done
    mod_fail "$what : toujours pas prêt après ${timeout}s"
}

have_proc() {
    local motif="$1" p ligne ns
    ns="$(readlink /proc/self/ns/pid 2>/dev/null)"
    for p in $(pgrep -f "$motif" 2>/dev/null); do
        [ "$p" = "$$" ] || [ "$p" = "$PPID" ] && continue
        [ -z "$ns" ] || [ "$(readlink "/proc/$p/ns/pid" 2>/dev/null)" = "$ns" ] || continue
        ligne="$( { tr '\0' ' ' < "/proc/$p/cmdline"; } 2>/dev/null )"
        [ -n "$ligne" ] || continue
        case "$ligne" in
            *"tail"|"*"tail -*"|"*"less"|"*"multitail"|"*"watch"|"*\
            *"sleep infinity"|"*"journalctl"*) continue ;;
        esac
        return 0
    done
    return 1
}

have_port() { (exec 3<>"/dev/tcp/127.0.0.1/$1") >/dev/null 2>&1; }
have_unix() { [ -S "$1" ]; }
log_has()   { [ -f "$1" ] && grep -q "$2" "$1" 2>/dev/null; }

modb_tail() {
    local f="$1" n="${2:-20}"
    if [ ! -r "$f" ]; then mod_say "journal absent ou illisible : $f"; return 0; fi

```

```

mod_say "--- $n dernières lignes de $f ---"
tail -n "$n" "$f" 2>/dev/null
mod_say "--- fin de $f ---"
}

```

0.3.53 /opt/GSM/qosmo-overlay/run_modules/_lib/modem.sh

1996 octets, 35 lignes → 35 lignes

```

[ -n "${_MODEM_LIB_LOADED:-}" ] && return 0
_MODEM_LIB_LOADED=1
# _lib/modem.sh - le MS GPRS du banc : le « modem » d'osmocom-bb (GMM/SM/SNDCP/LLC)
# branche comme SECOND MS sur le fake_trx du side-car (BTS#1), a cote de MS#2.
# fake_trx sait servir plusieurs MS (--trx) : aucun multiplexeur maison.
#   BTS#1 (bts 1, TS5-7 PDCH) --TRXD 5720--> fake_trx --6720--> trxcon MS#2 (voix)
#                                     \--6740--> trxcon modem (GPRS)
#   BTS#1 --/tmp/pcu_bts2--> osmo-pcu (second, nsei 11) --Gb--> osmo-sgsn --> osmo-ggsn (apn)
# MODEM_GPRS=0 coupe tout le lot (start-native.sh --no-gprs-modem).
: "${MODEM_GPRS:=1}"
: "${MODEM_BIN:=${OSMOCOM_BB:-${GSM_ROOT:-/opt/GSM}/osmocom-bb}/src/host/layer23/src/modem/modem}"
: "${MODEM_BB_PORT:=6740}" # base TRXD cote modem (fake_trx --trx ...:6740, trxcon ...)
: "${MODEM_L2_SOCKET:=/tmp/modem_l2}"
: "${MODEM_VTY_PORT:=4299}"
: "${MODEM_CFG:=${OSMOCOM_CFG:-/etc/osmocom}/modem-gprs.cfg}"
: "${MODEM_TUN:=tun_modem}"
: "${MODEM_MS_IDX:=3}" # 3e abonne : IMSI ...000003, MSISDN <noeud>00<op>03
: "${MODEM_APN:=internet}"
: "${MODEM_TRXCON_FN_ADVANCE:=100}" # -F : extension du delai FBSB (comme MS#2)
: "${MODEM_TRXCON_TRX_ADVANCE:=0}" # -f : avance de tirage UL ; 0 obligatoire sur PDCH (voix)
# Second PCU, celui du BTS#1 (le premier sert le BTS principal sur /tmp/pcu_bts).
: "${SC_PCU_UNIT:=osmo-pcu-bts1}"
: "${SC_PCU_SOCKET:=/tmp/pcu_bts2}"
: "${SC_PCU_CFG:=${OSMOCOM_CFG:-/etc/osmocom}/osmo-pcu-bts1.cfg}"
: "${SC_PCU_VTY_HOST:=127.0.0.2}" # meme port 4240 que le premier PCU, autre adresse
: "${SC_PCU_VTY_PORT:=4240}"
# Gb du BTS#1 (bloc gprs de « bts 1 », cf. 13-sidecar-cfg.sh) : distinct du BTS principal (n)
: "${SC_GPRS_NSEI:=11}"
: "${SC_GPRS_NSVC1:=11}"
: "${SC_GPRS_BVCI:=13}"
: "${SC_GPRS_NS_PORT:=23002}"
: "${SC_SGSN_IP:=192.168.2.10}"

```

```
: "${SC_SGSN_PORT:=23000}"
```

```
modem_enabled() { [ "${MODEM_GPRS:-1}" = 1 ]; }
```

0.3.54 /opt/GSM/qosmo-overlay/run_modules/_lib/radio.sh

1986 octets, 59 lignes → 59 lignes

```
[ -n "${_RADIO_LIB_LOADED:-}" ] && return 0
_RADIO_LIB_LOADED=1
```

```
: "${SC_TRX_PORT:=5720}"
: "${SC_BB_PORT:=6720}"
: "${SC_BTS_VTY_PORT:=4250}"
: "${SC_MOBILE_VTY_PORT:=4248}"
: "${SC_UNIT_ID:=6002}"
: "${SC_ARFCN:=516}"
: "${SC_BTS_CFG:=${OSMOCOM_CFG:-/etc/osmocom}/osmo-bts-trx-bts1.cfg}"
: "${SC_MOBILE_CFG:=/root/.osmocom/bb/mobile_faketrx_bts1.cfg}"
: "${SC_L2_SOCKET:=/tmp/ms2_l2}"
: "${TRX_BIND_IP:=127.0.0.1}"
: "${TRX_BTS_IP:=127.0.0.1}"
: "${TRX_BB_IP:=127.0.0.1}"
: "${RUN_DIR:=/tmp/calypso}"
: "${LOG_DIR:=$RUN_DIR/logs}"
```

```
radio_dirs() { mkdir -p "$RUN_DIR" "$LOG_DIR" 2>/dev/null || true; }
```

```
radio_log() { printf '%s/%s.log\n' "$LOG_DIR" "$1"; }
```

```
radio_pidfile() { printf '%s/%s.pid\n' "$RUN_DIR" "$1"; }
```

```
radio_save_pid() { radio_dirs; printf '%s\n' "$2" > "$(radio_pidfile "$1")"; }
```

```
radio_pid() { cat "$(radio_pidfile "$1")" 2>/dev/null || echo 0; }
```

```
radio_alive() { local p; p="$(radio_pid "$1")"; [ "$p" != 0 ] && kill -0 "$p" 2>/dev/null; }
```

```
radio_kill() {
```

```
    local slug="$1" pattern="${2:-}" p i
```

```
    p="$(radio_pid "$slug")"
```

```
    [ "$p" != 0 ] && kill "$p" 2>/dev/null
```

```
    [ -n "$pattern" ] && pkill -f "$pattern" 2>/dev/null
```

```
    for i in 1 2 3 4 5 6 7 8 9 10; do
```

```
        if [ "$p" = 0 ] || ! kill -0 "$p" 2>/dev/null; then break; fi
```

```
        sleep 0.2
```

```

done
if [ "$p" != 0 ] && kill -0 "$p" 2>/dev/null; then
    kill -9 "$p" 2>/dev/null
    [ -n "$pattern" ] && pkill -9 -f "$pattern" 2>/dev/null
fi
rm -f "$(radio_pidfile "$slug")"
return 0
}

radio_udp_bound() {
    ss -uan 2>/dev/null | grep -qE "[:.]\$1[[:space:]]"
}

radio_cfg_val() {
    local f="$1"; shift
    [ -r "$f" ] || return 1
    awk -v k="$*" '
    {
        line = $0
        sub(/^[[[:space:]]+/, "", line); sub(/[[:space:]]+$/, "", line)
        n = split(k, kk, /[[:space:]]+/)
        split(line, ff, /[[:space:]]+/)
        for (i = 1; i <= n; i++) if (ff[i] != kk[i]) next
        if (ff[n+1] != "") { print ff[n+1]; exit }
    }' "$f"
}

```

0.3.55 /opt/GSM/qosmo-overlay/run_modules/_lib/temps_reel.sh

1999 octets, 39 lignes → 39 lignes

```

[ -n "${_TEMPS_REEL_LIB_LOADED:-}" ] && return 0
_TEMPS_REEL_LIB_LOADED=1

# Priorité temps réel (SCHED_RR) pour les process qui ont une échéance dure.
#
# [2026-10-04] Le BTS#1 (side-car) s'est arrêté en pleine campagne banc-max :
#   DL1C ERROR PC clock skew: elapsed_us=231775, error_us=227160
#   BTS_SHUTDOWN(bts0): Shutting down BTS, exit 1, reason: PC clock skew too high
# Son timer de trame (4,615 ms) a été servi 227 ms trop tard. En SCHED_OTHER,
# sous charge (load ~11 ce soir-là), un process peut attendre aussi longtemps,

```

```

# et osmo-bts-trx se coupe de lui-même au-delà. Le BTS#0 frôle la même limite
# (« We missed 1 timers » en continu dans bts.log).
#
# Par chrt AU LANCEMENT, et pas par « cpu-sched / policy rr 1 » dans le .cfg :
# avec le .cfg, osmo-bts-trx 1.10 REFUSE DE DÉMARRER quand le droit manque
# (« Failed setting SCHED_RR priority 1 », exit 1 - vérifié sans CAP_SYS_NICE
# ni rtprio). Ici on essaie d'abord : sans le droit, on lance comme avant.
# Le droit : root (CAP_SYS_NICE) sur l'hôte ; « --ulimit rtprio=18 » dans les
# conteneurs (start-docker.sh, compose.yaml).
#
#   QOSMO_RT_PRIO=<1-99>  priorité SCHED_RR (défaut 1 : au-dessus de tout
#                           SCHED_OTHER, sous le reste du temps réel)
#   QOSMO_RT_PRIO=0       désactivé : SCHED_OTHER, comme avant

# Préfixe à placer devant la commande : « chrt -r <prio> », ou rien.
# Se développe SANS guillemets : setsid stdbuf -oL $(rt_prefixe) "$BIN" ...
# chrt fait un exec : le pid lancé reste celui du binaire, et les threads
# qu'il créera héritent de la politique.
rt_prefixe() {
    local prio="{QOSMO_RT_PRIO:-1}"
    [ "$prio" -gt 0 ] 2>/dev/null || return 0
    chrt -r "$prio" true 2>/dev/null && printf 'chrt -r %s\n' "$prio"
    return 0
}

# Ordonnancement effectif d'un process, pour les journaux : « SCHED_RR 1 ».
rt_etat() { # $1 = pid
    LC_ALL=C chrt -p "$1" 2>/dev/null | awk -F': ' '/policy/ { p = $2 } /priority/ { print p }'
}

```

0.3.56 /opt/GSM/qosmo-overlay/scripts/inject_fb.py

2925 octets, 90 lignes → 90 lignes

```

#!/usr/bin/env python3
"""
inject_fb.py - diagnostic script: send a clean FB burst (148 zero bits)
straight to QEMU's BSP UDP port (127.0.0.1:6702), bypassing osmo-bts-trx.

Purpose: confirm whether the DSP correlator at PROMO 0x82f6 actually
writes d_fb_det when fed a perfect frequency burst. If yes → the loop

```

in production runs is failing on amplitude/phase/timing of the bridge's GMSK simulation. If no → the DSP code path itself is broken.

We tail bridge.log to track QEMU's current FN (the bridge prints "bridge: QEMU tick #N FN=X" on every TDMA tick) and send each burst slightly in the future of the live FN so it lands in the BSP match window (± 64).

```
"""
import os
import re
import socket
import struct
import sys
import time

(removed) = "/tmp/bridge.log"
BSP_ADDR = ("127.0.0.1", 6702)
LOOKAHEAD = 30          # frames ahead of cur_fn (well inside  $\pm 64$  window)
SEND_PERIOD_S = 0.004   # ~one TDMA frame
DURATION_S = 30

def make_fb_burst(tn, fn, rssi=20, toa=0):
    """TRXDv0 DL: tn(1) fn(4 BE) rssi(1) toa(2 BE) + 148 zero bits."""
    return (
        bytes([tn & 0x07])
        + struct.pack(">I", fn & 0xFFFFFFFF)
        + bytes([rssi])
        + struct.pack(">H", toa & 0xFFFF)
        + bytes(148)
    )

def latest_fn_from_bridge():
    """Walk bridge.log backwards looking for the most recent 'fn=N' tag."""
    try:
        with open((removed), "rb") as f:
            f.seek(0, 2)
            size = f.tell()
            chunk = 8192
            f.seek(max(0, size - chunk))
            tail = f.read().decode("utf-8", errors="replace")
            # Prefer DL bursts (carry bts_fn and our rewritten fn) for freshness
            m = list(re.finditer(r"\bfn=(\d+)", tail))
```

```

        if m:
            return int(m[-1].group(1))
    except FileNotFoundError:
        pass
    return None

def main():
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    print(f"inject_fb: target={BSP_ADDR} lookahead={LOOKAHEAD} duration={DURATION_S}s",
          flush=True)

    # Wait until bridge.log has at least one fn= line
    start = time.time()
    while time.time() - start < 10:
        fn = latest_fn_from_bridge()
        if fn is not None:
            break
        time.sleep(0.2)
    else:
        print("inject_fb: no fn= seen in bridge.log within 10s", file=sys.stderr)
        sys.exit(1)
    print(f"inject_fb: starting FN={fn}", flush=True)

    sent = 0
    end = time.time() + DURATION_S
    while time.time() < end:
        cur = latest_fn_from_bridge()
        if cur is None:
            cur = fn
        target_fn = cur + LOOKAHEAD
        sock.sendto(make_fb_burst(tn=0, fn=target_fn), BSP_ADDR)
        sent += 1
        if sent % 250 == 0:
            print(f"inject_fb: sent={sent} cur_fn~={cur} target_fn={target_fn}",
                  flush=True)
        time.sleep(SEND_PERIOD_S)

    print(f"inject_fb: done sent={sent}", flush=True)

if __name__ == "__main__":
    main()

```

0.3.57 /opt/GSM/qosmo-overlay/scripts/make_dsp_bin_L1.py

8647 octets, 233 lignes → 227 lignes (3 groupes compactés)

```
#!/usr/bin/env python3

# --- racine de l'installation, resolue sans chemin en dur -----
# GSM_ROOT si l'environnement le pose (c'est le cas quand on passe par run.sh),
# sinon le parent du depot : les deux depots vivent cote a cote.
import os as _os
GSM_ROOT = _os.environ.get(
    "GSM_ROOT",
    _os.path.dirname(_os.path.dirname(_os.path.abspath(__file__))))

"""
make_dsp_bin_L1.py - Generate calypso_dsp.PROM0.bin with synthetic L1+L2 publisher.

Scope (extension 2026-05-28 : L1 → L2) :
- FB lock : d_fb_det + a_sync_demod[4] cells (TOA/PM/ANGLE/SNR)
- PM measurement : a_pm[3] WP0/WP1 with canned -85 dBm value
- SB lock : a_sch[5] WP0/WP1 with canned BSIC matching osmo-bsc.cfg
- BCCH SI3 (L2) : a_cd[0..14] with FIRE-OK status + 23-byte SI3 LAPDm frame
  (same canned bytes as CALYPSO_DSP_SHUNT shunt_dispatch_allc, MCC=001/MNC=01)

Entry point : 0x7120 (= SILICON-BOOT-REDIRECT target from c54x.c).
Patches existing PROM0.bin at offset 0x240 (= word 0x120, code addr 0x7120).
Rest of ROM preserved.

Architecture :
- Boot init at 0x7120 : sets SP=0x5AC8, IMR=0xFFFF
- Main loop : tight loop writing all cells continuously
- ARM reads cells anytime, always sees canned "lock detected" values

Strategy : continuous-write rather than dispatch-on-task. Simpler than full
dispatcher, plus ARM cell-read race is auto-resolved (whatever read happens
sees current canned value).

Usage :
    python3 make_dsp_bin_L1.py [input.bin] [output.bin]
    Default input   : /opt/GSM/calypso_dsp.PROM0.bin
    Default output  : /opt/GSM/calypso_dsp.PROM0.bin.L1
"""
import sys
```

```

import struct

INPUT  = sys.argv[1] if len(sys.argv) > 1 else f'{GSM_ROOT}/calypso_dsp.PROM0.bin'
# *** OUTPUT path EXPLICITLY separate - JAMAIS écrase l'original ***
OUTPUT = sys.argv[2] if len(sys.argv) > 2 else '/tmp/calypso_dsp_L1stub.PROM0.bin'

# ---- C54x opcode encoders ----

def stm_lk_mmr(lk, mmr):
    """STM #lk, MMR - 2 words (0x7700 | MMR_low7, lk)"""
    return [0x7700 | (mmr & 0x7F), lk & 0xFFFF]

def ld_lk_a(lk):
    """LD #lk, A - 2 words ALU class F0xx (alu_op=2 LD)"""
    # bits 7:4 = 0010 = LD ; bit 8 = SRC ; bit 9 = DST
    # Use src=A dst=A : bit 8 = 0 bit 9 = 0
    # opcode 0xF0 << 8 | (alu_op << 4) | shift = 0xF020 (shift=0)
    return [0xF020, lk & 0xFFFF]

def stl_a_arn(n, mod):
    """STL A, *ARn[mod] - 1 word (0x80xx hi8 = STL A)
       mod {0=*ARn, 1=*ARn-, 2=*ARn+}"""
    smem = 0x80 | ((mod & 0xF) << 3) | (n & 0x7)
    return [(0x80 << 8) | smem]

def b_unc(target):
    """B pmad (unconditional NEAR branch) - 2 words.
       Use 0xF880 (= FB FAR in c54x.c emulator but sets XPC=0 always,
       which is no-op when staying within PROM0). Same encoding as the
       SILICON-BOOT-REDIRECT vector synthesis in calypso_c54x.c.
       NOT 0xF800 : that's BANZ ARO (= conditional on ARO!=0) in this emu."""
    return [0xF880, target & 0xFFFF]

# ---- MMR table ----
MMR_IMR = 0x00
MMR_ARO = 0x10
MMR_SP  = 0x18

# ---- Cell addresses (DSP code space) ----
# API base = 0x0800
# NDB base = 0x08D4 (= byte 0x01A8 / 2)
D_FB_DET    = 0x08F8 # NDB + 0x48 byte

```

```

D_FB_MODE      = 0x08F9      # NDB + 0x4A byte
A_SYNC_DEMOD    = 0x08FA      # NDB + 0x4C..0x53 byte = 4 words
A_PM_WPO        = 0x0830      # API + 0x60 byte = read page 0 word 8
A_PM_WP1        = 0x0844      # API + 0x88 byte = read page 1 word 8
# a_sch[5] at words 15..19 of read page (after a_pm[3] words 8..10 + a_serv_demod[4] words 1
A_SCH_WPO       = 0x0837      # RP0 + 15 word = 0x0828 + 15
A_SCH_WP1       = 0x084B      # RP1 + 15 word = 0x083C + 15
# a_cd[0..14] = CCCH demod result.
# Per CLAUDE.md DWARF-validated 2026-05-26 : NDB+0x1DC byte = NDB+0xEE word
# = code 0x08D4 + 0xEE = 0x09C2. Le shunt utilise 0x1FC (= probablement une
# struct variant alternative). On suit le DWARF qui est l'autorité firmware.
A_CD_BASE       = 0x09C2      # a_cd[0]
A_CD_PAYLOAD    = 0x09C5      # a_cd[3] = start of 12-word SI3 packed payload

# ---- Canned values ----
FB_DET_VAL      = 0x4E20      # 20000 = nonzero detection flag
TOA_VAL         = 0x0240      # 576 (mid burst window)
PM_VAL          = 0xFB00      # signed -1280, firmware >>3 = -160, Power = (-160-520)/8 = -85 dB
ANGLE_VAL       = 0x0000      # AFC error = 0 Hz
SNR_VAL         = 0x4E20      # 20000 = above FB1_SNR_THRESH=3000

# SB encoding for BSIC=7, t1=0, t2=0, t3=1 : sb = (bsic & 0x3f) << 2 = 0x1C
SB_BSIC         = 7
SB_PACKED       = (SB_BSIC & 0x3F) << 2 # = 0x1C
SB_LO           = SB_PACKED & 0xFFFF    # = 0x001C
SB_HI           = (SB_PACKED >> 16) & 0xFFFF # = 0x0000

# ---- Canned SI3 L2 frame (23 bytes, identical to shunt_dispatch_allc) ----
SI3_BYTES = [
    0x49, 0x06, 0x1B,          # L2 hdr + RR PD + SI3 mt
    0x00, 0x01,                # Cell ID = 1
    0x00, 0xF1, 0x10,          # MCC=001 MNC=01
    0x00, 0x01,                # LAC = 1
    0x01, 0x00,                # cell opts + cell select
    0x18, 0xFF, 0xFF,          # RACH ctrl
    0x2B, 0x2B, 0x2B, 0x2B, 0x2B, 0x2B, 0x2B, 0x2B
]
assert len(SI3_BYTES) == 23

# Pack 23 bytes → 12 LE words (lo + hi<<8); last word pads with 0x2B
SI3_WORDS = []
for i in range(0, 23, 2):

```

```

    lo = SI3_BYTES[i]
    hi = SI3_BYTES[i+1] if i+1 < 23 else 0x2B
    SI3_WORDS.append(lo | (hi << 8))
assert len(SI3_WORDS) == 12

# ---- Build code ----
code = []

# Boot init
code += stm_lk_mmr(0x5AC8, MMR_SP)          # STM #0x5AC8, SP
code += stm_lk_mmr(0xFFFF, MMR_IMR)        # STM #0xFFFF, IMR

# Compute main_loop address now that we know boot prefix length
main_loop_addr = 0x7120 + len(code)

# --- d_fb_det = 0x4E20 ---
code += stm_lk_mmr(D_FB_DET, MMR_AR0)
code += ld_lk_a(FB_DET_VAL)
code += stl_a_arn(0, 0)                     # STL A, *AR0 (no mod)

# --- a_sync_demod[0..3] = TOA, PM, ANGLE, SNR ---
code += stm_lk_mmr(A_SYNC_DEMOD, MMR_AR0)
code += ld_lk_a(TOA_VAL)
code += stl_a_arn(0, 2)                     # STL A, *AR0+ (post-inc)
code += ld_lk_a(PM_VAL)
code += stl_a_arn(0, 2)
code += ld_lk_a(ANGLE_VAL)
code += stl_a_arn(0, 2)
code += ld_lk_a(SNR_VAL)
code += stl_a_arn(0, 2)

# --- a_pm[3] WP0 ---
code += stm_lk_mmr(A_PM_WP0, MMR_AR0)
code += ld_lk_a(PM_VAL)
code += stl_a_arn(0, 2) *3

# --- a_pm[3] WP1 (reuse A) ---
code += stm_lk_mmr(A_PM_WP1, MMR_AR0)
code += stl_a_arn(0, 2) *3

# --- a_sch[5] WP0 : a_sch[0]=0 (CRC OK), [1]=0, [2]=0, [3]=SB_L0, [4]=SB_HI ---
code += stm_lk_mmr(A_SCH_WP0, MMR_AR0)

```

```

code += ld_lk_a(0x0000)
code += stl_a_arn(0, 2) # a_sch[0] = 0 (CRC OK)
code += stl_a_arn(0, 2) # a_sch[1] = 0
code += stl_a_arn(0, 2) # a_sch[2] = 0
code += ld_lk_a(SB_LO)
code += stl_a_arn(0, 2) # a_sch[3] = SB low
code += ld_lk_a(SB_HI)
code += stl_a_arn(0, 2) # a_sch[4] = SB high

# --- a_sch[5] WP1 same ---
code += stm_lk_mmr(A_SCH_WP1, MMR_AR0)
code += ld_lk_a(0x0000)
code += stl_a_arn(0, 2) *3
code += ld_lk_a(SB_LO)
code += stl_a_arn(0, 2)
code += ld_lk_a(SB_HI)
code += stl_a_arn(0, 2)

# --- a_cd[0..2] = FIRE-OK status (CRC pass, no biterr) ---
code += stm_lk_mmr(A_CD_BASE, MMR_AR0)
code += ld_lk_a(0x0000)
code += stl_a_arn(0, 2) # a_cd[0] = FIRE bits = 0 (CRC OK)
code += stl_a_arn(0, 2) # a_cd[1] = reserved = 0
code += stl_a_arn(0, 2) # a_cd[2] = num_biterr = 0

# --- a_cd[3..14] = 12 words of SI3 LAPDm packed payload ---
code += stm_lk_mmr(A_CD_PAYLOAD, MMR_AR0)
for w in SI3_WORDS:
    code += ld_lk_a(w)
    code += stl_a_arn(0, 2)

# --- Loop back ---
code += b_unc(main_loop_addr)

# ---- Sanity check ----
print(f"Code length: {len(code)} words ({len(code)*2} bytes)")
print(f"Main loop entry: 0x{main_loop_addr:04x}")
print(f"Code spans 0x7120 - 0x{0x7120 + len(code) - 1:04x}")

# ---- Patch PROM0.bin ----
with open(INPUT, 'rb') as f:
    prom_bytes = bytearray(f.read())

```

```

print(f"Input PROM0 size: {len(prom_bytes)} bytes ({len(prom_bytes)//2} words)")

# 0x7120 → byte offset 0x240 from PROM0 start (= 2 × (0x7120 - 0x7000))
PATCH_BYTE_OFFSET = 2 * (0x7120 - 0x7000)
print(f"Patch byte offset: 0x{PATCH_BYTE_OFFSET:04x} ({PATCH_BYTE_OFFSET})")

for i, w in enumerate(code):
    off = PATCH_BYTE_OFFSET + 2*i
    prom_bytes[off] = w & 0xFF
    prom_bytes[off+1] = (w >> 8) & 0xFF

with open(OUTPUT, 'wb') as f:
    f.write(prom_bytes)

print(f"Wrote {OUTPUT}")
print()
print("Pour utiliser SANS toucher l'original :")
print(f"  Update QEMU launch dsp-prom0=...PROM0.bin → dsp-prom0={OUTPUT}")
print(f"  L'original /opt/GSM/calypso_dsp.PROM0.bin reste INTACT.")

```

0.3.58 /opt/GSM/qosmo-overlay/scripts/meson-buildoptions.sh

35406 octets, 589 lignes → 589 lignes

```

# This file is generated by meson-buildoptions.py, do not edit!
meson_options_help() {
    printf "%s\n" '  --audio-driv-list=CHOICES Set audio driver list [default] (choices: alsa/c'
    printf "%s\n" '                                reaudio/default/dsound/jack/oss/pa/pipewire/sdl/'
    printf "%s\n" '                                ndio)'
    printf "%s\n" '  --bindir=VALUE           Executable directory [bin]'
    printf "%s\n" '  --block-driv-ro-whitelist=VALUE'
    printf "%s\n" '                                set block driver read-only whitelist (by default'
    printf "%s\n" '                                affects only QEMU, not tools like qemu-img)'
    printf "%s\n" '  --block-driv-rw-whitelist=VALUE'
    printf "%s\n" '                                set block driver read-write whitelist (by default'
    printf "%s\n" '                                affects only QEMU, not tools like qemu-img)'
    printf "%s\n" '  --datadir=VALUE          Data file directory [share]'
    printf "%s\n" '  --disable-coroutine-pool coroutine freelist (better performance)'
    printf "%s\n" '  --disable-debug-info      Enable debug symbols and other information'
    printf "%s\n" '  --disable-hexagon-idef-parser'
    printf "%s\n" '                                use idef-parser to automatically generate TCG'
}

```

```

printf "%s\n" '                                code for the Hexagon frontend'
printf "%s\n" ' --disable-install-blobs    install provided firmware blobs'
printf "%s\n" ' --disable-qom-cast-debug    cast debugging support'
printf "%s\n" ' --disable-relocatable    toggle relocatable install'
printf "%s\n" ' --docdir=VALUE            Base directory for documentation installation'
printf "%s\n" '                                (can be empty) [share/doc]'
printf "%s\n" ' --enable-asan             enable address sanitizer'
printf "%s\n" ' --enable-block-drv-whitelist-in-tools'
printf "%s\n" '                                use block whitelist also in tools instead of only
printf "%s\n" '                                QEMU'
printf "%s\n" ' --enable-cfi              Control-Flow Integrity (CFI)'
printf "%s\n" ' --enable-cfi-debug        Verbose errors in case of CFI violation'
printf "%s\n" ' --enable-debug-graph-lock'
printf "%s\n" '                                graph lock debugging support'
printf "%s\n" ' --enable-debug-mutex      mutex debugging support'
printf "%s\n" ' --enable-debug-remap      syscall buffer debugging support'
printf "%s\n" ' --enable-debug-stack-usage'
printf "%s\n" '                                measure coroutine stack usage'
printf "%s\n" ' --enable-debug-tcg        TCG debugging'
printf "%s\n" ' --enable-fdt[=CHOICE]    Whether and how to find the libfdt library'
printf "%s\n" '                                (choices: auto/disabled/enabled/internal/system)
printf "%s\n" ' --enable-fuzzing          build fuzzing targets'
printf "%s\n" ' --enable-gcov             Enable coverage tracking.'
printf "%s\n" ' --enable-lto              Use link time optimization'
printf "%s\n" ' --enable-malloc=CHOICE   choose memory allocator to use [system] (choices
printf "%s\n" '                                jemalloc/system/tcmalloc)'
printf "%s\n" ' --enable-module-upgrades try to load modules from alternate paths for'
printf "%s\n" '                                upgrades'
printf "%s\n" ' --enable-rng-none        dummy RNG, avoid using /dev/(u)random and'
printf "%s\n" '                                getrandom()'
printf "%s\n" ' --enable-safe-stack      SafeStack Stack Smash Protection (requires'
printf "%s\n" '                                clang/llvm and coroutine backend ucontext)'
printf "%s\n" ' --enable-strict-rust-lints'
printf "%s\n" '                                Enable stricter set of Rust warnings'
printf "%s\n" ' --enable-strip           Strip targets on install'
printf "%s\n" ' --enable-tcg-interpreter TCG with bytecode interpreter (slow)'
printf "%s\n" ' --enable-trace-backends=CHOICES'
printf "%s\n" '                                Set available tracing backends [log] (choices:'
printf "%s\n" '                                dtrace/ftrace/log/nop/simple/syslog/ust)'
printf "%s\n" ' --enable-tsan            enable thread sanitizer'
printf "%s\n" ' --enable-ubsan           enable undefined behaviour sanitizer'
printf "%s\n" ' --firmwarepath=VALUES    search PATH for firmware files [share/qemu-'

```

```

printf "%s\n" '      firmware]'
printf "%s\n" '      --iasl=VALUE      Path to ACPI disassembler'
printf "%s\n" '      --includedir=VALUE      Header file directory [include]'
printf "%s\n" '      --interp-prefix=VALUE    where to find shared libraries etc., use %M for'
printf "%s\n" '      cpu name [/usr/gnemul/qemu-%M]'
printf "%s\n" '      --libdir=VALUE          Library directory [system default]'
printf "%s\n" '      --libexecdir=VALUE      Library executable directory [libexec]'
printf "%s\n" '      --localedir=VALUE       Locale data directory [share/locale]'
printf "%s\n" '      --localstatedir=VALUE    Localstate data directory [/var/local]'
printf "%s\n" '      --mandir=VALUE          Manual page directory [share/man]'
printf "%s\n" '      --prefix=VALUE          Installation prefix [/usr/local]'
printf "%s\n" '      --qemu-ga-distro=VALUE    second path element in qemu-ga registry entries'
printf "%s\n" '      [Linux]'
printf "%s\n" '      --qemu-ga-manufacturer=VALUE'
printf "%s\n" '      "manufacturer" name for qemu-ga registry entries'
printf "%s\n" '      [QEMU]'
printf "%s\n" '      --qemu-ga-version=VALUE  version number for qemu-ga installer'
printf "%s\n" '      --rtsig-map=VALUE        default value of QEMU_RTSIG_MAP [NULL]'
printf "%s\n" '      --smbd=VALUE            Path to smbd for slirp networking'
printf "%s\n" '      --sysconfdir=VALUE       Sysconf data directory [etc]'
printf "%s\n" '      --tls-priority=VALUE     Default TLS protocol/cipher priority string'
printf "%s\n" '      [NORMAL]'
printf "%s\n" '      --with-coroutine=CHOICE  coroutine backend to use (choices:'
printf "%s\n" '      auto/sigaltstack/ucontext/windows)'
printf "%s\n" '      --with-pkgversion=VALUE  use specified string as sub-version of the'
printf "%s\n" '      package'
printf "%s\n" '      --with-suffix=VALUE      Suffix for QEMU data/modules/config directories'
printf "%s\n" '      (can be empty) [qemu]'
printf "%s\n" '      --with-trace-file=VALUE  Trace file prefix for simple backend [trace]'
printf "%s\n" '      --x86-version=CHOICE     tweak required x86_64 architecture version beyond'
printf "%s\n" '      compiler default [1] (choices: 0/1/2/3/4)'
printf "%s\n" ''
printf "%s\n" 'Optional features, enabled with --enable-FEATURE and'
printf "%s\n" 'disabled with --disable-FEATURE, default is enabled if available'
printf "%s\n" '(unless built with --without-default-features):'
printf "%s\n" ''
printf "%s\n" '      af-xdp      AF_XDP network backend support'
printf "%s\n" '      alsa        ALSA sound support'
printf "%s\n" '      attr        attr/xattr support'
printf "%s\n" '      auth-pam    PAM access control'
printf "%s\n" '      avx2        AVX2 optimizations'
printf "%s\n" '      avx512bw    AVX512BW optimizations'

```

```

printf "%s\n" ' blkio          libblkio block device driver'
printf "%s\n" ' bochs          bochs image format support'
printf "%s\n" ' bpf            eBPF support'
printf "%s\n" ' brlapi         brlapi character device driver'
printf "%s\n" ' bzip2          bzip2 support for DMG images'
printf "%s\n" ' canokey        CanoKey support'
printf "%s\n" ' cap-ng         cap_ng support'
printf "%s\n" ' capstone       Whether and how to find the capstone library'
printf "%s\n" ' cloop          cloop image format support'
printf "%s\n" ' cocoa          Cocoa user interface (macOS only)'
printf "%s\n" ' colo-proxy     colo-proxy support'
printf "%s\n" ' coreaudio      CoreAudio sound support'
printf "%s\n" ' crypto-afalg   Linux AF_ALG crypto backend driver'
printf "%s\n" ' curl           CURL block device driver'
printf "%s\n" ' curses         curses UI'
printf "%s\n" ' dbus-display   -display dbus support'
printf "%s\n" ' dmg            dmg image format support'
printf "%s\n" ' docs           Documentations build support'
printf "%s\n" ' dsound         DirectSound sound support'
printf "%s\n" ' fuse           FUSE block device export'
printf "%s\n" ' fuse-lseek     SEEK_HOLE/SEEK_DATA support for FUSE exports'
printf "%s\n" ' gcrypt         libgcrypt cryptography support'
printf "%s\n" ' gettext        Localization of the GTK+ user interface'
printf "%s\n" ' gio            use libgio for D-Bus support'
printf "%s\n" ' glusterfs      Glusterfs block device driver'
printf "%s\n" ' gnutls         GNUTLS cryptography support'
printf "%s\n" ' gtk            GTK+ user interface'
printf "%s\n" ' gtk-clipboard  clipboard support for the gtk UI (EXPERIMENTAL, MAY HANG)
printf "%s\n" ' guest-agent     Build QEMU Guest Agent'
printf "%s\n" ' guest-agent-msi Build MSI package for the QEMU Guest Agent'
printf "%s\n" ' hv-balloon     hv-balloon driver (requires Glib 2.68+ GTree API)'
printf "%s\n" ' hvf            HVF acceleration support'
printf "%s\n" ' iconv          Font glyph conversion support'
printf "%s\n" ' jack           JACK sound support'
printf "%s\n" ' keyring        Linux keyring support'
printf "%s\n" ' kvm            KVM acceleration support'
printf "%s\n" ' l1-dsp         Calypso: couche 1 DSP C54x (mask-ROM TI reelle)'
printf "%s\n" ' l1-grgsm       Calypso: couche 1 gr-gsm (demodulation cote hote)'
printf "%s\n" ' l2tpv3         l2tpv3 network backend support'
printf "%s\n" ' libcbor        libcbor support'
printf "%s\n" ' libdaxctl      libdaxctl support'
printf "%s\n" ' libdw          debuginfo support'

```

```

printf "%s\n" ' libiscsi      libiscsi userspace initiator'
printf "%s\n" ' libkeyutils  Linux keyutils support'
printf "%s\n" ' libnfs      libnfs block device driver'
printf "%s\n" ' libpmem     libpmem support'
printf "%s\n" ' libssh      ssh block device support'
printf "%s\n" ' libudev     Use libudev to enumerate host devices'
printf "%s\n" ' libusb      libusb support for USB passthrough'
printf "%s\n" ' libvduse    build VDUSE Library'
printf "%s\n" ' linux-aio   Linux AIO support'
printf "%s\n" ' linux-io-uring Linux io_uring support'
printf "%s\n" ' lzfs        lzfs support for DMG images'
printf "%s\n" ' lzo         lzo compression support'
printf "%s\n" ' malloc-trim enable libc malloc_trim() for memory optimization'
printf "%s\n" ' membarrier  membarrier system call (for Linux 4.14+ or Windows'
printf "%s\n" ' modules     modules support (non Windows)'
printf "%s\n" ' mpath       Multipath persistent reservation passthrough'
printf "%s\n" ' multiprocess Out of process device emulation support'
printf "%s\n" ' netmap      netmap network backend support'
printf "%s\n" ' nettle      nettle cryptography support'
printf "%s\n" ' numa        libnuma support'
printf "%s\n" ' nvmm        NVMM acceleration support'
printf "%s\n" ' opengl      OpenGL support'
printf "%s\n" ' oss         OSS sound support'
printf "%s\n" ' pa          PulseAudio sound support'
printf "%s\n" ' parallels   parallels image format support'
printf "%s\n" ' pipewire    PipeWire sound support'
printf "%s\n" ' pixman      pixman support'
printf "%s\n" ' plugins     TCG plugins via shared library loading'
printf "%s\n" ' png         PNG support with libpng'
printf "%s\n" ' qatzip      QATzip compression support'
printf "%s\n" ' qcow1       qcow1 image format support'
printf "%s\n" ' qed         qed image format support'
printf "%s\n" ' qga-vss     build QGA VSS support (broken with MinGW)'
printf "%s\n" ' qpl         Query Processing Library support'
printf "%s\n" ' rbd         Ceph block device driver'
printf "%s\n" ' rdma        Enable RDMA-based migration'
printf "%s\n" ' replication replication support'
printf "%s\n" ' rust        Rust support'
printf "%s\n" ' rutabaga-gfx rutabaga_gfx support'
printf "%s\n" ' sdl         SDL user interface'
printf "%s\n" ' sdl-image   SDL Image support for icons'
printf "%s\n" ' seccomp     seccomp support'

```

```

printf "%s\n" ' selinux      SELinux support in qemu-nbd'
printf "%s\n" ' slirp       libslirp user mode network backend support'
printf "%s\n" ' slirp-smbd   use smbd (at path --smbd=*) in slirp networking'
printf "%s\n" ' smartcard    CA smartcard emulation support'
printf "%s\n" ' snappy       snappy compression support'
printf "%s\n" ' sndio        sndio sound support'
printf "%s\n" ' sparse       sparse checker'
printf "%s\n" ' spice        Spice server support'
printf "%s\n" ' spice-protocol  Spice protocol support'
printf "%s\n" ' stack-protector compiler-provided stack protection'
printf "%s\n" ' tcg          TCG support'
printf "%s\n" ' tools        build support utilities that come with QEMU'
printf "%s\n" ' tpm          TPM support'
printf "%s\n" ' u2f          U2F emulation support'
printf "%s\n" ' uadk         UADK Library support'
printf "%s\n" ' usb-redir    libusbredir support'
printf "%s\n" ' vde          vde network backend support'
printf "%s\n" ' vdi          vdi image format support'
printf "%s\n" ' vduse-blk-export'
printf "%s\n" '              VDUSE block export support'
printf "%s\n" ' vfio-user-server'
printf "%s\n" '              vfio-user server support'
printf "%s\n" ' vhdx         vhdx image format support'
printf "%s\n" ' vhost-crypto vhost-user crypto backend support'
printf "%s\n" ' vhost-kernel vhost kernel backend support'
printf "%s\n" ' vhost-net    vhost-net kernel acceleration support'
printf "%s\n" ' vhost-user    vhost-user backend support'
printf "%s\n" ' vhost-user-blk-server'
printf "%s\n" '              build vhost-user-blk server'
printf "%s\n" ' vhost-vdpa    vhost-vdpa kernel backend support'
printf "%s\n" ' virglrenderer virgl rendering support'
printf "%s\n" ' virtfs       virtio-9p support'
printf "%s\n" ' vmdk         vmdk image format support'
printf "%s\n" ' vmnet        vmnet.framework network backend support'
printf "%s\n" ' vnc          VNC server'
printf "%s\n" ' vnc-jpeg     JPEG lossy compression for VNC server'
printf "%s\n" ' vnc-sasl     SASL authentication for VNC server'
printf "%s\n" ' vpc          vpc image format support'
printf "%s\n" ' vite         vite support for the gtk UI'
printf "%s\n" ' vvfat        vvfat image format support'
printf "%s\n" ' werror       Treat warnings as errors'
printf "%s\n" ' whpx         WHPX acceleration support'

```

```

printf "%s\n" ' xen Xen backend support'
printf "%s\n" ' xen-pci-passthrough'
printf "%s\n" ' Xen PCI passthrough support'
printf "%s\n" ' xkbcommon xkbcommon support'
printf "%s\n" ' zstd zstd compression support'
}
_meson_option_parse() {
case $1 in
--enable-af-xdp) printf "%s" -Daf_xdp=enabled ;;
--disable-af-xdp) printf "%s" -Daf_xdp=disabled ;;
--enable-alsa) printf "%s" -Dalsa=enabled ;;
--disable-alsa) printf "%s" -Dalsa=disabled ;;
--enable-asan) printf "%s" -Dasan=true ;;
--disable-asan) printf "%s" -Dasan=false ;;
--enable-attr) printf "%s" -Dattrib=enabled ;;
--disable-attr) printf "%s" -Dattrib=disabled ;;
--audio-driv-list=*) quote_sh "-Daudio_driv_list=$2" ;;
--enable-auth-pam) printf "%s" -Dauth_pam=enabled ;;
--disable-auth-pam) printf "%s" -Dauth_pam=disabled ;;
--enable-avx2) printf "%s" -Davx2=enabled ;;
--disable-avx2) printf "%s" -Davx2=disabled ;;
--enable-avx512bw) printf "%s" -Davx512bw=enabled ;;
--disable-avx512bw) printf "%s" -Davx512bw=disabled ;;
--enable-gcov) printf "%s" -Dgcov=true ;;
--disable-gcov) printf "%s" -Dgcov=false ;;
--enable-lto) printf "%s" -Dlto=true ;;
--disable-lto) printf "%s" -Dlto=false ;;
--bindir=*) quote_sh "-Dbindir=$2" ;;
--enable-blkio) printf "%s" -Dblkio=enabled ;;
--disable-blkio) printf "%s" -Dblkio=disabled ;;
--block-driv-ro-whitelist=*) quote_sh "-Dblock_driv_ro_whitelist=$2" ;;
--block-driv-rw-whitelist=*) quote_sh "-Dblock_driv_rw_whitelist=$2" ;;
--enable-block-driv-whitelist-in-tools) printf "%s" -Dblock_driv_whitelist_in_tools=true ;
--disable-block-driv-whitelist-in-tools) printf "%s" -Dblock_driv_whitelist_in_tools=false ;
--enable-bochs) printf "%s" -Dbochs=enabled ;;
--disable-bochs) printf "%s" -Dbochs=disabled ;;
--enable-bpf) printf "%s" -Dbpf=enabled ;;
--disable-bpf) printf "%s" -Dbpf=disabled ;;
--enable-brlapi) printf "%s" -Dbrlapi=enabled ;;
--disable-brlapi) printf "%s" -Dbrlapi=disabled ;;
--enable-bzip2) printf "%s" -Dbzip2=enabled ;;
--disable-bzip2) printf "%s" -Dbzip2=disabled ;;

```

```

--enable-canokey) printf "%s" -Dcanokey=enabled ;;
--disable-canokey) printf "%s" -Dcanokey=disabled ;;
--enable-cap-ng) printf "%s" -Dcap_ng=enabled ;;
--disable-cap-ng) printf "%s" -Dcap_ng=disabled ;;
--enable-capstone) printf "%s" -Dcapstone=enabled ;;
--disable-capstone) printf "%s" -Dcapstone=disabled ;;
--enable-cfi) printf "%s" -Dcfi=true ;;
--disable-cfi) printf "%s" -Dcfi=false ;;
--enable-cfi-debug) printf "%s" -Dcfi_debug=true ;;
--disable-cfi-debug) printf "%s" -Dcfi_debug=false ;;
--enable-cloop) printf "%s" -Dcloop=enabled ;;
--disable-cloop) printf "%s" -Dcloop=disabled ;;
--enable-cocoa) printf "%s" -Dcocoa=enabled ;;
--disable-cocoa) printf "%s" -Dcocoa=disabled ;;
--enable-colo-proxy) printf "%s" -Dcolo_proxy=enabled ;;
--disable-colo-proxy) printf "%s" -Dcolo_proxy=disabled ;;
--enable-coreaudio) printf "%s" -Dcoreaudio=enabled ;;
--disable-coreaudio) printf "%s" -Dcoreaudio=disabled ;;
--with-coroutine=*) quote_sh "-Dcoroutine_backend=$2" ;;
--enable-coroutine-pool) printf "%s" -Dcoroutine_pool=true ;;
--disable-coroutine-pool) printf "%s" -Dcoroutine_pool=false ;;
--enable-crypto-afalg) printf "%s" -Dcrypto_afalg=enabled ;;
--disable-crypto-afalg) printf "%s" -Dcrypto_afalg=disabled ;;
--enable-curl) printf "%s" -Dcurl=enabled ;;
--disable-curl) printf "%s" -Dcurl=disabled ;;
--enable-curses) printf "%s" -Dcurses=enabled ;;
--disable-curses) printf "%s" -Dcurses=disabled ;;
--datadir=*) quote_sh "-Ddatadir=$2" ;;
--enable-dbus-display) printf "%s" -Ddbus_display=enabled ;;
--disable-dbus-display) printf "%s" -Ddbus_display=disabled ;;
--enable-debug-info) printf "%s" -Ddebug=true ;;
--disable-debug-info) printf "%s" -Ddebug=false ;;
--enable-debug-graph-lock) printf "%s" -Ddebug_graph_lock=true ;;
--disable-debug-graph-lock) printf "%s" -Ddebug_graph_lock=false ;;
--enable-debug-mutex) printf "%s" -Ddebug_mutex=true ;;
--disable-debug-mutex) printf "%s" -Ddebug_mutex=false ;;
--enable-debug-remap) printf "%s" -Ddebug_remap=true ;;
--disable-debug-remap) printf "%s" -Ddebug_remap=false ;;
--enable-debug-stack-usage) printf "%s" -Ddebug_stack_usage=true ;;
--disable-debug-stack-usage) printf "%s" -Ddebug_stack_usage=false ;;
--enable-debug-tcg) printf "%s" -Ddebug_tcg=true ;;
--disable-debug-tcg) printf "%s" -Ddebug_tcg=false ;;

```

```

--enable-dmg) printf "%s" -Ddmg=enabled ;;
--disable-dmg) printf "%s" -Ddmg=disabled ;;
--docdir=*) quote_sh "-Ddocdir=$2" ;;
--enable-docs) printf "%s" -Ddocs=enabled ;;
--disable-docs) printf "%s" -Ddocs=disabled ;;
--enable-dsound) printf "%s" -Ddsound=enabled ;;
--disable-dsound) printf "%s" -Ddsound=disabled ;;
--enable-fdt) printf "%s" -Dfdt=enabled ;;
--disable-fdt) printf "%s" -Dfdt=disabled ;;
--enable-fdt=*) quote_sh "-Dfdt=$2" ;;
--enable-fuse) printf "%s" -Dfuse=enabled ;;
--disable-fuse) printf "%s" -Dfuse=disabled ;;
--enable-fuse-lseek) printf "%s" -Dfuse_lseek=enabled ;;
--disable-fuse-lseek) printf "%s" -Dfuse_lseek=disabled ;;
--enable-fuzzing) printf "%s" -Dfuzzing=true ;;
--disable-fuzzing) printf "%s" -Dfuzzing=false ;;
--enable-gcrypt) printf "%s" -Dgcrypt=enabled ;;
--disable-gcrypt) printf "%s" -Dgcrypt=disabled ;;
--enable-gettext) printf "%s" -Dgettext=enabled ;;
--disable-gettext) printf "%s" -Dgettext=disabled ;;
--enable-gio) printf "%s" -Dgio=enabled ;;
--disable-gio) printf "%s" -Dgio=disabled ;;
--enable-glusterfs) printf "%s" -Dglusterfs=enabled ;;
--disable-glusterfs) printf "%s" -Dglusterfs=disabled ;;
--enable-gnutls) printf "%s" -Dgnutls=enabled ;;
--disable-gnutls) printf "%s" -Dgnutls=disabled ;;
--enable-gtk) printf "%s" -Dgtk=enabled ;;
--disable-gtk) printf "%s" -Dgtk=disabled ;;
--enable-gtk-clipboard) printf "%s" -Dgtk_clipboard=enabled ;;
--disable-gtk-clipboard) printf "%s" -Dgtk_clipboard=disabled ;;
--enable-guest-agent) printf "%s" -Dguest_agent=enabled ;;
--disable-guest-agent) printf "%s" -Dguest_agent=disabled ;;
--enable-guest-agent-msi) printf "%s" -Dguest_agent_msi=enabled ;;
--disable-guest-agent-msi) printf "%s" -Dguest_agent_msi=disabled ;;
--enable-hexagon-idef-parser) printf "%s" -Dhexagon_idef_parser=true ;;
--disable-hexagon-idef-parser) printf "%s" -Dhexagon_idef_parser=false ;;
--enable-hv-balloon) printf "%s" -Dhv_balloon=enabled ;;
--disable-hv-balloon) printf "%s" -Dhv_balloon=disabled ;;
--enable-hvf) printf "%s" -Dhvf=enabled ;;
--disable-hvf) printf "%s" -Dhvf=disabled ;;
--iasl=*) quote_sh "-Diasl=$2" ;;
--enable-iconv) printf "%s" -Diconv=enabled ;;

```

```

--disable-iconv) printf "%s" -Diconv=disabled ;;
--includedir=*) quote_sh "-Dincludedir=$2" ;;
--enable-install-blobs) printf "%s" -Dinstall_blobs=true ;;
--disable-install-blobs) printf "%s" -Dinstall_blobs=false ;;
--interp-prefix=*) quote_sh "-Dinterp_prefix=$2" ;;
--enable-jack) printf "%s" -Djack=enabled ;;
--disable-jack) printf "%s" -Djack=disabled ;;
--enable-keyring) printf "%s" -Dkeyring=enabled ;;
--disable-keyring) printf "%s" -Dkeyring=disabled ;;
--enable-kvm) printf "%s" -Dkvm=enabled ;;
--disable-kvm) printf "%s" -Dkvm=disabled ;;
--enable-l1-dsp) printf "%s" -Dl1_dsp=enabled ;;
--disable-l1-dsp) printf "%s" -Dl1_dsp=disabled ;;
--enable-l1-grgsm) printf "%s" -Dl1_grgsm=enabled ;;
--disable-l1-grgsm) printf "%s" -Dl1_grgsm=disabled ;;
--enable-l2tpv3) printf "%s" -Dl2tpv3=enabled ;;
--disable-l2tpv3) printf "%s" -Dl2tpv3=disabled ;;
--enable-libcbor) printf "%s" -Dlibcbor=enabled ;;
--disable-libcbor) printf "%s" -Dlibcbor=disabled ;;
--enable-libdaxctl) printf "%s" -Dlibdaxctl=enabled ;;
--disable-libdaxctl) printf "%s" -Dlibdaxctl=disabled ;;
--libdir=*) quote_sh "-Dlibdir=$2" ;;
--enable-libdw) printf "%s" -Dlibdw=enabled ;;
--disable-libdw) printf "%s" -Dlibdw=disabled ;;
--libexecdir=*) quote_sh "-Dlibexecdir=$2" ;;
--enable-libiscsi) printf "%s" -Dlibiscsi=enabled ;;
--disable-libiscsi) printf "%s" -Dlibiscsi=disabled ;;
--enable-libkeyutils) printf "%s" -Dlibkeyutils=enabled ;;
--disable-libkeyutils) printf "%s" -Dlibkeyutils=disabled ;;
--enable-libnfs) printf "%s" -Dlibnfs=enabled ;;
--disable-libnfs) printf "%s" -Dlibnfs=disabled ;;
--enable-libpmem) printf "%s" -Dlibpmem=enabled ;;
--disable-libpmem) printf "%s" -Dlibpmem=disabled ;;
--enable-libssh) printf "%s" -Dlibssh=enabled ;;
--disable-libssh) printf "%s" -Dlibssh=disabled ;;
--enable-libudev) printf "%s" -Dlibudev=enabled ;;
--disable-libudev) printf "%s" -Dlibudev=disabled ;;
--enable-libusb) printf "%s" -Dlibusb=enabled ;;
--disable-libusb) printf "%s" -Dlibusb=disabled ;;
--enable-libvduse) printf "%s" -Dlibvduse=enabled ;;
--disable-libvduse) printf "%s" -Dlibvduse=disabled ;;
--enable-linux-aio) printf "%s" -Dlinux_aio=enabled ;;

```

```

--disable-linux-aio) printf "%s" -Dlinux_aio=disabled ;;
--enable-linux-io-uring) printf "%s" -Dlinux_io_uring=enabled ;;
--disable-linux-io-uring) printf "%s" -Dlinux_io_uring=disabled ;;
--localedir=*) quote_sh "-Dlocaledir=$2" ;;
--localstatedir=*) quote_sh "-Dlocalstatedir=$2" ;;
--enable-lzfs) printf "%s" -Dlzfs=enabled ;;
--disable-lzfs) printf "%s" -Dlzfs=disabled ;;
--enable-lzo) printf "%s" -Dlzo=enabled ;;
--disable-lzo) printf "%s" -Dlzo=disabled ;;
--enable-malloc=*) quote_sh "-Dmalloc=$2" ;;
--enable-malloc-trim) printf "%s" -Dmalloc_trim=enabled ;;
--disable-malloc-trim) printf "%s" -Dmalloc_trim=disabled ;;
--mandir=*) quote_sh "-Dmandir=$2" ;;
--enable-membarrier) printf "%s" -Dmembarrier=enabled ;;
--disable-membarrier) printf "%s" -Dmembarrier=disabled ;;
--enable-module-upgrades) printf "%s" -Dmodule_upgrades=true ;;
--disable-module-upgrades) printf "%s" -Dmodule_upgrades=false ;;
--enable-modules) printf "%s" -Dmodules=enabled ;;
--disable-modules) printf "%s" -Dmodules=disabled ;;
--enable-mpath) printf "%s" -Dmpath=enabled ;;
--disable-mpath) printf "%s" -Dmpath=disabled ;;
--enable-multiprocess) printf "%s" -Dmultiprocess=enabled ;;
--disable-multiprocess) printf "%s" -Dmultiprocess=disabled ;;
--enable-netmap) printf "%s" -Dnetmap=enabled ;;
--disable-netmap) printf "%s" -Dnetmap=disabled ;;
--enable-nettle) printf "%s" -Dnettle=enabled ;;
--disable-nettle) printf "%s" -Dnettle=disabled ;;
--enable-numa) printf "%s" -Dnuma=enabled ;;
--disable-numa) printf "%s" -Dnuma=disabled ;;
--enable-nvmm) printf "%s" -Dnvmm=enabled ;;
--disable-nvmm) printf "%s" -Dnvmm=disabled ;;
--enable-opengl) printf "%s" -Dopengl=enabled ;;
--disable-opengl) printf "%s" -Dopengl=disabled ;;
--enable-oss) printf "%s" -Doss=enabled ;;
--disable-oss) printf "%s" -Doss=disabled ;;
--enable-pa) printf "%s" -Dpa=enabled ;;
--disable-pa) printf "%s" -Dpa=disabled ;;
--enable-parallels) printf "%s" -Dparallels=enabled ;;
--disable-parallels) printf "%s" -Dparallels=disabled ;;
--enable-pipewire) printf "%s" -Dpipewire=enabled ;;
--disable-pipewire) printf "%s" -Dpipewire=disabled ;;
--enable-pixman) printf "%s" -Dpixman=enabled ;;

```

```

--disable-pixman) printf "%s" -Dpixman=disabled ;;
--with-pkgversion=*) quote_sh "-Dpkgversion=$2" ;;
--enable-plugins) printf "%s" -Dplugins=true ;;
--disable-plugins) printf "%s" -Dplugins=false ;;
--enable-png) printf "%s" -Dpng=enabled ;;
--disable-png) printf "%s" -Dpng=disabled ;;
--prefix=*) quote_sh "-Dprefix=$2" ;;
--enable-qatzip) printf "%s" -Dqatzip=enabled ;;
--disable-qatzip) printf "%s" -Dqatzip=disabled ;;
--enable-qcow1) printf "%s" -Dqcow1=enabled ;;
--disable-qcow1) printf "%s" -Dqcow1=disabled ;;
--enable-qed) printf "%s" -Dqed=enabled ;;
--disable-qed) printf "%s" -Dqed=disabled ;;
--firmwarepath=*) quote_sh "-Dqemu_firmwarepath=$(meson_option_build_array $2)" ;;
--qemu-ga-distro=*) quote_sh "-Dqemu_ga_distro=$2" ;;
--qemu-ga-manufacturer=*) quote_sh "-Dqemu_ga_manufacturer=$2" ;;
--qemu-ga-version=*) quote_sh "-Dqemu_ga_version=$2" ;;
--with-suffix=*) quote_sh "-Dqemu_suffix=$2" ;;
--enable-qga-vss) printf "%s" -Dqga_vss=enabled ;;
--disable-qga-vss) printf "%s" -Dqga_vss=disabled ;;
--enable-qom-cast-debug) printf "%s" -Dqom_cast_debug=true ;;
--disable-qom-cast-debug) printf "%s" -Dqom_cast_debug=false ;;
--enable-qpl) printf "%s" -Dqpl=enabled ;;
--disable-qpl) printf "%s" -Dqpl=disabled ;;
--enable-rbd) printf "%s" -Drbd=enabled ;;
--disable-rbd) printf "%s" -Drbd=disabled ;;
--enable-rdma) printf "%s" -Drdma=enabled ;;
--disable-rdma) printf "%s" -Drdma=disabled ;;
--enable-relocatable) printf "%s" -Drelocatable=true ;;
--disable-relocatable) printf "%s" -Drelocatable=false ;;
--enable-replication) printf "%s" -Dreplication=enabled ;;
--disable-replication) printf "%s" -Dreplication=disabled ;;
--enable-rng-none) printf "%s" -Drng_none=true ;;
--disable-rng-none) printf "%s" -Drng_none=false ;;
--rtsig-map=*) quote_sh "-Drtsig_map=$2" ;;
--enable-rust) printf "%s" -Drust=enabled ;;
--disable-rust) printf "%s" -Drust=disabled ;;
--enable-rutabaga-gfx) printf "%s" -Drutabaga_gfx=enabled ;;
--disable-rutabaga-gfx) printf "%s" -Drutabaga_gfx=disabled ;;
--enable-safe-stack) printf "%s" -Dsafe_stack=true ;;
--disable-safe-stack) printf "%s" -Dsafe_stack=false ;;
--enable-sdl) printf "%s" -Dsdl=enabled ;;

```

```

--disable-sdl) printf "%s" -Dsdl=disabled ;;
--enable-sdl-image) printf "%s" -Dsdl_image=enabled ;;
--disable-sdl-image) printf "%s" -Dsdl_image=disabled ;;
--enable-seccomp) printf "%s" -Dseccomp=enabled ;;
--disable-seccomp) printf "%s" -Dseccomp=disabled ;;
--enable-selinux) printf "%s" -Dselinux=enabled ;;
--disable-selinux) printf "%s" -Dselinux=disabled ;;
--enable-slirp) printf "%s" -Dslirp=enabled ;;
--disable-slirp) printf "%s" -Dslirp=disabled ;;
--enable-slirp-smbd) printf "%s" -Dslirp_smbd=enabled ;;
--disable-slirp-smbd) printf "%s" -Dslirp_smbd=disabled ;;
--enable-smartcard) printf "%s" -Dsmartcard=enabled ;;
--disable-smartcard) printf "%s" -Dsmartcard=disabled ;;
--smbd=*) quote_sh "-Dsmbd=$2" ;;
--enable-snappy) printf "%s" -Dsnappy=enabled ;;
--disable-snappy) printf "%s" -Dsnappy=disabled ;;
--enable-sndio) printf "%s" -Dsndio=enabled ;;
--disable-sndio) printf "%s" -Dsndio=disabled ;;
--enable-sparse) printf "%s" -Dsparse=enabled ;;
--disable-sparse) printf "%s" -Dsparse=disabled ;;
--enable-spice) printf "%s" -Dspice=enabled ;;
--disable-spice) printf "%s" -Dspice=disabled ;;
--enable-spice-protocol) printf "%s" -Dspice_protocol=enabled ;;
--disable-spice-protocol) printf "%s" -Dspice_protocol=disabled ;;
--enable-stack-protector) printf "%s" -Dstack_protector=enabled ;;
--disable-stack-protector) printf "%s" -Dstack_protector=disabled ;;
--enable-strict-rust-lints) printf "%s" -Dstrict_rust_lints=true ;;
--disable-strict-rust-lints) printf "%s" -Dstrict_rust_lints=false ;;
--enable-strip) printf "%s" -Dstrip=true ;;
--disable-strip) printf "%s" -Dstrip=false ;;
--sysconfdir=*) quote_sh "-Dsysconfdir=$2" ;;
--enable-tcg) printf "%s" -Dtcg=enabled ;;
--disable-tcg) printf "%s" -Dtcg=disabled ;;
--enable-tcg-interpreter) printf "%s" -Dtcg_interpreter=true ;;
--disable-tcg-interpreter) printf "%s" -Dtcg_interpreter=false ;;
--tls-priority=*) quote_sh "-Dtls_priority=$2" ;;
--enable-tools) printf "%s" -Dtools=enabled ;;
--disable-tools) printf "%s" -Dtools=disabled ;;
--enable-tpm) printf "%s" -Dtpm=enabled ;;
--disable-tpm) printf "%s" -Dtpm=disabled ;;
--enable-trace-backends=*) quote_sh "-Dtrace_backends=$2" ;;
--with-trace-file=*) quote_sh "-Dtrace_file=$2" ;;

```

```

--enable-tsan) printf "%s" -Dtsan=true ;;
--disable-tsan) printf "%s" -Dtsan=false ;;
--enable-u2f) printf "%s" -Du2f=enabled ;;
--disable-u2f) printf "%s" -Du2f=disabled ;;
--enable-uadk) printf "%s" -Duadk=enabled ;;
--disable-uadk) printf "%s" -Duadk=disabled ;;
--enable-ubsan) printf "%s" -Dubsan=true ;;
--disable-ubsan) printf "%s" -Dubsan=false ;;
--enable-usb-redir) printf "%s" -Dusb_redir=enabled ;;
--disable-usb-redir) printf "%s" -Dusb_redir=disabled ;;
--enable-vde) printf "%s" -Dvde=enabled ;;
--disable-vde) printf "%s" -Dvde=disabled ;;
--enable-vdi) printf "%s" -Dvdi=enabled ;;
--disable-vdi) printf "%s" -Dvdi=disabled ;;
--enable-vduse-blk-export) printf "%s" -Dvduse_blk_export=enabled ;;
--disable-vduse-blk-export) printf "%s" -Dvduse_blk_export=disabled ;;
--enable-vfio-user-server) printf "%s" -Dvfio_user_server=enabled ;;
--disable-vfio-user-server) printf "%s" -Dvfio_user_server=disabled ;;
--enable-vhdx) printf "%s" -Dvhdx=enabled ;;
--disable-vhdx) printf "%s" -Dvhdx=disabled ;;
--enable-vhost-crypto) printf "%s" -Dvhost_crypto=enabled ;;
--disable-vhost-crypto) printf "%s" -Dvhost_crypto=disabled ;;
--enable-vhost-kernel) printf "%s" -Dvhost_kernel=enabled ;;
--disable-vhost-kernel) printf "%s" -Dvhost_kernel=disabled ;;
--enable-vhost-net) printf "%s" -Dvhost_net=enabled ;;
--disable-vhost-net) printf "%s" -Dvhost_net=disabled ;;
--enable-vhost-user) printf "%s" -Dvhost_user=enabled ;;
--disable-vhost-user) printf "%s" -Dvhost_user=disabled ;;
--enable-vhost-user-blk-server) printf "%s" -Dvhost_user_blk_server=enabled ;;
--disable-vhost-user-blk-server) printf "%s" -Dvhost_user_blk_server=disabled ;;
--enable-vhost-udpa) printf "%s" -Dvhost_udpa=enabled ;;
--disable-vhost-udpa) printf "%s" -Dvhost_udpa=disabled ;;
--enable-virglrenderer) printf "%s" -Dvirglrenderer=enabled ;;
--disable-virglrenderer) printf "%s" -Dvirglrenderer=disabled ;;
--enable-virtfs) printf "%s" -Dvirtfs=enabled ;;
--disable-virtfs) printf "%s" -Dvirtfs=disabled ;;
--enable-vmx) printf "%s" -Dvmx=enabled ;;
--disable-vmx) printf "%s" -Dvmx=disabled ;;
--enable-vmnet) printf "%s" -Dvmnet=enabled ;;
--disable-vmnet) printf "%s" -Dvmnet=disabled ;;
--enable-vnc) printf "%s" -Dvnc=enabled ;;
--disable-vnc) printf "%s" -Dvnc=disabled ;;

```

```

--enable-vnc-jpeg) printf "%s" -Dvnc_jpeg=enabled ;;
--disable-vnc-jpeg) printf "%s" -Dvnc_jpeg=disabled ;;
--enable-vnc-sasl) printf "%s" -Dvnc_sasl=enabled ;;
--disable-vnc-sasl) printf "%s" -Dvnc_sasl=disabled ;;
--enable-vpc) printf "%s" -Dvpc=enabled ;;
--disable-vpc) printf "%s" -Dvpc=disabled ;;
--enable-vte) printf "%s" -Dvte=enabled ;;
--disable-vte) printf "%s" -Dvte=disabled ;;
--enable-vvfat) printf "%s" -Dvvfat=enabled ;;
--disable-vvfat) printf "%s" -Dvvfat=disabled ;;
--enable-werror) printf "%s" -Dwerror=true ;;
--disable-werror) printf "%s" -Dwerror=false ;;
--enable-whpx) printf "%s" -Dwhpx=enabled ;;
--disable-whpx) printf "%s" -Dwhpx=disabled ;;
--x86-version=*) quote_sh "-Dx86_version=$2" ;;
--enable-xen) printf "%s" -Dxen=enabled ;;
--disable-xen) printf "%s" -Dxen=disabled ;;
--enable-xen-pci-passthrough) printf "%s" -Dxen_pci_passthrough=enabled ;;
--disable-xen-pci-passthrough) printf "%s" -Dxen_pci_passthrough=disabled ;;
--enable-xkbcommon) printf "%s" -Dxkbcommon=enabled ;;
--disable-xkbcommon) printf "%s" -Dxkbcommon=disabled ;;
--enable-zstd) printf "%s" -Dzstd=enabled ;;
--disable-zstd) printf "%s" -Dzstd=disabled ;;
*) return 1 ;;
esac
}

```

0.3.59 /opt/GSM/qosmo-overlay/scripts/populate-si.sh

3638 octets, 96 lignes → 94 lignes (1 groupes compactés)

```

#!/bin/bash
# populate-si.sh - XXX TRANSITIONAL STUB (étape 2b, disponible).
#
# Écrit les 5 SI hex blobs RSL-extracted (osmo-bsc → osmo-bts trace 2026-04-30)
# dans /dev/shm/calypso_si.bin per doc/MMAP_SI_FORMAT.md v1.
#
# Permet de valider l'interface mmap (étape 2a) sans dépendance sur le RSL
# parser (étape 3). À supprimer dès que scripts/rsl_si_tap.py est opérationnel
# en steady-state.
#

```



```

    s = bytearray(32)
    s[0] = si_type
    s[1] = SLOT_VALID
    s[2] = len(blob)
    s[3] = 0 # padding
    s[4:4+23] = blob
    # 27..31 padding zero (already)
    return bytes(s)

buf = bytearray(176)
# Header
buf[0:4] = b"CSI1"
buf[4] = 0x01 # version
buf[5] = 0x05 # slot_count
buf[6:8] = struct.pack("<H", 0) # last_update_fn = 0 (stub static, no live FN)
# bytes 8..15 reserved zero (already)

# Slots in fixed order : SI1, SI2, SI3, SI4, SI13
buf[16:48] = make_slot(0x01, SI1)
buf[48:80] = make_slot(0x02, SI2)
buf[80:112] = make_slot(0x03, SI3)
buf[112:144] = make_slot(0x04, SI4)
buf[144:176] = make_slot(0x0d, SI13)

with open(OUTPUT, "wb") as f:
    f.write(buf)

# Verify
with open(OUTPUT, "rb") as f:
    chk = f.read()
assert chk == bytes(buf), "readback mismatch"
print(f"OK : wrote {len(buf)} bytes to {OUTPUT}")
print(f"    magic={chk[0:4]!r} version={chk[4]} slot_count={chk[5]}")
for slot_idx, name in enumerate(["SI1", "SI2", "SI3", "SI4", "SI13"]):
    off = 16 + slot_idx * 32
    si_type = chk[off]
    flags = chk[off+1]
    blob_len = chk[off+2]
    blob = chk[off+4:off+4+23]
    print(f"    slot[{slot_idx}] {name}: type=0x{si_type:02x} flags=0x{flags:02x} "
          f"len={blob_len} blob[0:4]={blob[:4].hex()} blob[-4:]={blob[-4:].hex()}")
PYEOF

```

0.3.60 /opt/GSM/qosmo-overlay/scripts/replay_pm_payload.py

6731 octets, 211 lignes → 211 lignes

```
#!/usr/bin/env python3
"""
replay_pm_payload.py - Reproduit le payload L1CTL PM scan observe dans
les logs (hdlc_send/recv sur DLCI 5).

Encode des L1CTL_PM_REQ + decode/print des L1CTL_PM_CONF.

Trois modes :
    1. --dump      : imprime les bytes attendus (verification format)
    2. --to-socket /tmp/osmocom_l2 : push vers un socket L1CTL existant
    3. --hdlc-pty  /dev/pts/X   : enveloppe en HDLC DLCI 5 et envoie sur PTY

Format L1CTL observe (32-bit big-endian) :
    hdlc payload = CMD(4B) ARG(4B) [PAYLOAD...]

Commandes vues dans le log :
    0x08 PM_REQ   : ARG=band, payload = 2x uint16 BE (arfcn_lo, arfcn_hi)
    0x09 PM_CONF  : ARG=batch_id, payload = N x uint32 BE arfcn list
    0x0d          : sub-cmd ack/next (suit chaque PM_CONF)

Usage :
    ./replay_pm_payload.py --dump
    ./replay_pm_payload.py --to-socket /tmp/osmocom_l2
    ./replay_pm_payload.py --hdlc-pty /dev/pts/1
"""

import argparse
import socket
import struct
import sys

# L1CTL command IDs (deduits du log + osmocom-bb l1ctl_proto.h)
CMD_PM_REQ   = 0x08
CMD_PM_CONF  = 0x09
CMD_DONE     = 0x0d

# GSM band IDs
BAND_GSM900 = 0x00
```

```

BAND_DCS1800 = 0x01

def encode_pm_req(band: int, arfcn_lo: int, arfcn_hi: int) -> bytes:
    """
    Encode L1CTL_PM_REQ comme observe dans le log :
        08 00 00 00 BB 00 00 00 LL LL HH HH

    band = BB (0x00 GSM900, 0x01 DCS1800)
    arfcn_lo = LL LL (16-bit BE)
    arfcn_hi = HH HH (16-bit BE)
    """
    return struct.pack(
        ">II HH",
        CMD_PM_REQ,          # 32-bit BE cmd
        band,                # 32-bit BE band
        arfcn_lo,            # 16-bit BE
        arfcn_hi,            # 16-bit BE
    )

def encode_done(sub: int = 1) -> bytes:
    """Encode L1CTL_DONE (0x0d) : 0d 00 00 00 SS 00 00 00"""
    return struct.pack(">II", CMD_DONE, sub)

def decode_pm_conf(b: bytes) -> tuple[int, int, list[int]]:
    """
    Decode L1CTL_PM_CONF :
        09 00 00 00 SS 00 00 00 [arfcn 32-bit BE]*

    Returns (cmd, sub, arfcn_list)
    """
    if len(b) < 8:
        raise ValueError("PM_CONF too short")
    cmd, sub = struct.unpack(">II", b[:8])
    if cmd != CMD_PM_CONF:
        raise ValueError(f"not a PM_CONF (cmd=0x{cmd:02x})")
    n = (len(b) - 8) // 4
    arfcns = list(struct.unpack(f">{n}I", b[8 : 8 + 4 * n]))
    return cmd, sub, arfcns

```

```

# ----- HDLC framing (DLCI 5) -----
#
# osmocon hdlc protocol : frame_start(0x7E) [escaped payload] frame_end(0x7E)
# Escape : 0x7D 0x5D = literal 0x7D ; 0x7D 0x5E = literal 0x7E
#
# DLCI byte prepended to payload, then HDLC framing.
HDLC_FLAG      = 0x7E
HDLC_ESC       = 0x7D
HDLC_ESC_XOR   = 0x20
DLCI_SERCOMM   = 5

def hdlc_escape(b: bytes) -> bytes:
    out = bytearray()
    for x in b:
        if x in (HDLC_FLAG, HDLC_ESC):
            out.append(HDLC_ESC)
            out.append(x ^ HDLC_ESC_XOR)
        else:
            out.append(x)
    return bytes(out)

def hdlc_frame(dlci: int, payload: bytes) -> bytes:
    """Wrap payload in HDLC framing for given DLCI."""
    body = bytes([dlci]) + payload
    return bytes([HDLC_FLAG]) + hdlc_escape(body) + bytes([HDLC_FLAG])

# ----- Scenarios -----

def build_observed_sequence() -> list[tuple[str, bytes]]:
    """Reproduit la sequence exacte du log."""
    seq = []
    # DCS 1800 scan : ARFCN 940..954
    seq.append(("PM_REQ DCS 940-954", encode_pm_req(BAND_DCS1800, 940, 954)))
    # CONF expected : list of arfcn 940..954
    arfcn_list = list(range(940, 955))
    conf = struct.pack(">II", CMD_PM_CONF, 1) + b"".join(
        struct.pack(">I", a) for a in arfcn_list
    )
    seq.append(("PM_CONF DCS 940-954 (expected)", conf))

```

```

seq.append(("DONE sub=1", encode_done(1)))

# GSM 900 scan : ARFCN 1..124
seq.append(("PM_REQ GSM900 1-124", encode_pm_req(BAND_GSM900, 1, 124)))
arfcn_list2 = list(range(1, 125))
conf2 = struct.pack(">II", CMD_PM_CONF, 0) + b"".join(
    struct.pack(">I", a) for a in arfcn_list2
)
seq.append(("PM_CONF GSM900 1-124 (expected)", conf2))
seq.append(("DONE sub=1", encode_done(1)))
return seq

# ----- Main -----

def hex_pretty(b: bytes, line_len: int = 32) -> str:
    out = []
    for i in range(0, len(b), line_len):
        chunk = b[i : i + line_len]
        hexpart = " ".join(f"{x:02x}" for x in chunk)
        asciipart = "".join(chr(x) if 32 <= x < 127 else "." for x in chunk)
        out.append(f" {hexpart:<{line_len*3}} {asciipart}")
    return "\n".join(out)

def main():
    ap = argparse.ArgumentParser(description="Replay L1CTL PM scan payload")
    ap.add_argument("--dump", action="store_true",
                    help="print expected bytes (no I/O)")
    ap.add_argument("--to-socket", metavar="PATH",
                    help="send PM_REQs as L1CTL frames to unix socket")
    ap.add_argument("--hdlc-pty", metavar="PATH",
                    help="wrap in HDLC DLCI 5 and write to PTY")
    ap.add_argument("--band", choices=["gsm900", "dcs1800"], default="gsm900")
    ap.add_argument("--arfcn-lo", type=int, default=1)
    ap.add_argument("--arfcn-hi", type=int, default=124)
    args = ap.parse_args()

    band = BAND_DCS1800 if args.band == "dcs1800" else BAND_GSM900
    custom_req = encode_pm_req(band, args.arfcn_lo, args.arfcn_hi)

    if args.dump:

```

```

print("=== Observed sequence reproduction ===\n")
for label, payload in build_observed_sequence():
    print(f"-- {label} ({len(payload)} bytes) --")
    print(hex_pretty(payload))
    print()
print("=== Custom PM_REQ ===")
print(f"  band={args.band} arfcn={args.arfcn_lo}-{args.arfcn_hi}")
print(hex_pretty(custom_req))
print()
print("=== HDLC framed (DLCI 5) ===")
framed = hdlc_frame(DLCI_SERCOMM, custom_req)
print(hex_pretty(framed))
return

if args.to_socket:
    s = socket.socket(socket.AF_UNIX, socket.SOCK_STREAM)
    s.connect(args.to_socket)
    # L1CTL frame = 16-bit BE length prefix + payload (per osmocom-bb l1ctl)
    framed = struct.pack(">H", len(custom_req)) + custom_req
    s.sendall(framed)
    print(f"sent {len(framed)} bytes to {args.to_socket}")
    # Optionally wait for CONF
    s.settimeout(2.0)
    try:
        r = s.recv(4096)
        print(f"recv {len(r)} bytes :")
        print(hex_pretty(r))
    except socket.timeout:
        print("(no response in 2s)")
    s.close()
    return

if args.hdlc_pty:
    framed = hdlc_frame(DLCI_SERCOMM, custom_req)
    with open(args.hdlc_pty, "wb") as f:
        f.write(framed)
    print(f"sent HDLC frame ({len(framed)} bytes) to {args.hdlc_pty}")
    return

ap.print_help()
sys.exit(1)

```

```
if __name__ == "__main__":
    main()
```

0.3.61 /opt/GSM/qosmo-overlay/scripts/test_record_replay.sh

5599 octets, 131 lignes → 131 lignes

```
#!/bin/bash
# test_record_replay.sh - diagnostic determinisme : record live + N replays
#
# Workflow :
#   1. RECORD : run normal avec BSP_DUMP_RX_FILE=/tmp/bsp_rx.dump
#   2. REPLAY 1/2/3 : runs avec CALYPSO_BSP_REPLAY_FILE=/tmp/bsp_rx.dump
#   3. Affiche tableau comparatif des signatures d_fb_det / SOFT-RESET-TRIG /
#       BURST-IN / etc. pour decider : determinisme restaure (= course feed) ou
#       encore nondeterministe (= autre source).
#
# Usage : ./test_record_replay.sh [duration_sec]      (default 30)
#
# Hypotheses :
#   - run.sh est dans le PATH ou /root/run.sh
#   - CALYPSO_NO_ATTACH=1 supporte par run.sh pour spawn en background

set -u
DUR="${1:-30}"
DUMP=/tmp/bsp_rx.dump
LOGDIR=/tmp/record_replay_logs
mkdir -p "$LOGDIR"
RUN_SH="${RUN_SH:-${QEMU_TREE}/run.sh}"
QEMU_LOG="${QEMU_LOG:-/root/qemu.log}"

# ---- helper : kill toute la stack et nettoyer ----
killall_run() {
    tmux kill-session -t calypso 2>/dev/null || true
    killall -9 qemu-system-arm osmo-bts-trx mobile osmocon osmo-trx-ipc 2>/dev/null || true
    pkill -9 -f bridge.py 2>/dev/null || true
    pkill -9 -f calypso-ipc-device 2>/dev/null || true
    pkill -9 -f irda_capture.py 2>/dev/null || true
    sleep 1
}
```

```

# ---- helper : lance un run en background avec env, attend DUR secondes ----
do_run() {
    local label="$1"; shift
    echo "==== $label : start (timeout=${DUR}s) ====="
    killall_run

    # CALYPSO_NO_ATTACH=1 : run.sh exit apres tmux session lance en bg
    CALYPSO_NO_ATTACH=1 "$@" bash "$RUN_SH" > "$LOGDIR/$label.stdout" 2>&1 &
    local launcher_pid=$!

    # Attente DUR secondes, qemu.log se remplit
    sleep "$DUR"

    # Snapshot qemu.log et kill stack
    cp -f "$QEMU_LOG" "$LOGDIR/$label.qemu.log" 2>/dev/null || true
    killall_run
    wait "$launcher_pid" 2>/dev/null || true
    echo "==== $label : done, log=$LOGDIR/$label.qemu.log ($(wc -l < $LOGDIR/$label.qemu.log))"
    echo
}

# ---- helper : extrait une signature compact d'un qemu.log ----
sig() {
    local log="$1"
    if [ ! -s "$log" ]; then
        echo " (vide ou inexistant)"
        return
    fi
    local boot_redirect soft_reset_trig fb_wr_total fb_wr_dom_pc fb_wr_dom_val
    local latch_consume_dom burst_mod burst_tonal burst_silent
    local task5 task6 sync_demod fbsb_state imr_zero last_insn

    boot_redirect=$(grep -c 'SILICON-BOOT-REDIRECT' "$log")
    soft_reset_trig=$(grep -c 'SOFT-RESET-TRIG' "$log")
    fb_wr_total=$(grep -c 'FB-DET-WR' "$log")
    fb_wr_dom_pc=$(grep 'FB-DET-WR' "$log" | grep -oE 'PC=0x[0-9a-f]+' | sort | uniq -c | sort -n)
    fb_wr_dom_val=$(grep 'FB-DET-WR' "$log" | grep -oE '<= 0x[0-9a-f]+' | sort | uniq -c | sort -n)
    latch_consume_dom=$(grep 'LATCH-CONSUME' "$log" | grep -oE '= 0x[0-9a-f]+' | sort | uniq -c | sort -n)
    burst_mod=$(grep 'BURST-IN' "$log" | grep -c MODULATED)
    burst_tonal=$(grep 'BURST-IN' "$log" | grep -c TONAL_FB)
    burst_silent=$(grep 'BURST-IN' "$log" | grep -c SILENT)
    task5=$(grep 'on_dsp_task_change' "$log" | grep -c 'task=5')

```

```

task6=$(grep 'on_dsp_task_change' "$log" | grep -c 'task=6')
sync_demod=$(grep '\[fbsb\]' "$log" | grep -oE 'last\(\snr=[0-9]+ toa=[0-9]+ ang=[0-9]+ p'
fbsb_state=$(grep '\[fbsb\]' "$log" | grep -oE 'state=[A-Z_0-9]+' | sort | uniq -c | tr
imr_zero=$(grep -c 'IMR-W \*ZERO\*' "$log")
last_insn=$(grep -oE 'insn=[0-9]+' "$log" | grep -oE '[0-9]+' | sort -n | tail -1)

printf "  SILICON-BOOT-REDIRECT : %s\n" "$boot_redirect"
printf "  SOFT-RESET-TRIG       : %s\n" "$soft_reset_trig"
printf "  FB-DET-WR total        : %s\n" "$fb_wr_total"
printf "  FB-DET-WR dom PC       : %s\n" "$fb_wr_dom_pc"
printf "  FB-DET-WR dom val      : %s\n" "$fb_wr_dom_val"
printf "  LATCH-CONSUME dom      : %s\n" "$latch_consume_dom"
printf "  BURST-IN cat (M/T/S)   : %s / %s / %s\n" "$burst_mod" "$burst_tonal" "$burst_si
printf "  ARM task=5 / task=6    : %s / %s\n" "$task5" "$task6"
printf "  fbsb state distrib     : %s\n" "$fbsb_state"
printf "  a_sync_demod uniq      : %s\n" "$sync_demod"
printf "  IMR-W ZERO             : %s\n" "$imr_zero"
printf "  Last insn_count        : %s\n" "$last_insn"
}

# ---- 1. RECORD ----
rm -f "$DUMP"
do_run RECORD env BSP_DUMP_RX_FILE="$DUMP"

if [ ! -s "$DUMP" ]; then
    echo "!!! RECORD failed: dump '$DUMP' vide ou absent (taille $(stat -c%s "$DUMP" 2>/dev/
    echo "Verifier qu'au moins 1 burst a ete livre au DSP (gate BDLENA)."
    exit 1
fi
echo ">>> RECORD dump size: $(stat -c%s "$DUMP") bytes"
echo

# ---- 2. REPLAY 1/2/3 ----
for i in 1 2 3; do
    do_run REPLAY$i env CALYPSO_BSP_REPLAY_FILE="$DUMP"
done

# ---- 3. Comparaison ----
echo "#####"
echo "##          SIGNATURE COMPARISON TABLE          ##"
echo "#####"
for label in RECORD REPLAY1 REPLAY2 REPLAY3; do

```

```

    echo "----- $label -----"
    sig "$LOGDIR/$label.qemu.log"
    echo
done

echo "#####"
echo "## Interpretation :                ##"
echo "## - REPLAY 1/2/3 identiques => determinisme ##"
echo "##   retabli avec feed fixe => course feed   ##"
echo "##   est la root cause                        ##"
echo "## - REPLAY 1/2/3 differents => nondeter-   ##"
echo "##   minisme ailleurs (DSP/host thread/etc) ##"
echo "#####"

```

0.3.62 /opt/GSM/qosmo-overlay/scripts/qosmo/test-base.sh

2192 octets, 46 lignes → 46 lignes

```

#!/bin/bash
# Le test de ce que qosmo EST : une plateforme qui se construit sans couche 1,
# et qui dit laquelle elle porte.
#
# [2026-09-16] Il remplace les 19 fichiers de test herites de qemu-calypso, qui
# visaient tous soit l'orchestration (run.sh, /tmp/calypso), soit le DSP (c54x,
# calypso_dsp.txt) - deux choses absentes de ce depot. Une suite dont le sujet
# n'est pas dans l'arbre ne teste rien, elle rassure.
set -uo pipefail
cd "$(dirname "${BASH_SOURCE[0]}")/../../.."
RACINE="$PWD"
B="${QOSMO_BUILD:-$RACINE/build}"
QEMU="$B/qemu-system-arm"
rouge=$'\033[31m'; vert=$'\033[32m'; nc=$'\033[0m'
ok=0; ko=0

verdict() { # <attendu> <obtenu> <intitule>
    if [ "$2" = "$1" ]; then printf '%s  %s %s\n' "$vert" "$nc" "$3"; ok=$((ok+1))
    else printf '%s  %s %s\n      attendu : %s\n      obtenu  : %s\n' "$rouge" "$nc" "$3" "$1"
    }
}

[ -x "$QEMU" ] || { echo "qemu-system-arm introuvable dans $B - construire d'abord"; exit 2;

```

```

# 1. la machine existe
verdict "oui" "$("$QEMU" -M help 2>/dev/null | grep -q '^calypso ' && echo oui || echo non)"
    "la machine 'calypso' est enregistree"

# 2. elle nomme la couche 1 reellement liee. C'est le seul moyen court de
#    savoir ce qu'on a construit ; une L1 qui s'invite dans le build se voit ici.
l1="$("$QEMU" -M help 2>/dev/null | sed -n 's/.*couche 1 : \([a-z0-9]*\)).*/\1/p')"
attendu="${QOSMO_L1_ATTENDUE:-aucune}"
verdict "$attendu" "${l1:-<absent>}" "la machine annonce sa couche 1 (QOSMO_L1_ATTENDUE=$attendu)"

# 3. sans L1, aucun symbole de couche 1 ne doit avoir ete lie
if [ "$attendu" = "aucune" ]; then
    fuites="$(nm -C "$QEMU" 2>/dev/null | grep -cE ' [Tt] (calypso_l1_grgsm|calypso_c54x|c54x)' )"
    verdict "0" "${fuites:-0}" "aucun symbole de couche 1 lie dans la base nue"
fi

# 4. le binaire demarre et tient la seconde (pas de segfault au realize)
timeout 3 "$QEMU" -M calypso -nographic -serial null -monitor none >/dev/null 2>&1
verdict "124" "$?" "la machine demarre et tourne (124 = toujours vivante au timeout)"

echo
if [ "$ko" -gt 0 ]; then printf '%s%d echec(s), %d succes%s\n' "$rouge" "$ko" "$ok" "$nc"; else
printf '%s%d succes%s\n' "$vert" "$ok" "$nc"

```

0.3.63 /opt/GSM/qosmo-overlay/scripts/qosmo/verif-liens-doc.sh

3399 octets, 67 lignes → 67 lignes

```

#!/bin/bash
# Verifie que tout chemin local cite dans un .md du depot existe reellement.
#
# [2026-09-16] Ce script existe a cause d'un cas precis. Apres le split de
# qemu-calypso, les README de qosmo-grgsm et qosmo-dsp sont restes IDENTIQUES
# (meme md5) et ont continue a renvoyer vers hw/arm/calypso/doc/ETAT_ACTUEL.md
# en le presentant comme « la source de verite, en cas de conflit c'est lui qui
# prime » - fichier qui n'existait plus dans l'un des deux. Meme histoire pour
# calypso_dsp.txt, parti avec le C54x, encore symlinke par le Dockerfile
# d'osmo-operator « pour les outils d'analyse » qui avaient migre avec lui.
#
# Aucune de ces deux ruptures n'a fait de bruit. Une doc qui pointe dans le vide
# ne plante pas : elle ment, et elle ment d'autant mieux qu'elle est detaillee.

```

```

set -uo pipefail
cd "$(dirname "${BASH_SOURCE[0]}")/../../.."

rouge=$'\033[31m'; vert=$'\033[32m'; nc=$'\033[0m'
casses=0; verifies=0

while IFS= read -r md; do
    # 1. les cibles de liens markdown [texte](chemin)
    # 2. les chemins entre backticks qui ressemblent a des fichiers du depot
    {
        grep -oE '\]\([^\)]*\)' "$md" | sed -e 's/~/\\/' -e 's/)/$/'
        # On retire d'abord les liens markdown complets : un nom entre backticks
        # qui sert de LIBELLE ([`x.c`](url)) est deja documente par sa cible,
        # traitee au-dessus. Sans ca, une reference inter-depot legitime est
        # signalee comme cassee.
        sed -E 's/\[[^\]]*\]\([^\)]*\)//g' "$md" \
        | grep -oE '`[a-zA-Z0-9_./-]+\.(c|h|sh|py|md|txt|cfg|conf|build|yaml)`' | tr -d '`'
    } 2>/dev/null | sort -u | while IFS= read -r cible; do
        case "$cible" in
            http://*|https://*|mailto:*|') continue ;;
            esac
            cible="${cible%#*}"
            [ -z "$cible" ] && continue
            # relatif au .md, puis a la racine du depot
            base="$(dirname "$md")"
            # Un nom nu entre backticks (« calypso_trx.c ») est une MENTION, pas un
            # chemin : il suffit qu'un fichier de ce nom existe quelque part dans le
            # depot. Un chemin explicite (« hw/arm/.../x.c », ou la cible d'un lien
            # markdown) doit exister LA ou il est ecrit - c'est le cas qui a laisse
            # passer ETAT_ACTUEL.md.
            if [ -e "$base/$cible" ] || [ -e "$cible" ]; then
                echo "OK"
            elif [ "${cible%/*}" = "$cible" ] && \
                [ -n "$(find . -name "$cible" -not -path './build*' -print -quit 2>/dev/null)" ]; then
                echo "OK"
            else
                printf '%s %s:%s cible introuvable : %s%s\n' \
                    "$rouge" "$md" "$(grep -n -- "$cible" "$md" | head -1 | cut -d: -f1)" "$c"
                echo "KO"
            fi
        done
    done
done < <(git ls-files '*.md' 2>/dev/null || find . -name '*.md' -not -path './build/*') > /tr

```

```

# grep -c imprime « 0 » quand il ne trouve rien ET rend 1 : un « || echo 0 »
# ici produit « 0\n0 », que [ -gt ] refuse. On ignore le code, on garde la sortie.
verifies=$(grep -c . /tmp/.qosmo-liens.$$ 2>/dev/null); verifies=${verifies:-0}
casses=$(grep -c '^K0$' /tmp/.qosmo-liens.$$ 2>/dev/null); casses=${casses:-0}
rm -f /tmp/.qosmo-liens.$$

if [ "$casses" -gt 0 ]; then
    printf '%s%d lien(s) casse(s) sur %d verifie(s)%s\n' "$rouge" "$casses" "$verifies" "$nc"
    exit 1
fi
printf '%s%d lien(s) de doc verifie(s), aucun casse%s\n' "$vert" "$verifies" "$nc"

```

0.3.64 /opt/GSM/qosmo-overlay/tests/run_tests.sh

1463 octets, 32 lignes → 32 lignes

```

#!/bin/bash
# Runner tests Calypso - pile VIVANTE (lit /root/qemu.log, /root/osmocon.log...).
# COMPLET : tout tests/*.py (calypso + génériques top-level), on IGNORE seulement
# functional/ + guest-debug/ (suite d'intégration QEMU : binaires/downloads absents).
# --continue-on-collection-errors = ne bloque jamais.
# Sortie EN DIRECT (pytest -v + tee, line-buffered) - chaque test s'affiche au fil.
# Génère grafcet.html/md + rapport_final.pdf via hook conftest::pytest_sessionfinish.
#
# Usage : run_tests.sh [loop]
set -u
PY="${CALYPSO_PY:-/root/.env/bin/python3}"
TESTS=${QEMU_TREE}/tests
OUT="${CALYPSO_TEST_OUT:-/root/test_reports}"
PERIOD="${CALYPSO_TEST_PERIOD:-90}"
mkdir -p "$OUT"

run_once() {
    echo "===== pytest COMPLET (pile vivante) $(date +%H:%M:%S) ====="
    ( cd "$TESTS" && CALYPSO_TEST_OUT="$OUT" \
        stdbuf -oL -eL "$PY" -m pytest -v -ra --color=yes \
        -p no:cacheprovider --continue-on-collection-errors \
        --ignore=functional --ignore=guest-debug --ignore=migration \
        --ignore=qemu-iotests 2>&1 | stdbuf -oL tee "$OUT/pytest_last.log" )
    echo "-> $OUT/{grafcet.html, grafcet.md, rapport_final.pdf} (log: $OUT/pytest_last.log)"
}

```

```

if [ "${1:-}" = "loop" ]; then
    echo "[run_tests] LOOP mode, période=${PERIOD}s (Ctrl-C pour stop) - sortie en direct"
    while true; do run_once; echo "[run_tests] prochain run dans ${PERIOD}s..."; sleep "$PERIOD"
else
    run_once
fi

```

0.3.65 /opt/GSM/qosmo-overlay/tests/test_runtime_net_fs.py

3406 octets, 93 lignes → 93 lignes

```

"""
test_runtime_net_fs.py - surfaces réseau et FS du run (Phase 2 plan IrDA).

Markers : `runtime_net`, `runtime_fs`. Tests anti-régression sur des aspects
opérationnels du long-running (leak FDs, log size cap, ports inattendus,
disk container).
"""
from __future__ import annotations

import os
import subprocess

import pytest

CONTAINER = os.environ.get("CALYPSO_CONTAINER", "trying")
QEMU_LOG_CT = os.environ.get("CALYPSO_QEMU_LOG", "/root/qemu.log")
INSIDE = os.path.exists("/.dockerenv")

def _dexec_sh(cmd: str, timeout: float = 4.0) -> subprocess.CompletedProcess:
    """Exec un shell command à l'intérieur du container."""
    if INSIDE:
        return subprocess.run(["bash", "-c", cmd],
                               capture_output=True, text=True, timeout=timeout)
    return subprocess.run(["docker", "exec", CONTAINER, "bash", "-c", cmd],
                           capture_output=True, text=True, timeout=timeout)

# -----
@pytest.mark.runtime_net

```

```

def test_qemu_ports_listening():
    """Ports attendus listening : 1234 (gdb stub) + 4247 (mobile VTY)."""
    r = _dexec_sh("ss -tln 2>/dev/null || netstat -tln 2>/dev/null")
    out = r.stdout
    missing = []
    for port in ("1234", "4247"):
        if f":{port} " not in out and f":{port}\n" not in out:
            missing.append(port)
    assert not missing, f"ports manquants : {missing} - out : {out[:300]!r}"

@pytest.mark.runtime_net
def test_no_unexpected_high_ports():
    """Aucun listener sur port > 60000 (régression leak)."""
    r = _dexec_sh("ss -tln 2>/dev/null | awk 'NR>1 {print $4}' | grep -oE ':[0-9]+$' | sort -n")
    high = []
    for line in r.stdout.split():
        if line.startswith(":"):
            try:
                p = int(line[1:])
                if p > 60000:
                    high.append(p)
            except ValueError:
                continue
    assert not high, f"ports inattendus (>60000) : {high}"

@pytest.mark.runtime_fs
def test_qemu_fd_usage_below_limit():
    """qemu-system-arm doit avoir moins de 1024 fd (early warning leak)."""
    r = _dexec_sh("pid=$(pgrep qemu-system-arm | head -1); [ -n \"$pid\" ] && ls /proc/$pid/fd")
    s = r.stdout.strip()
    if not s or s == "0":
        pytest.skip("qemu-system-arm pas trouvé")
    n_fds = int(s)
    assert n_fds < 1024, f"fd leak probable : {n_fds} fds (limite 1024)"
    print(f"\n[fs] qemu fd count = {n_fds}")

@pytest.mark.runtime_fs
def test_qemu_log_disk_size_under_2gb():
    """/root/qemu.log < 2 GB (anti-leak log)."""

```

```

r = _dexec_sh(f"stat -c%s {QEMU_LOG_CT} 2>/dev/null")
s = r.stdout.strip()
if not s:
    pytest.skip(f"{QEMU_LOG_CT} absent")
size_bytes = int(s)
assert size_bytes < 2 * 1024**3, \
    f"qemu.log trop gros : {size_bytes / 1024**2:.0f} MB (limite 2 GB)"
print(f"\n[fs] qemu.log = {size_bytes / 1024**2:.1f} MB")

@pytest.mark.runtime_fs
def test_container_disk_space_above_min():
    """Le filesystem `/` du container a au moins 500 MB libre."""
    r = _dexec_sh("df / | tail -1 | awk '{print $4}'")
    s = r.stdout.strip()
    if not s:
        pytest.skip("df pas dispo")
    free_kb = int(s)
    assert free_kb > 500_000, \
        f"disque container trop plein : {free_kb // 1024} MB libre (min 500 MB)"
    print(f"\n[fs] / free = {free_kb // 1024} MB")

```

0.3.66 /opt/GSM/qosmo-overlay/tests/test_timer_invariants.py

11360 octets, 300 lignes → 300 lignes

```

"""
test_timer_invariants.py - tests sur les compteurs de timers instrumentés.

Cible les logs `[tdma]`, `[frame_irq]`, `[kick]` produits par
`calypso_trx.c` + le CSV `/tmp/log_timeline.csv` produit par `log_timeline.py`.

Invariants testés :
- log [tdma] présent (rebuild fait, env CALYPSO_DSP_BUDGET respecté)
- dsp_n_exec_* <= dsp_budget (sanity)
- dsp saturation : si dsp_n_exec_5 == budget sur N ticks consécutifs → signal
- period virtual entre [tdma] thinned = 1000 × 4.615 ms ± 2%
- ratio frame_irq/tdma 1
- table log_timeline.csv produite, pas de bucket totalement vide

Lancement :

```

```

    pytest -v -m timer_invariant
"""
from __future__ import annotations

import csv
import os
import re
import shutil
import subprocess
from pathlib import Path

import pytest

CONTAINER = os.environ.get("CALYPSO_CONTAINER", "trying")
QEMU_LOG = os.environ.get("CALYPSO_QEMU_LOG", "/root/qemu.log")
CSV_PATH = os.environ.get("CALYPSO_TIMELINE_CSV", "/tmp/log_timeline.csv")
# Marqueur Docker standard : présent dans container, absent sur host.
INSIDE = os.path.exists("/.dockerenv")

TDMA_RE = re.compile(
    r'\[tdma\] tick #(\d+) fn=(\d+) t_virt=(\d+) '
    r'dsp_n_exec_2=(-?\d+) dsp_n_exec_5=(-?\d+) '
    r'dsp_insn_total=(\d+) budget=(\d+)'
)
FIRQ_RE = re.compile(r'\[frame_irq\] lower #(\d+) t_virt=(\d+)')
KICK_RE = re.compile(r'\[kick\] fire #(\d+) vt=(\d+) rt=(\d+)')

GSM_TDMA_NS = 4_615_000

def _read_lines(path: str, tail_n: int = 100000) -> list[str]:
    if INSIDE:
        try:
            with open(path, errors="replace") as f:
                return f.readlines()[-tail_n:]
        except FileNotFoundError:
            return []
    out = subprocess.run(
        ["docker", "exec", CONTAINER, "bash", "-c",
         f"tail -n {tail_n} {path} 2>/dev/null"],
        capture_output=True, text=True, timeout=8)
    return out.stdout.splitlines(keepends=True)

```

```

def _parse_tdma() -> list[dict]:
    out = []
    for line in _read_lines(QEMU_LOG):
        m = TDMA_RE.search(line)
        if not m: continue
        out.append({
            "tick":          int(m.group(1)),
            "fn":            int(m.group(2)),
            "t_virt_ns":     int(m.group(3)),
            "dsp_n_exec_2":  int(m.group(4)),
            "dsp_n_exec_5":  int(m.group(5)),
            "dsp_insn_total": int(m.group(6)),
            "budget":        int(m.group(7)),
        })
    return out

def _parse_firq() -> list[dict]:
    out = []
    for line in _read_lines(QEMU_LOG):
        m = FIRQ_RE.search(line)
        if not m: continue
        out.append({"n": int(m.group(1)), "t_virt_ns": int(m.group(2))})
    return out

def _parse_kick() -> list[dict]:
    out = []
    for line in _read_lines(QEMU_LOG):
        m = KICK_RE.search(line)
        if not m: continue
        out.append({"n": int(m.group(1)),
                    "vt": int(m.group(2)),
                    "rt": int(m.group(3))})
    return out

@pytest.fixture(scope="module")
def tdma_samples():
    s = _parse_tdma()
    if not s:

```

```

        pytest.skip(f"no [tdma] log lines in {QEMU_LOG} - rebuild QEMU pas appliqué ?")
    return s

@pytest.fixture(scope="module")
def firq_samples():
    s = _parse_firq()
    if not s:
        pytest.skip("no [frame_irq] log lines")
    return s

@pytest.fixture(scope="module")
def kick_samples():
    s = _parse_kick()
    if not s:
        pytest.skip("no [kick] log lines")
    return s

# -----
# Tests timers
# -----

@pytest.mark.timer_invariant
def test_tdma_log_present(tdma_samples):
    """Au moins 2 lignes [tdma] (= 2000+ ticks observés)."""
    assert len(tdma_samples) >= 2, \
        f"only {len(tdma_samples)} [tdma] entries - wait for 2000+ ticks (~9s virtual)"

@pytest.mark.timer_invariant
def test_dsp_n_exec_within_budget(tdma_samples):
    """dsp_n_exec_2 et dsp_n_exec_5 doivent rester <= budget."""
    for s in tdma_samples:
        assert 0 <= s["dsp_n_exec_2"] <= s["budget"], \
            f"tick #{s['tick']}: dsp_n_exec_2={s['dsp_n_exec_2']} hors [0, {s['budget']}]"
        assert 0 <= s["dsp_n_exec_5"] <= s["budget"], \
            f"tick #{s['tick']}: dsp_n_exec_5={s['dsp_n_exec_5']} hors [0, {s['budget']}]"

@pytest.mark.timer_invariant
def test_dsp_budget_saturation_signal(tdma_samples):
    """Si dsp_n_exec_5 == budget sur les 3 dernières entrées → DSP saturé,
    XFAIL avec message - c'est un état observable, pas un bug du test."""

```

```

if len(tdma_samples) < 3:
    pytest.skip("need >= 3 samples")
last3 = tdma_samples[-3:]
saturated = all(s["dsp_n_exec_5"] == s["budget"] for s in last3)
if saturated:
    pytest.xfail(
        f"DSP satur  : dsp_n_exec_5 == budget ({last3[0]['budget']}) sur "
        f"les 3 derniers [tdma] ticks. Descendre CALYPSO_DSP_BUDGET cassera "
        f"probablement la fb-det convergence. Voir REPORT_CLAUDE_WEB_20260516_DSP_OVERRUN")
# Else: pass - DSP non satur   en steady state
assert True

@pytest.mark.timer_invariant
def test_tdma_period_virtual_close_to_target(tdma_samples):
    """P  riode virtual entre 2 [tdma] thinned (1000 ticks) 4.615 s    2%."""
    if len(tdma_samples) < 2:
        pytest.skip("need >= 2 samples")
    dt_virt_ns = tdma_samples[-1]["t_virt_ns"] - tdma_samples[-2]["t_virt_ns"]
    dtick = tdma_samples[-1]["tick"] - tdma_samples[-2]["tick"]
    if dtick != 1000:
        pytest.skip(f"non-1000-tick gap: {dtick}")
    expected_ns = 1000 * GSM_TDMA_NS
    rel_err = abs(dt_virt_ns - expected_ns) / expected_ns
    # Sous icount le temps VIRTUAL n'est pas pr  cis (d  terministe mais pas
    # cal   sur le mur) : on garde un contr  le de sant   large, pas un assert
    # dur   2%. On   choue seulement sur un ordre de grandeur manifestement faux.
    if rel_err > 0.50:
        pytest.skip(
            f"period_virtual = {dt_virt_ns:,} ns vs target {expected_ns:,} "
            f"(err {rel_err*100:.1f}%) - VIRTUAL sous icount, hors fen  tre de sant  ")
    assert rel_err <= 0.50, \
        f"period_virtual = {dt_virt_ns:,} ns vs target {expected_ns:,} (err {rel_err*100:.2f}%)

@pytest.mark.timer_invariant
def test_frame_irq_per_tdma_ratio(tdma_samples, firq_samples):
    """Ratio frame_irq/tdma 1 (le frame_irq lower fire    chaque tdma_tick)."""
    if not firq_samples:
        pytest.skip("no firq samples")
    n_tdma = tdma_samples[-1]["tick"]
    n_firq = firq_samples[-1]["n"]

```

```

if n_tdma == 0:
    pytest.skip()
ratio = n_firq / n_tdma
# Tolérance large : 0.5..1.5 (frame_irq peut ne pas fire au boot avant que
# le hardware-frame-arm soit set par firmware)
assert 0.5 <= ratio <= 1.5, \
    f"frame_irq/tdma = {ratio:.2f} (firq={n_firq} / tdma={n_tdma})"

@pytest.mark.timer_invariant
def test_kick_realtime_cadence(kick_samples):
    """Kick fires sur REALTIME ~5 ms. Délta moyen entre 2 logs thinned 1/200
    devrait être ~1 s wall = ~1e9 ns realtime."""
    if len(kick_samples) < 3:
        pytest.skip("few kick samples")
    rts = [s["rt"] for s in kick_samples]
    deltas = [(rts[i+1] - rts[i]) for i in range(len(rts)-1)]
    mean = sum(deltas) / len(deltas)
    # 200 fires × 5 ms = 1 s = 1e9 ns. Tolérance ±50%.
    assert 0.5e9 <= mean <= 1.5e9, \
        f"kick realtime mean delta = {mean/1e9:.2f}s, expected ~1s (1e9 ns)"

# -----
# Tests sur la table log_timeline.csv
# -----

def _has_docker(): return shutil.which("docker") is not None

def _csv_exists() -> tuple[bool, str]:
    """(exists, where) - check CSV side : local ou container."""
    if Path(CSV_PATH).exists():
        return (True, "local")
    if _has_docker():
        r = subprocess.run(
            ["docker", "exec", CONTAINER, "test", "-f", CSV_PATH],
            capture_output=True, timeout=4)
        if r.returncode == 0:
            return (True, "container")
    return (False, "none")

def _generate_csv() -> str:
    """Tente de produire le CSV. Priorité : container si docker dispo

```

```

(qemu.log canonique côté container), sinon local SEULEMENT si in-container.
Évite la race où on lit un /root/qemu.log host stale.
"""
# 1) Container path (priorité) - sauf si on EST dans le container
if _has_docker() and not INSIDE:
    r = subprocess.run(
        ["docker", "exec", CONTAINER, "python3",
         "/opt/GSM/osmo-qemu-calypto/log_timeline.py",
         "--bucket-s", "10", "--csv", CSV_PATH],
        capture_output=True, text=True, timeout=20)
    return r.stderr or r.stdout[:200]
# 2) Local (in-container : /root/qemu.log est canonique)
script_local = Path(__file__).resolve().parent.parent / "log_timeline.py"
if script_local.exists() and Path("/root/qemu.log").exists():
    r = subprocess.run(
        ["python3", str(script_local), "--bucket-s", "10", "--csv", CSV_PATH],
        capture_output=True, text=True, timeout=20)
    return r.stderr or r.stdout[:200]
return "no path to run log_timeline.py"

@pytest.mark.timer_invariant
def test_log_timeline_csv_produced():
    """log_timeline.py doit avoir produit le CSV (local ou container).
    Si absent, tente une génération à la volée puis re-check."""
    ok, where = _csv_exists()
    if ok:
        return
    err = _generate_csv()
    ok, where = _csv_exists()
    assert ok, f"{CSV_PATH} non produit (local={Path(CSV_PATH).exists()}, docker={_has_docker})

def _read_csv_rows() -> list[dict]:
    if INSIDE:
        try:
            with open(CSV_PATH) as f:
                return list(csv.DictReader(f))
        except FileNotFoundError:
            return []
    out = subprocess.run(
        ["docker", "exec", CONTAINER, "cat", CSV_PATH],
        capture_output=True, text=True, timeout=5)

```

```

    return list(csv.DictReader(out.stdout.splitlines()))

@pytest.mark.timer_invariant
def test_log_timeline_csv_no_dead_bucket():
    """Aucun bucket avec 0 events all-sources confondus (= run pas stallé)."""
    rows = _read_csv_rows()
    if not rows:
        pytest.skip("CSV empty")
    dead = []
    sources = ["qemu", "bridge", "osmocon", "mobile"]
    for r in rows:
        total = sum(int(r.get(s, 0) or 0) for s in sources)
        if total == 0:
            dead.append(r["t_rel"])
    assert not dead, f"buckets totalement vides à t_rel = {dead[:5]} (run stallé ?)"

@pytest.mark.timer_invariant
def test_log_timeline_csv_steady_qemu_rate():
    """qemu rate par bucket doit rester dans une fenêtre raisonnable
    (pas de spike 10× ni de chute 10× sur la majorité des buckets)."""
    rows = _read_csv_rows()
    if len(rows) < 5:
        pytest.skip("too few buckets")
    qemu_rates = [int(r.get("qemu", 0) or 0) for r in rows[1:]] # skip warmup
    if not qemu_rates: pytest.skip("no qemu rates")
    mean = sum(qemu_rates) / len(qemu_rates)
    outliers = [v for v in qemu_rates if v > 10*mean or v < mean/10]
    # Tolère jusqu'à 20% d'outliers
    assert len(outliers) <= 0.2 * len(qemu_rates), \
        f"{len(outliers)}/{len(qemu_rates)} buckets outliers (>10× ou <1/10 mean={mean:.0f})

```

0.3.67 /opt/GSM/qosmo-overlay/tests/test_timer_physical_audit.py

9977 octets, 261 lignes → 261 lignes

```

"""
test_timer_physical_audit.py - audit physique des timers + graphes temporels.

Vérifie que chaque param timer dans le code respecte l'équation physique

```

documentée dans TIMING_PHYSICS.md. Pour chaque timer, mesure aussi le rate effectif côté run (depuis qemu.log / bridge.log) et compare au voulu théorique.

Génère des graphes mermaid xychart pour visualisation des périodes.

Markers :

- osmocom_clock (déjà enregistré)
- timer_audit (nouveau)
- timer_graph (nouveau)

"""

```
from __future__ import annotations

import os
import re
import subprocess
from pathlib import Path
from collections import defaultdict
from statistics import mean, median, stdev
```

```
import pytest
```

```
CONTAINER = os.environ.get("CALYPSO_CONTAINER", "trying")
INSIDE = os.path.exists("/.dockerenv")
QEMU_SRC = "/opt/GSM/osmo-qemu-calypso"
QEMU_LOG = "/root/qemu.log"
(reMOVED) = "/tmp/bridge.log"
BTS_LOG = "/tmp/bts.log"
GRAPHS_OUT = "/tmp/timer_graphs.mmd"
```

```
def _read(path: str) -> str:
    if INSIDE:
        try:
            return Path(path).read_text(errors="replace")
        except Exception:
            return ""
    try:
        r = subprocess.run(
            ["docker", "exec", CONTAINER, "cat", path],
            capture_output=True, text=True, timeout=10)
        return r.stdout
```

```

except Exception:
    return ""

# === Audit physique : voulu vs réel =====
# Format : (param_name, voulu_value, voulu_units, source_file, search_regex,
#          parse_fn, tolerance_pct)
AUDIT_PARAMS = [
    # 1. TINTO_PERIOD_NS - voulu 60_000_000/13 = 4_615_384 ns
    ("TINTO_PERIOD_NS", 4_615_384, "ns",
     "hw/arm/calypso/calypso_tint0.h",
     r"#define\s+TINTO_PERIOD_NS\s+(\d+)",
     int, 0.1),
    # 2. GSM_TDMA_NS - voulu 4_615_384 ns
    ("GSM_TDMA_NS", 4_615_384, "ns",
     "include/hw/arm/calypso/calypso_trx.h",
     r"#define\s+GSM_TDMA_NS\s+(\d+)",
     int, 0.1),
    # 3. GSM_HYPERFRAME - exact 2_715_648
    ("GSM_HYPERFRAME", 2_715_648, "frames",
     "hw/arm/calypso/calypso_tint0.h",
     r"#define\s+GSM_HYPERFRAME\s+(\d+)",
     int, 0.0),
    # 4. BSP_DRAIN_PERIOD_MS - voulu ~5 (1 TDMA frame)
    ("BSP_DRAIN_PERIOD_MS", 5, "ms",
     "hw/arm/calypso/calypso_bsp.c",
     r"#define\s+BSP_DRAIN_PERIOD_MS\s+(\d+)",
     int, 20.0), # ±20% acceptable
    # 5. WT_DELAY_NS - voulu 25_000_000 ns (spec ISO SIM)
    #    Mais on accepte 2_000_000 ns (emulation rapide, documenté)
    ("WT_DELAY_NS", 25_000_000, "ns",
     "hw/arm/calypso/calypso_sim.c",
     r"#define\s+WT_DELAY_NS\s+(\d+)",
     int, 95.0), # connu off by 12× pour speed
    # 6. PCB_TICK_TDMA_US - voulu 4615 µs
    ("PCB_TICK_TDMA_US", 4615, "µs",
     "hw/arm/calypso/calypso_full_pcb.c",
     r"#define\s+PCB_TICK_TDMA_US\s+(\d+)",
     int, 0.1),
    # 7. PCB_TICK_TINTO_US - voulu 4615 µs
    ("PCB_TICK_TINTO_US", 4615, "µs",
     "hw/arm/calypso/calypso_full_pcb.c",

```

```

    r"#define\s+PCB_TICK_TINTO_US\s+(\d+)",
    int, 0.1),
# 8. PCB_TICK_KICK_US - voulu 5000 µs (legacy 200 Hz)
("PCB_TICK_KICK_US", 5000, "µs",
 "hw/arm/calypso/calypso_full_pcb.c",
 r"#define\s+PCB_TICK_KICK_US\s+(\d+)",
 int, 0.0),
# 9. PCB_TICK_FRAME_IRQ_US - voulu 1000 µs (existant)
("PCB_TICK_FRAME_IRQ_US", 1000, "µs",
 "hw/arm/calypso/calypso_full_pcb.c",
 r"#define\s+PCB_TICK_FRAME_IRQ_US\s+(\d+)",
 int, 0.0),
]

@pytest.mark.timer_audit
@pytest.mark.osmocom_clock
@pytest.mark.parametrize("param,voulu,units,src,regex,parse,tol",
                          AUDIT_PARAMS,
                          ids=[p[0] for p in AUDIT_PARAMS])
def test_timer_physical_param(param, voulu, units, src, regex, parse, tol):
    """Vérifie que chaque param timer dans le code matche la valeur voulue
    par l'équation physique HW (tolérance %)."""
    path = f"{QEMU_SRC}/{src}"
    content = _read(path)
    assert content, f"can't read {path}"
    m = re.search(regex, content)
    assert m, f"can't find {param} in {path} (regex={regex})"
    actual = parse(m.group(1))
    pct_diff = abs(actual - voulu) / voulu * 100.0 if voulu else 0.0
    msg = (f"{param}: voulu={voulu} {units}, actual={actual} {units}, "
           f"Δ={pct_diff:+.3f}% (tol={tol}%)")
    print(f"  AUDIT  {msg}")
    assert pct_diff <= tol, f"OFF SPEC : {msg}"

# === Mesures runtime depuis logs =====

def _parse_timestamps(log_content: str, pattern: str):
    """Retourne list[(unix_ts, wall_offset)] pour chaque ligne matching pattern."""
    out = []
    for line in log_content.splitlines():

```

```

        if not re.search(pattern, line):
            continue
        # Format des logs : `

```

```

p_med = median(periods)
p_mean = mean(periods)
p_std = stdev(periods) if len(periods) > 1 else 0.0
pct = abs(p_med - expected_ms) / expected_ms * 100.0
msg = (f"{name} ({desc}): expected={expected_ms} ms, "
      f"median={p_med:.2f} ms, mean={p_mean:.2f} ms, "
      f"std={p_std:.2f} ms, Δ_med={pct:+.1f}% (N={len(periods)})")
print(f"  RUNTIME {msg}")
# Tolérance large (50%) : sous icount=auto le rate dévie
assert pct < 50.0, f"OFF EXPECTED : {msg}"

# === Graphes mermaid pour les périodes =====

@pytest.mark.timer_graph
def test_timer_temporal_graphs_emit_mermaid():
    """Génère un fichier mermaid xychart pour chaque timer instrumenté.
    Output : /tmp/timer_graphs.mmd contient les xycharts à coller dans un
    rapport markdown."""
    out_lines = ["# Calypso Timer temporal graphs",
                 "",
                 "Auto-generated by `test_timer_temporal_graphs_emit_mermaid`. ",
                 "Une xychart par timer : Y = période entre 2 events (ms), "
                 "X = numéro d'event.",
                 ""]
    found_any = False
    for name, log_path, pattern, expected_ms, desc in TIMER_RUNTIME_PROBES:
        log = _read(log_path)
        if not log:
            continue
        ts = _parse_timestamps(log, pattern)
        if len(ts) < 10:
            continue
        periods = _periods_ms(ts)
        # Sample down to ~50 points max for mermaid render
        N = min(50, len(periods))
        step = max(1, len(periods) // N)
        sampled = periods[::step][:N]
        if not sampled:
            continue
        found_any = True

```

```

out_lines.extend([
    f"## {name} - {desc}",
    f"Expected: {expected_ms} ms ; samples: {len( periods )} (showing {len( sampled )})"
    "",
    "`mermaid",
    "xychart-beta",
    f' title "{name} period (ms) vs event #"',
    f' x-axis "event #" 1 --> {len( sampled )}',
    f' y-axis "period (ms)" 0 --> {max( sampled ) * 1.2:.1f}',
    f' line [{", ".join(f"{p:.2f}" for p in sampled)}]',
    "`",
    "",
])
if not found_any:
    pytest.skip("aucun timer log dispo pour générer graph")
out = "\n".join(out_lines)
Path( GRAPHS_OUT ).write_text( out )
print( f"\n emitted {Path( GRAPHS_OUT )} ({len( out )} chars, "
        f"{out.count( 'xychart-beta' )} graphs)" )

```

`@pytest.mark.timer_graph`

```

def test_timer_drift_summary_table():
    """Génère une table mermaid récap drift entre tous les timers."""
    rows = []
    for name, log_path, pattern, expected_ms, desc in TIMER_RUNTIME_PROBES:
        log = _read( log_path )
        if not log:
            rows.append( (name, "-", expected_ms, "no log") )
            continue
        ts = _parse_timestamps( log, pattern )
        if len( ts ) < 20:
            rows.append( (name, "-", expected_ms, f"only {len( ts )} samples") )
            continue
        periods = [p for p in _periods_ms( ts ) if p < expected_ms * 10]
        if not periods:
            rows.append( (name, "-", expected_ms, "no clean samples") )
            continue
        med = median( periods )
        pct = (med - expected_ms) / expected_ms * 100.0
        status = " " if abs( pct ) < 10 else " " if abs( pct ) < 30 else " "
        rows.append( (name, f"{med:.2f} ms", f"{expected_ms} ms",

```

```

        f"{status} Δ={pct:+.1f}%"))
print("\n  Timer drift summary :")
print(f"   {'name':<20} {'median':<12} {'expected':<12} {'status'}")
for r in rows:
    print(f"   {r[0]:<20} {r[1]:<12} {r[2]:<12} {r[3]}")

```

0.3.68 /opt/GSM/qosmo-overlay/tmux_modules/_commun.sh

5027 octets, 113 lignes → 113 lignes

```

C_RADIO=colour81
C_COEUR=colour114
C_MS1=colour179
C_MS2=colour215
C_VOIX=colour176
SUIVRE="tail -n 100 -F"
# [2026-10-19] Couleurs a l'affichage (tools/qosmo-couleurs.py) : tags [trx] [sim]
# [c54x]..., fn=, hex, erreurs en rouge, trace asm (adresse, octets, PC, symbole),
# mailbox (ARM>WR ...). Rien dans les fichiers. QOSMO_COULEURS=0 coupe.
_couleurs() {
    local p="${QOSMO_COULEURS_PY:-${QEMU_TREE:-.}/tools/qosmo-couleurs.py}"
    [ "${QOSMO_COULEURS:-1}" != 0 ] && [ -f "$p" ] && printf " | python3 -u '%s' --force" "$p"
}
: "${OSMO_LOG_DIR:=/var/log/osmocom}"

_paint() {
    [ -n "${2:-}" ] || return 0
    tmux select-pane -t "$TMUX_SESSION:$1" -P "fg=$2" 2>/dev/null || true
}

_w() {
    tmux new-window -t "$TMUX_SESSION" -n "$1" "$3" 2>/dev/null \
        && tmux select-pane -t "$TMUX_SESSION:$1" -T "$2" 2>/dev/null \
        && _paint "$1" "${4:-}"
}

_split() {
    tmux split-window -t "$TMUX_SESSION:$1" "$3" 2>/dev/null \
        && tmux select-pane -t "$TMUX_SESSION:$1" -T "$2" 2>/dev/null \
        && _paint "$1" "${4:-}" \
        && tmux select-layout -t "$TMUX_SESSION:$1" tiled 2>/dev/null
}

```

```

}

_tail() {
    printf "%s %s 2>/dev/null | stdbuf -oL tr -d '\\\\007'%s || sleep infinity" \
        "$SUIVRE" "$*" "$(_couleurs)"
}

# [2026-10-19] Pane de la trace asm (ASSEMBLY_LOGS=1 / QEMU_ASM) : PAS un tail -F,
# le fichier grossit jusqu'a 150 Mo/s avec exec,nochain et un tail -F a des secondes
# de retard (et gonfle le scrollbar). On rafraichit les 60 dernieres lignes deux
# fois par seconde ; en --dsp le fichier est celui de c54x_exe (/tmp/c54x-pont).
_asm_watch() {
    printf "watch -t -n 0.5 --color \"for f in /tmp/c54x-pont/qemu-asm.log '%s/qemu-asm.log'
        \"${LOG_DIR}\" \"$_couleurs\""
}

_fenetre_coeur() {
    local O="${OSMO_LOG_DIR:-/var/log/osmocom}"
    _w    coeur "MSC | commutation circuit - appels, SMS, authentication" \
        "$(_tail "'$O/osmo-msc.log')" "$C_COEUR"
    _split coeur "BSC | controleur de stations - allocation des canaux" \
        "$(_tail "'$O/osmo-bsc.log')" "$C_COEUR"
    _split coeur "HLR | registre des abonnees - IMSI, cles, souscriptions" \
        "$(_tail "'$O/osmo-hlr.log')" "$C_COEUR"
    _split coeur "MGW + STP | media MGCP/RTP et routage SS7" \
        "$(_tail "'$O/osmo-mgw.log' '$O/osmo-stp.log')" "$C_COEUR"
}

_fenetre_voix() {
    local L="${LOG_DIR}" O="${OSMO_LOG_DIR:-/var/log/osmocom}"
    _w    voix "gapk | transcodage GSM-FR <-> PCM (actif pendant un appel)" \
        "$(_tail "'$L/gapk-auto.log')" "$C_VOIX"
    _split voix "SIP | passerelle MNCC <-> SIP vers Asterisk" \
        "$(_tail "'$O/osmo-sip-connector.log')" "$C_VOIX"
}

_fenetre_shell() {
    local aide=""
    local l; for l in "$@"; do aide="${aide} ${l}\\n"; done
    _w shell "shell | depot qosmo" \
        "cd '${QEMU_TREE:-.}' && printf '\n %s\n %s\n${aide}\\n' \
        'etat : ./run.sh --status' 'arret : ./run.sh --stop' \

```

```

    && exec ${SHELL:-/bin/bash}"
}

_tmux_env_reprop() {
    command -v tmux >/dev/null 2>&1 || return 0
    tmux show-environment -g 2>/dev/null \
        | sed -n 's/^\(CALYPSO_[A-Za-z0-9_]*\)=.*/\1/p' \
        | while IFS= read -r _v; do
            [ -n "$_v" ] && tmux set-environment -g -u "$_v" 2>/dev/null
        done
    unset _v
}

_tmux_style() {
    local s="$TMUX_SESSION"
    tmux set-environment -t "$s" LANG "${LANG:-C.UTF-8}" 2>/dev/null
    tmux set-environment -t "$s" LC_ALL "${LC_ALL:-C.UTF-8}" 2>/dev/null
    tmux set -t "$s" -g mouse on 2>/dev/null
    tmux set -t "$s" -g base-index 1 2>/dev/null
    tmux set -t "$s" -g pane-base-index 1 2>/dev/null
    tmux set -t "$s" -g history-limit 20000 2>/dev/null
    tmux set -t "$s" -g renumber-windows on 2>/dev/null
    tmux set -t "$s" -g bell-action none 2>/dev/null
    tmux set -t "$s" -g visual-bell off 2>/dev/null
    tmux set -t "$s" -g monitor-bell off 2>/dev/null
    tmux set -t "$s" -g status-style "bg=colour234,fg=colour250" 2>/dev/null
    tmux set -t "$s" -g status-left-length 40 2>/dev/null
    tmux set -t "$s" -g status-left \
        "#[bg=colour31,fg=colour15,bold] calypso #[default] " 2>/dev/null
    tmux set -t "$s" -g status-right-length 76 2>/dev/null
    tmux set -t "$s" -g status-right \
        "#[fg=colour215]gr-gsm#[fg=colour244] · #[fg=colour114]pont ARFCN ${PONT_ARFCN:-?} B" 2>/dev/null
    tmux set -t "$s" -g window-status-format " #I #W " 2>/dev/null
    tmux set -t "$s" -g window-status-current-format \
        "#[bg=colour31,fg=colour15,bold] #I #W #[default]" 2>/dev/null
    tmux set -t "$s" -g pane-border-style "fg=colour238" 2>/dev/null
    tmux set -t "$s" -g pane-active-border-style "fg=colour31" 2>/dev/null
    tmux set -t "$s" -g pane-border-status top 2>/dev/null
    tmux set -t "$s" -g pane-border-format "#{pane_title} " 2>/dev/null
}

```

0.3.69 /opt/GSM/qosmo-overlay/tmux_modules/bench.sh

2378 octets, 43 lignes → 43 lignes

```
TMUX_FENETRE_PREMIERE=radio
TMUX_RESUME="6 fenêtres : radio · coeur · ms1 · ms2 (side-car) · voix · shell"

tmux_layout_premiere() {
    printf "tail -n 200 -F '%s/qemu.log' 2>/dev/null | stdbuf -oL tr -d '\\\\007'%s || sleep
        "${LOG_DIR}" "${_couleurs}"
}

tmux_layout() {
    local L="${LOG_DIR}"
    tmux select-pane -t "$TMUX_SESSION:radio" -T \
        "QEMU | baseband emule (ARM7, couche 1 gr-gsm) - traces du modele" 2>/dev/null
    _paint radio "$C_RADIO"
    _split radio "osmocon | lien serie firmware - romload puis transport L1CTL" \
        "$(_tail "'$L/osmocon.log'")" "$C_RADIO"
    _split radio "pont | transceiver du BTS#0 - bursts TRXD 5700-5702 <-> QEMU 4730/4731" \
        "$(_tail "'/dev/shm/pont.log'")" "$C_RADIO"
    # [2026-10-19] trace asm de l'ARM quand elle est demandee (--asm / --assembly-logs)
    if { [ -n "${QEMU_ASM:-}" ] && [ "${QEMU_ASM}" != 0 ]; } || [ "${ASSEMBLY_LOGS:-0}" = 1 ]
        _split radio "asm | blocs ARM (-d ${QEMU_ASM_FLAGS:-in_asm}), 60 dernieres lignes, 2
            "$(_asm_watch)" "$C_RADIO"
    fi
    _fenetre_coeur
    # [2026-10-24] MS1_LOG (osmo-operator, 09-mobile-cfg-tmux.sh) : rvinterf.log en Magnetit
    _w      ms1 "MS#1 | ${MS1_LOG_NOM:-mobile} du telephone emule - camp, LU, appels" \
        "$(_tail "'${MS1_LOG:-$L/mobile.log}'")" "$C_MS1"
    _split ms1 "BTS#0 | sa station de base - ARFCN ${PONT_ARFCN:-514}, unit-id 6001" \
        "$(_tail "'$L/bts.log'")" "$C_MS1"
    _split ms1 "GSMTAP | capture pcap - a ouvrir dans Wireshark" \
        "$(_tail "'$L/gsmtap.log'")" "$C_MS1"
    _w      ms2 "MS#2 | second abonne - campe sur le BTS#1, appelle le MS#1" \
        "$(_tail "'$L/sidecar-mobile.log'")" "$C_MS2"
    _split ms2 "trxcon | sa couche 1 - TRXD <-> L1CTL (remplace le firmware)" \
        "$(_tail "'$L/sidecar-trxcon.log'")" "$C_MS2"
    _split ms2 "BTS#1 | seconde station - ARFCN ${SC_ARFCN:-516}, unit-id ${SC_UNIT_ID:-6002}
        "$(_tail "'$L/sidecar-bts.log'")" "$C_MS2"
    _split ms2 "fake_trx | commutateur de bursts UDP ${SC_TRX_PORT:-5720} <-> ${SC_BB_PORT:-}
        "$(_tail "'$L/sidecar-faketrx.log'")" "$C_MS2"
    _fenetre_voix
```

```

_fenetre_shell \
    "MS#1 : telnet 127.0.0.1 4247 puis show ms" \
    "MS#2 : telnet 127.0.0.1 ${SC_MOBILE_VTY_PORT:-4248} puis call 1 10001"
}

```

0.3.70 /opt/GSM/qosmo-overlay/tools/demo-layer1.py

6169 octets, 157 lignes → 157 lignes

```

# demo-layer1.py - « demo » : montrer qu'on a la main sur le firmware layer1 (osmocom-bb)
# qui tourne dans QEMU, depuis la console telnet (tools/gdb-telnet.py, port 44444).
#
# demo          tout : arrete l'ARM, etat, une trame TDMA au point d'arret, un printf
#                dans le firmware (visible dans osmocon.log), puis l'ARM repart
# demo etat      seulement l'etat (pc, pile, fn, cellule, boite aux lettres DSP) ; reste a
# demo trame      tourne jusqu'a la prochaine trame TDMA (tdma_sched_execute) ; reste arrete
# demo hello      call printf(...) dans le firmware -> console sercomm (osmocon.log) ; reste a
# demo go         l'ARM repart (continue &)
#
# Entre deux etapes : 4 s pour lire (set $demo_pause = N pour changer, 0 = enchaîner).
# Si l'ARM tourne, « demo » l'arrete d'abord (interrupt) et enchaîne au prochain événement
# d'arret (gdb.events.stop) : rien n'est lu tant que la cible n'est pas arretée.
# Les lectures passent par les commandes du panneau cmd.gdb (fn, l1, tasks), charge avant.
import gdb, time

def ex(c):
    try:
        return gdb.execute(c, to_string=True)
    except gdb.error as e:
        return " (%s : %s)\n" % (c, e)

def out(t):
    gdb.write(t if t.endswith("\n") else t + "\n")
    gdb.flush()

def en_marche():
    try:
        return any(t.is_running() for t in gdb.selected_inferior().threads())

```

```

except gdb.error:
    return False

def pause_etape():
    # entre deux etapes, le temps de lire : 4 s par default, « set $demo_pause = 2 » pour char
    p = gdb.convenience_variable("demo_pause")
    t = float(p) if p is not None else 4.0
    if t > 0:
        out("[demo]          ... (suite dans %.0f s)" % t)
        time.sleep(t)

def val(expr):
    try:
        return int(gdb.parse_and_eval(expr))
    except gdb.error:
        return None

def etat():
    fr = gdb.newest_frame()
    sal = fr.find_sal()
    ou = "%s:%d" % (sal.symtab.filename, sal.line) if sal.symtab else "?"
    out("[demo] ===== 1. L'ARM EST ARRETE : pc=0x%08x dans %s (%s)" % (fr.pc(), fr.name() or
    out(ex("bt 6"))
    out(ex("x/3i $pc"))
    pause_etape()
    out("[demo] ===== 2. CE QUE LA COUCHE 1 SAIT (symboles l1s du firmware) :")
    out(ex("fn"))
    out(ex("l1"))
    pause_etape()
    out("[demo] ===== 3. LA BOITE AUX LETTRES ARM<->DSP, lue par l'ARM a 0xffd00000")
    out("[demo]          (= « d@ 0800 8 » dans harvard.sh : meme memoire, /dev/shm/calypso_api
    out(ex("x/8hx 0xffd00000"))
    out(ex("tasks"))
    pause_etape()

def trame():
    avant = val("l1s.current_time.fn")
    out("[demo] ===== 4. ON RELANCE JUSQU'A LA PROCHAINE TRAME TDMA (tbreak tdma_sched_execu

```

```

t0 = time.time()
ex("tbreak tdma_sched_execute")
gdb.execute("continue")
apres = val("l1s.current_time.fn")
d = (apres - avant) if (apres is not None and avant is not None) else "?"
out("[demo]          arrete dans tdma_sched_execute : fn=%s (+%s) en %.0f ms  (une trame =
      % (apres, d, 1000 * (time.time() - t0)))
out(ex("bt 4"))
pause_etape()

```

```

def hello():
    # Pas de malloc dans le firmware : gdb ne peut pas y poser une chaine litterale. On l'ecrit
    # nous-memes juste au-dessus du bss (_end + 0x100) : zone libre (pas de tas, les piles sont
    # tout en haut de la RAM, TOP_OF_RAM 0x83fff0 dans board/compal/macros.S), puis puts() de
    fn = val("l1s.current_time.fn")
    out("[demo] ===== 5. ON FAIT PARLER LE FIRMWARE : la chaine ecrite en RAM libre, puis ca
    adr = val("(unsigned long)&_end")
    if adr is None:
        out("[demo]          (_end introuvable : ELF sans symboles ?)"); return
    adr = (adr + 0x100) & ~0xf
    # Couleurs ANSI dans la chaine elle-meme : osmocon recopie la console sercomm octet pour
    # (hdlc_console_cb -> write(1, ...)), donc « tail -f osmocon.log » sous terminal l'affiche
    # hello (vert) a (blanc) 40c3 (violet) : « hello a 40c3 », ecrit en RAM puis sorti par p
    texte = ("\033[1;32mhello\033[0m \033[1;37ma\033[0m \033[1;35m40c3\033[0m "
            "\033[36m(0x%x, fn=%s)\033[0m\n" % (adr, fn))
    msg = texte.encode() + b"\0"
    gdb.selected_inferior().write_memory(adr, msg)
    out("[demo]          %d octets ecrits a 0x%08x : %s" % (len(msg), adr, ex("x/s 0x%x" % adr)
    out("[demo]          soit, rendu : " + texte.rstrip("\n"))
    out(ex("call (int)puts((const char *)0x%x)" % adr))
    out("[demo]          (regarder : tail -f /tmp/c54x-pont/osmocon.log)")
    pause_etape()

```

```

def fin():
    out(ex("continue &"))
    out("[demo] ===== FIN : l'ARM est RELANCE et tourne a nouveau, rien de casse (Ctrl-C ou s
    try:
        time.sleep(1.0)
        out("[demo]          1 s plus tard, l'ARM a bien repris : %s" % ("il tourne (continue &
    except Exception:

```

```

pass

def corps(arg):
    try:
        if arg in ("etat", "tout"):
            etat()
        if arg in ("trame", "tout"):
            trame()
        if arg in ("hello", "tout"):
            hello()
        if arg in ("go", "tout"):
            fin()
        if arg not in ("etat", "trame", "hello", "go", "tout"):
            out("usage : demo [tout|etat|trame|hello|go]")
    except Exception as e:
        out("[demo] erreur : %s" % e)

class Demo(gdb.Command):
    """demo [tout|etat|trame|hello|go] - la main sur le firmware layer1 : arret, etat, une t

    def __init__(self):
        super().__init__("demo", gdb.COMMAND_USER)

    def invoke(self, arg, from_tty):
        arg = arg.strip() or "tout"
        if en_marche():
            out("[demo] l'ARM tourne : interrupt ; la suite des que la cible est arretee...")

            def on_stop(ev):
                gdb.events.stop.disconnect(on_stop)
                gdb.post_event(lambda: corps(arg))
                gdb.events.stop.connect(on_stop)
                gdb.execute("interrupt")
            else:
                corps(arg)

Demo()
gdb.write("[demo-layer1] tape demo (ou demo etat | trame | hello | go)\n")

```

0.3.71 /opt/GSM/qosmo-overlay/tools/gdb-telnet.py

8356 octets, 204 lignes → 204 lignes

```
#!/usr/bin/env python3
"""
gdb-telnet.py - console gdb (ARM du Calypso emule) servie en telnet, NON BLOQUANTE.

    telnet localhost 44444

Ce qu'on obtient : un gdb-multiarch attache au gdbstub de QEMU (-gdb tcp::1234),
la cible EN MARCHE (`continue &`), et le prompt (gdb) disponible tout de suite.
    Ctrl-C          arrete l'ARM (SIGINT -> gdb interrompt la cible), on inspecte
    continue &      repart, sans bloquer le prompt
    quit / Ctrl-]    la session se ferme, gdb est tue, QEMU reprend l'ARM

Pourquoi un serveur maison et pas `socat ... EXEC:gdb,pty` : telnet reste en
« ligne par ligne » face a un serveur muet, et Ctrl-C part alors en sequence
IAC IP que gdb ne comprend pas. Ici on negocie ECHO+SGA (mode caractere) : le
client envoie 0x03 brut, le pty (isig) le transforme en SIGINT pour gdb. Les
sequences IAC restantes sont filtrees, IAC IP/BRK deviennent un 0x03.

Une session a la fois : le gdbstub de QEMU n'accepte qu'un client. Tant que
personne n'est connecte, aucun gdb ne tourne et l'ARM n'est jamais touche.
"""
import argparse, os, pty, select, signal, socket, sys, time

IAC, DONT, DO, WONT, WILL, SB, SE = 255, 254, 253, 252, 251, 250, 240
IP, BRK = 244, 243
ECHO, SGA = 1, 3

# Ctrl-C : quand la cible tourne EN ARRIERE-PLAN (continue &), un SIGINT au
# prompt gdb ne l'arrete pas - readline repond juste « Quit ». Ce qui l'arrete,
# c'est la commande `interrupt`. On efface la ligne en cours (Ctrl-U) puis on
# l'envoie ; gdb affiche « Program received signal SIGINT » et la main revient.
CTRL_C = b'\x15interrupt\n'

def log(msg):
    sys.stderr.write(time.strftime('%H:%M:%S ') + msg + '\n'); sys.stderr.flush()

def filter_telnet(data, state):
```

```

"""Retire les commandes telnet, rend les octets destines a gdb."""
out = bytearray()
buf = state['pending'] + data
i = 0
while i < len(buf):
    b = buf[i]
    if b == IAC:
        if i + 1 >= len(buf):
            break # commande coupee : on attend la suite
        c = buf[i + 1]
        if c == IAC:
            out.append(IAC); i += 2; continue
        if c in (IP, BRK):
            out += CTRL_C; i += 2; continue # Ctrl-C telnet -> interrupt
        if c in (DO, DONT, WILL, WONT):
            if i + 2 >= len(buf):
                break
            i += 3; continue
        if c == SB:
            j = buf.find(bytes([IAC, SE]), i)
            if j < 0:
                break
            i = j + 2; continue
        i += 2; continue
    if b == 13:
        # Entree telnet : CR LF ou CR NUL -> CR
        out.append(13)
        i += 2 if i + 1 < len(buf) and buf[i + 1] in (0, 10) else 1
        continue
    if b == 0:
        i += 1; continue
    if b == 3:
        # Ctrl-C brut (mode caractere)
        out += CTRL_C; i += 1; continue
    out.append(b); i += 1
state['pending'] = bytes(buf[i:])
return bytes(out)

def session(conn, args):
    conn.sendall(bytes([IAC, WILL, ECHO, IAC, WILL, SGA, IAC, DO, SGA]))
    banner = (f"\r\n[gdb-telnet] gdb -> QEMU gdbstub :{args.stub} elf={args.elf}\r\n"
              "[gdb-telnet] cible EN MARCHE (continue &). Ctrl-C = stop, go = repart, "
              "quit = ferme (QEMU reprend)\r\n")

```

```

        "[gdb-telnet] panneau osmocom charge : tape  help_osmo\r\n\r\n")
conn.sendall(banner.encode())
pid, fd = pty.fork()
if pid == 0:
    os.environ['TERM'] = 'vt100'
    # readline sans bracketed-paste : sinon chaque prompt est encadre de
    # sequences ESC[?2004h/l illisibles dans un telnet nu.
    rc = '/tmp/gdb-telnet.inputrc'
    try:
        with open(rc, 'w') as f:
            f.write('set enable-bracketed-paste off\nset bell-style none\n')
        os.environ['INPUTRC'] = rc
    except OSError:
        pass
    cmd = [args.gdb, '-q', '-nx',
           '-iex', 'set pagination off', '-iex', 'set confirm off',
           '-iex', 'set height 0', '-iex', 'set width 0', '-iex', 'set style enabled off',
           '-ex', 'set architecture armv5te',
           '-ex', f'target remote 127.0.0.1:{args.stub}',
           '-ex', 'continue &']
    # panneau osmocom (tools/cmd.gdb, genere par gdb_cmd.sh) : source auto
    tools = os.path.dirname(os.path.abspath(__file__))
    panel = os.path.join(tools, 'cmd.gdb')
    if os.path.isfile(panel):
        # [2026-10-19] cmd.gdb cherche demo-layer1.py d'abord ici : le chemin
        # fige a sa generation peut etre celui d'un rootfs de build.
        os.environ['OSMO_GDB_TOOLS'] = tools
        cmd += ['-x', panel]
    if args.elf:
        cmd.append(args.elf)
    try:
        os.execvp(cmd[0], cmd)
    except OSError as e:
        os.write(2, f"exec {cmd[0]} : {e}\n".encode()); os._exit(127)
state = {'pending': b''}
try:
    while True:
        r, _, _ = select.select([conn, fd], [], [], 1.0)
        if conn in r:
            try:
                data = conn.recv(4096)
            except OSError:

```

```

        break
    if not data:
        break
    if b'\x03' in data or bytes([IAC, IP]) in data or bytes([IAC, BRK]) in data:
        # vrai SIGINT a gdb : coupe une boucle (trace_*, every) ou une
        # execution en avant-plan ; le `interrupt` qui suit arrete
        # la cible si elle tournait en arriere-plan (continue &).
        try:
            os.killpg(os.getpgid(pid), signal.SIGINT)
        except OSError:
            pass
    out = filter_telnet(data, state)
    if out:
        os.write(fd, out)
if fd in r:
    try:
        data = os.read(fd, 4096)
    except OSError:
        break
    if not data:
        break
    conn.sendall(data)
try:
    if os.waitpid(pid, os.WNOHANG)[0] == pid:
        pid = 0; break
except ChildProcessError:
    pid = 0; break
finally:
    if pid:
        # detach d'abord : la cible repart proprement meme si elle etait arretee
        try:
            os.write(fd, b'\x03')
            time.sleep(0.2)
            os.write(fd, b'detach\nquit\n')
            for _ in range(10):
                if os.waitpid(pid, os.WNOHANG)[0] == pid:
                    pid = 0; break
                time.sleep(0.1)
        except OSError:
            pass
    if pid:
        try:

```

```

        os.kill(pid, signal.SIGKILL); os.waitpid(pid, 0)
    except OSError:
        pass

    try:
        os.close(fd)
    except OSError:
        pass

def main():
    ap = argparse.ArgumentParser(description=__doc__, formatter_class=argparse.RawDescriptionHelpFormatter)
    ap.add_argument('--port', type=int, default=44444, help='port telnet (default 44444)')
    ap.add_argument('--bind', default='127.0.0.1', help="adresse d'ecoute (default 127.0.0.1)")
    ap.add_argument('--stub', type=int, default=1234, help='port du gdbstub QEMU (default 1234)')
    ap.add_argument('--elf', default=os.environ.get('CALYPSO_GDB_ELF', ''), help='ELF du firmware')
    ap.add_argument('--gdb', default='gdb-multiarch', help='gdb')
    args = ap.parse_args()
    signal.signal(signal.SIGCHLD, signal.SIG_DFL)
    srv = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    srv.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    srv.bind((args.bind, args.port)); srv.listen(2)
    log(f"ecoute telnet {args.bind}:{args.port} -> gdbstub :{args.stub} ({args.gdb}, elf={args.elf})")
    while True:
        conn, peer = srv.accept()
        log(f"session {peer[0]}:{peer[1]}")
        try:
            session(conn, args)
        except Exception as e:
            log(f"session : {e}")
        finally:
            try:
                conn.close()
            except OSError:
                pass
            log("session fermee, gdb termine, QEMU reprend")

if __name__ == '__main__':
    main()

```

0.3.72 /opt/GSM/qosmo-overlay/tools/gdb_cmd.sh

26005 octets, 791 lignes → 781 lignes (2 groupes compactés)

```
#!/bin/bash
# =====
#  gdb_cmd.sh - genere cmd.gdb : le panneau de commandes gdb du banc osmocom
# =====
#  ./gdb_cmd.sh          ecrit tools/cmd.gdb (a cote de ce script)
#  ./gdb_cmd.sh --install idem + ajoute `source ../cmd.gdb` a ~/.gdbinit
#  ./gdb_cmd.sh --out FICH ecrit ailleurs
#
#  cmd.gdb est source AUTOMATIQUEMENT par la console telnet (tools/gdb-telnet.py,
#  `telnet localhost 44444`) et par gdb -x. Dedans : `help_osmo` liste tout.
#  Les commandes lisent le firmware osmocom-bb PAR SES SYMBOLES (l1s, dsp_api,
#  fbs, rxnb, afc_state...) : il faut l'ELF non strippe (layer1.highram.elf),
#  c'est celui que -kernel charge et que le lanceur passe a gdb.
#  -----
set -eu
HERE="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
OUT="$HERE/cmd.gdb"; INSTALL=0
while [ $# -gt 0 ]; do
    case "$1" in
        --install) INSTALL=1 ;;
        --out) OUT="$2"; shift ;;
        -h|--help) sed -n '2,15p' "$0"; exit 0 ;;
        *) echo "option inconnue : $1" >&2; exit 2 ;;
    esac; shift
done

cat > "$OUT" <<'GDB'
# cmd.gdb - panneau osmocom pour gdb (genere par tools/gdb_cmd.sh, ne pas editer)
# Tape `help_osmo` pour la liste. Tout est en francais, court, et lit les
# symboles du firmware osmocom-bb (layer1.highram.elf).
set pagination off
set confirm off
set print pretty on
set height 0
set width 0
set $API = 0xffd00000

#  constantes dsp_api / l1_environment.h
set $B_BLUD = 15
```

```

set $B_SCH_CRC = 8
set $B_FIRE0 = 5
set $B_FIRE1 = 6
set $B_TASK_ABORT = 15

# =====
# EXECUTION
# =====

define go
    continue &
end
document go
go - reprend la cible SANS bloquer le prompt (continue &). Ctrl-C = stop.
end

define stop
    interrupt
end
document stop
stop - arrete l'ARM (equivalent de Ctrl-C). Puis regs, bt10, fn, ndb...
end

define regs
    info registers
end
document regs
regs - tous les registres ARM (cible arretee).
end

define pcx
    x/8i $pc
end
document pcx
pcx - 8 instructions a partir de $pc.
end

define bt10
    bt 10
end
document bt10
bt10 - pile d'appels, 10 niveaux.
end

```

```

define si1
    stepi
    x/1i $pc
end
document si1
si1 - une instruction, affiche la suivante.
end

define ni1
    nexti
    x/1i $pc
end
document ni1
ni1 - une instruction sans entrer dans les appels.
end

define frame_step
    tbreak tdma_sched_execute
    continue
    fn
end
document frame_step
frame_step - laisse tourner jusqu'a la PROCHAINE trame TDMA (tdma_sched_execute) puis s'arrete.
end

# =====
#  TEMPS / ORDONNANCEUR
# =====

define fn
    printf "fn=%u  (t1=%u t2=%u t3=%u)  next fn=%u  tpu_offset=%u corr=%d\n", l1s.current_time,
end
document fn
fn - numero de trame courant de la L1 (l1s.current_time), suivant, et offset TPU.
end

define l1
    fn
    printf "cell    : arfcn=%u bsic=%u  fn_offset=%u  time_align=%u\n", l1s.serving_cell.arfcn,
    printf "modes   : fb.mode=%u  pm.mode=%u  dedicated.type=%d tn=%u tsc=%u h=%u\n", l1s.fb.mode,
    printf "radio   : ta=%d tx_power=%u tch_mode=%u audio=%u  compl=0x%08x\n", l1s.ta, l1s.tx_power,
    pages
end

```

```

document l1
l1 - resume de l'etat L1 : temps, cellule servante, modes FB/PM/dedie, TA, puissance, pages L1
end

define cell
    p l1s.serving_cell
end
document cell
cell - la cellule servante (arfcn, bsic, fn_offset, time_alignment, rxlev...).
end

define neigh
    p l1s.neigh_cell
    printf "neigh_pm : n=%u running=%u pos=%u\n", l1s.neigh_pm.n, l1s.neigh_pm.running, l1s.neigh_pm.pos;
end
document neigh
neigh - les 6 voisines connues de la L1 et l'etat des mesures voisines.
end

define sched
    p l1s.tdma_sched
end
document sched
sched - l'ordonnanceur TDMA (les items programmes pour les prochaines trames).
end

define mframe
    p l1s.mframe_sched
end
document mframe
mframe - l'ordonnanceur multitrame (BCCH/CCCH/SDCCH/TCH actifs).
end

define compl
    printf "scheduled_compl = 0x%08x  (bit0 FB, 1 RACH, 2 TX_NB, 3 TCH, 4 TX_SB?, voir l1s.compl);
end
document compl
compl - masque des completions L1 en attente (l1s.scheduled_compl).
end

define arfcn
    printf "rf_arfcn = %u\n", rf_arfcn

```

```

end
document arfcn
arfcn - ARFCN courant du frontal radio.
end

define afc
    p afc_state
end
document afc
afc - etat de l'AFC (boucle de correction de frequence).
end

# =====
#  FB / SB (synchro) - ce que la L1 a vu
# =====
define fbs
    p fbs
end
document fbs
fbs - etat complet de l'acquisition FB/SB (fbs.mon : bsic, time, toa, pm, angle, snr, attempt)
end

define mon
    printf "mon : bsic=%u toa=%d pm=%d angle=%d snr=%u attempt=%d fn_report=%u time fn=%u\n"
end
document mon
mon - le dernier resultat FB/SB tel que l'ARM l'a lu (fbs.mon).
end

# =====
#  MAILBOX ARM <-> DSP (dsp_api : ndb / db_r / db_w)
# =====
define pages
    printf "pages : r_page=%d w_page=%d r_page_used=%d frame_ctr=%d ndb.d_dsp_page=0x%04x\n"
    printf "          db_r=%p db_w=%p ndb=%p param=%p\n", dsp_api.db_r, dsp_api.db_w, dsp_api.ndb, dsp_api.param
end
document pages
pages - pages de la double mise en tampon MCU<->DSP (r_page/w_page vus par l'ARM, d_dsp_page)
end

define err
    set $e = dsp_api.ndb->d_error_status & 0xffff

```

```

printf "d_error_status = 0x%04x", $e
if $e & 0x0008
    printf "    DSP_ERR_DMA_PROG"
end
if $e & 0x0010
    printf "    DSP_ERR_DMA_TASK"
end
printf "\n"
end
document err
err - d_error_status du DSP, decode (8 = DMA_PROG, 0x10 = DMA_TASK ; témoin, pas un blocage)
end

define taskname
    if $arg0 == 0
        printf "-"
    end
    if $arg0 == 1
        printf "PM"
    end
    if $arg0 == 5
        printf "FB"
    end
    if $arg0 == 6
        printf "SB"
    end
    if $arg0 == 10
        printf "RACH"
    end
    if $arg0 == 19
        printf "NBS"
    end
    if $arg0 == 21
        printf "NP"
    end
    if $arg0 == 24
        printf "ALLC"
    end
    if $arg0 == 33
        printf "CHECKSUM"
    end
end
end

```

```

define tasks
    set $td = dsp_api.db_w->d_task_d & 0xffff
    set $tu = dsp_api.db_w->d_task_u & 0xffff
    set $tm = dsp_api.db_w->d_task_md & 0xffff
    set $tr = dsp_api.db_w->d_task_ra & 0xffff
    printf "db_w (ARM->DSP) : task_d=%u(", $td
    taskname $td
    printf ") burst_d=%u task_u=%u(", dsp_api.db_w->d_burst_d & 0xffff, $tu
    taskname $tu
    printf ") burst_u=%u task_md=%u(", dsp_api.db_w->d_burst_u & 0xffff, $tm
    taskname $tm
    printf ") task_ra=%u\n", $tr
    printf "          d_fn=0x%04x ctrl_abb=0x%04x afc=%d(0x%04x) ctrl_system=0x%04x pow
    set $rd = dsp_api.db_r->d_task_d & 0xffff
    printf "db_r (DSP->ARM) : task_d=%u(", $rd
    taskname $rd
    printf ") burst_d=%u task_u=%u burst_u=%u task_md=%u task_ra=%u\n", dsp_api.db_r->d_burst_u
end
document tasks
tasks - les taches programmees par l'ARM (page W) et celles que le DSP a rendues (page R), a
end

define dbw
    p *dsp_api.db_w
end
document dbw
dbw - la page d'ecriture ARM->DSP complete (d_task_*, d_fn, d_ctrl_abb, a_a5fn, d_afc, d_ctrl_abb)
end

define dbr
    p *dsp_api.db_r
end
document dbr
dbr - la page de lecture DSP->ARM complete (taches rendues, a_serv_demod, a_pm, a_sch).
end

define sync
    printf "ndb : d_fb_det=%u d_fb_mode=%u a_sync_demod TOA=%d PM=%d ANGLE=%d SNR=%u\n", dsp
    printf "db_r : a_serv_demod TOA=%d PM=%d(>>3=%d) ANGLE=%d SNR=%u\n", (short)(dsp_api.db_r->a_serv_demod)
end
document sync
sync - resultats de synchro : d_fb_det + a_sync_demod (NDB, chemin FB) et a_serv_demod (page

```

```

end

define sb
    set $s0 = dsp_api.db_r->a_sch[0] & 0xffff
    set $sbw = (dsp_api.db_r->a_sch[3] & 0xffff) | ((dsp_api.db_r->a_sch[4] & 0xffff) << 16)
    printf "a_sch = %04x %04x %04x %04x %04x  ", $s0, dsp_api.db_r->a_sch[1] & 0xffff, dsp_ap
    if $s0 & (1 << $B_SCH_CRC)
        printf "CRC FAUX (B_SCH_CRC)"
    else
        printf "CRC ok"
    end
    if $s0 & (1 << $B_BLUD)
        printf "  BLUD"
    end
    set $t1 = (($sbw >> 23) & 1) | (($sbw >> 7) & 0x1fe) | (($sbw << 9) & 0x600)
    set $t2 = ($sbw >> 18) & 0x1f
    set $t3p = (($sbw >> 24) & 1) | (($sbw >> 15) & 6)
    set $t3 = $t3p * 10 + 1
    printf "\nSB 0x%08x : BSIC=%u  t1=%u t2=%u t3=%u  (fn = 51*26*t1 + 51*((t3-t2+26)%26) + t
end
document sb
sb - decode a_sch comme prim_fbsb.c : CRC, mot SB, BSIC, t1/t2/t3 et fn.
end

define cd
    set $c0 = dsp_api.ndb->a_cd[0] & 0xffff
    printf "a_cd[0]=0x%04x", $c0
    if $c0 & (1 << $B_BLUD)
        printf "  BLUD"
    end
    printf "  fire_crc=%u  biterr=%u\n", ($c0 & ((1 << $B_FIRE1) | (1 << $B_FIRE0))) >> $B_FIRE
    printf "a_cd[1]=0x%04x  data[3..14] = ", dsp_api.ndb->a_cd[1] & 0xffff
    set $i = 3
    while $i < 15
        printf "%04x ", dsp_api.ndb->a_cd[$i] & 0xffff
        set $i = $i + 1
    end
    printf "\n"
end
document cd
cd - le bloc de donnees decode (a_cd : etat BLUD, CRC Fire, erreurs bits, 23 octets du bloc
end

```

```

define pm
    printf "a_pm = %d %d %d    (dBm8 : %d)\n", (short)(dsp_api.db_r->a_pm[0] & 0xffff), (short)
end
document pm
pm - les 3 mesures de puissance rendues par le DSP (page R).
end

define ndb
    err
    sync
    printf "d_dsp_state=0x%04x  bg enable=0x%04x state=0x%04x abort=0x%04x  d_tch_mode=0x%04x
    printf "d_dsp_page=0x%04x  version=%04x/%04x  d_rach=0x%04x\n", dsp_api.ndb->d_dsp_page &
end
document ndb
ndb - les cellules NDB qui comptent : erreur, synchro, etat DSP, taches de fond, page, versio
end

define dsp
    pages
    tasks
    ndb
    sb
end
document dsp
dsp - vue d'ensemble de la mailbox : pages + taches + ndb + sb. La commande a taper d'abord.
end

define apiram
    x/$arg1hx ($API + 2*($arg0 - 0x800))
end
document apiram
apiram OFF N - N mots de la memoire API vus de l'ARM, a partir du mot DSP OFF (0xffd00000 = r
end

define apiw
    set *(unsigned short *)($API + 2*($arg0 - 0x800)) = $arg1
    printf "api[0x%04x] <- 0x%04x\n", $arg0, $arg1
end
document apiw
apiw OFF VAL - ecrit le mot DSP OFF (adresse cote DSP, ex 0x08f8) dans la memoire API partag
end

```

```

define rxnb
    p rxnb
end
document rxnb
rxnb - etat de la reception des bursts normaux (rxnb : meas[4], msg en cours).
end

# =====
#  FORCAGES / FAUX RESULTATS  (ecrivent dans la mailbox : a utiliser cible ARRETEE)
#  =====
define fake_fb
    set dsp_api.ndb->d_fb_det = 1
    set dsp_api.ndb->a_sync_demod[1] = $arg2 *4
        = (0,0) (1,1) (2,2) (3,3)
    printf "FB force : d_fb_det=1 toa=%d pm=%d angle=%d snr=%d\n", $arg0, $arg1, $arg2, $arg3
    sync
end
document fake_fb
fake_fb TOA PM ANGLE SNR - fait croire a l'ARM que le DSP a detecte un FB (ex : fake_fb 48 50 0 0)
end

define fake_sb
    set $t1 = $arg1
    set $t2 = $arg2
    set $t3p = ($arg3 - 1) / 10
    set $w = (($arg0 & 0x3f) << 2) | (($t1 & 1) << 23) | (($t1 & 0x1fe) << 7) | (($t1 & 0x600) << 1)
    set dsp_api.db_r->a_sch[0] = 0
    set dsp_api.db_r->a_sch[3] = $w & 0xffff
    set dsp_api.db_r->a_sch[4] = ($w >> 16) & 0xffff
    printf "SB force : 0x%08x bsic=%u t1=%u t2=%u t3=%u\n", $w, $arg0, $t1, $t2, $arg3
    sb
end
document fake_sb
fake_sb BSIC T1 T2 T3 - compose un mot SB valide (CRC ok) dans a_sch pour que l'ARM decode correctement
end

define sb_force
    set dsp_api.db_r->a_sch[0] = dsp_api.db_r->a_sch[0] & ~(1 << $B_SCH_CRC)
    printf "B_SCH_CRC efface : l'ARM decodera a_sch[3..4] tel quel\n"
    sb
end
document sb_force

```

```

sb_force - efface le bit CRC de a_sch[0] : l'ARM accepte le mot SB present, meme si le DSP l
end

define si_force
    set dsp_api.ndb->a_cd[0] = (1 << $B_BLUD)
    set dsp_api.ndb->a_cd[2] = 0
    printf "a_cd force : BLUD, fire_crc=0, biterr=0 -> l'ARM prend le bloc present comme bon\n"
    cd
end
document si_force
si_force - force l'acceptation du bloc a_cd present (BLUD + CRC Fire ok + 0 erreur) : l'ARM
end

define fake_cd
    set dsp_api.ndb->a_cd[ 1 ] = $arg 2    ×10
        = (3,0) (4,1) (5,2) (6,3) (7,4) (8,5) (9,6) (10,7) (11,8) (12,9)
    si_force
end
document fake_cd
fake_cd W3 W4 ... W12 - pose 10 mots de donnees dans a_cd[3..12] puis si_force (les 2 dernier
end

define fake_err
    set dsp_api.ndb->d_error_status = $arg0
    err
end
document fake_err
fake_err VAL - ecrit d_error_status (ex : fake_err 0 pour effacer DMA_PROG).
end

define fake_task
    set dsp_api.db_w->d_task_d = $arg0
    printf "db_w->d_task_d <- %u\n", $arg0
    tasks
end
document fake_task
fake_task N - programme la tache DL N dans la page W (5=FB 6=SB 1=PM 24=ALLC 19=NBS 21=NP 10=
end

define fake_page
    set dsp_api.ndb->d_dsp_page = $arg0
    pages

```

```

end
document fake_page
fake_page VAL - force d_dsp_page (bit0 = page W du DSP, bit1 = ...) pour tester la selection
end

define abort_dsp
    set dsp_api.db_w->d_ctrl_system = dsp_api.db_w->d_ctrl_system | (1 << $B_TASK_ABORT)
    printf "B_TASK_ABORT pose dans d_ctrl_system (page W)\n"
end
document abort_dsp
abort_dsp - pose B_TASK_ABORT : demande au DSP d'abandonner sa tache en cours (ce que fait L1)
end

# =====
# POINTS D'ARRET / SURVEILLANCE
# =====
define bfb
    break l1s_fbdet_resp
end
document bfb
bfb - s'arrete a chaque reponse de detection FB (l1s_fbdet_resp). Puis : mon, sync, fn.
end

define bsb
    break l1s_sbdet_resp
end
document bsb
bsb - s'arrete a chaque reponse SB (l1s_sbdet_resp). Puis : sb, mon.
end

define bfbsb
    break l1s_fbsb_req
end
document bfbsb
bfbsb - s'arrete quand la L1 recoit une demande FBSB du mobile (l1s_fbsb_req).
end

define bsync
    break synchronize_tdma
end
document bsync
bsync - s'arrete au recalage TDMA (synchronize_tdma), juste apres un FB/SB accepte.

```

```

end

define bnb
    break l1s_nb_resp
end
document bnb
bnb - s'arrete a chaque bloc normal reçu (l1s_nb_resp). Puis : cd, rxnb.
end

define bpm
    break l1s_pm_resp
end
document bpm
bpm - s'arrete a chaque mesure de puissance rendue (l1s_pm_resp). Puis : pm.
end

define breset
    break l1s_reset
end
document breset
breset - s'arrete a chaque reset L1 (l1s_reset) : qui remet la L1 a zero et quand.
end

define bcompl
    break l1s_compl_sched
end
document bcompl
bcompl - s'arrete a chaque completion programmee (l1s_compl_sched) : voir compl.
end

define babort
    break l1s_dsp_abort
end
document babort
babort - s'arrete quand l'ARM abandonne la tache DSP (l1s_dsp_abort).
end

define bframe
    break tdma_sched_execute
end
document bframe
bframe - s'arrete a CHAQUE trame TDMA (tdma_sched_execute). Lourd : utiliser avec trace_frame

```

```

end

define bl2
    break l1_queue_for_l2
end
document bl2
bl2 - s'arrete quand la L1 remonte un message a la L2 (SI, bloc, mesure).
end

define bafc
    break afc_input
end
document bafc
bafc - s'arrete a chaque entree AFC (afc_input freq_diff).
end

define wfb
    watch dsp_api.ndb->d_fb_det
end
document wfb
wfb - point d'arret sur ECRITURE de d_fb_det (vue ARM ; les ecritures du DSP ne passent pas p
end

define werr
    watch dsp_api.ndb->d_error_status
end
document werr
werr - point d'arret sur ecriture ARM de d_error_status.
end

define wtask
    watch dsp_api.db_w->d_task_d
end
document wtask
wtask - point d'arret sur programmation d'une tache DL par l'ARM.
end

define bl
    info breakpoints
end
document bl
bl - liste des points d'arret/surveillance.

```

```

end

define bclr
    delete
    printf "tous les points d'arret supprimes\n"
end
document bclr
bclr - supprime tous les points d'arret.
end

# =====
# BOUCLES DE TRACE (cible en avant-plan : Ctrl-C interrompt la boucle)
# =====
define trace_frames
    set $n = $arg0
    tbreak tdma_sched_execute
    continue
    while $n > 0
        fn
        tasks
        set $n = $n - 1
        if $n > 0
            tbreak tdma_sched_execute
            continue
        end
    end
    printf "-- %d trames tracees, cible ARRETEE (go pour reprendre)\n", $arg0
end
document trace_frames
trace_frames N - s'arrete N trames de suite (tdma_sched_execute) et affiche fn + taches a cha
end

define trace_fb
    set $n = $arg0
    while $n > 0
        tbreak lls_fbdet_resp
        continue
        fn
        sync
        set $n = $n - 1
    end
    printf "-- %d reponses FB tracees, cible ARRETEE (go pour reprendre)\n", $arg0

```

```

end
document trace_fb
trace_fb N - attend N reponses FB (l1s_fbdet_resp), affiche fn + sync a chacune.
end

define trace_sb
  set $n = $arg0
  while $n > 0
    tbreak l1s_sbdet_resp
    continue
    fn
    sb
    set $n = $n - 1
  end
  printf "-- %d reponses SB tracees, cible ARRETEE (go pour reprendre)\n", $arg0
end
document trace_sb
trace_sb N - attend N reponses SB (l1s_sbdet_resp), affiche fn + decodage a_sch a chacune.
end

define trace_nb
  set $n = $arg0
  while $n > 0
    tbreak l1s_nb_resp
    continue
    fn
    cd
    set $n = $n - 1
  end
  printf "-- %d blocs traces, cible ARRETEE (go pour reprendre)\n", $arg0
end
document trace_nb
trace_nb N - attend N blocs normaux (l1s_nb_resp), affiche fn + a_cd a chacun.
end

define trace_l2
  set $n = $arg0
  while $n > 0
    tbreak l1_queue_for_l2
    continue
    fn
    printf "msg -> L2 : "

```

```

    p *(struct msgb *)$r0
    set $n = $n - 1
end
printf "-- %d messages L2 traces, cible ARRETEE (go pour reprendre)\n", $arg0
end
document trace_l2
trace_l2 N - attend N messages remontes a la L2 (l1_queue_for_l2) et les affiche.
end

define mon_dsp
    set $n = $arg0
    while $n > 0
        tbreak tdma_sched_execute
        continue
        printf "==== trame %u ==== \n", l1s.current_time.fn
        dsp
        set $n = $n - 1
    end
    printf "-- %d echantillons, cible ARRETEE (go pour reprendre)\n", $arg0
end
document mon_dsp
mon_dsp N - N trames de suite, la vue `dsp` complete a chacune (pages, taches, ndb, sb).
end

define every
    set $n = $arg0
    while $n > 0
        tbreak tdma_sched_execute
        continue
        $arg1
        set $n = $n - 1
    end
    printf "-- fini, cible ARRETEE (go pour reprendre)\n"
end
document every
every N CMD - execute la commande CMD a chacune des N prochaines trames (ex : every 20 sync)
end

# =====
# AIDE
# =====
define help_osmo

```

```

printf "\n===== PANNEAU OSMOCOM (cmd.gdb) =====\n"
printf "EXECUTION      go          reprend sans bloquer (continue &)\n"
printf "                stop        arrete l'ARM (= Ctrl-C)\n"
printf "                regs / pcx / bt10 / si1 / ni1  registres, code a $pc, pile, pas a
printf "                frame_step  tourne jusqu'a la prochaine trame TDMA puis s'arrete
printf "TEMPS            fn          trame courante, suivante, tpu_offset\n"
printf "                l1           resume L1 : temps, cellule, modes, TA, pages\n"
printf "                cell / neigh / sched / mframe / compl / arfcn / afc\n"
printf "SYNCHRO           fbs / mon    etat FB/SB vu par l'ARM (fbs, fbs.mon)\n"
printf "                sync         d_fb_det + a_sync_demod + a_serv_demod\n"
printf "                sb           a_sch decode : CRC, BSIC, t1/t2/t3, fn\n"
printf "MAILBOX           dsp          TOUT : pages + tasks + ndb + sb\n"
printf "                pages / tasks / dbw / dbr / ndb / err / cd / pm / rxnb\n"
printf "                apiram OFF N  mots API bruts (adresse cote DSP)      apiw OFF VAL
printf "FORCAGES          fake_fb TOA PM ANGLE SNR    un FB detecte\n"
printf "                fake_sb BSIC T1 T2 T3        un SB valide          sb_force  effac
printf "                si_force                    accepte le bloc a_cd    fake_cd W3..W12
printf "                fake_err V / fake_task N / fake_page V / abort_dsp\n"
printf "POINTS D'ARRET    bfb bsb bfbsb bsync bnb bpm breset bcompl babort bframe bl2 bafc\n"
printf "                wfb werr wtask (watch)      bl liste          bclr efface tout\n"
printf "BOUCLES           trace_frames N / trace_fb N / trace_sb N / trace_nb N / trace_l2 N
printf "                mon_dsp N    la vue dsp a chaque trame, N fois\n"
printf "                every N CMD   CMD a chaque trame, N fois (ex : every 20 sync)\n"
printf "                Ctrl-C coupe une boucle ; la cible reste ARRETEE : go pour reprend
printf "DEMO             demo         la main sur le firmware : arret, etat, une trame, pr
printf "                demo etat | trame | hello | go      les etapes une a une\n"
printf "AIDE              help_osmo     ceci          help CMD   le detail d'une commande\n"
printf "Regle : les lectures (fn, dsp...) exigent la cible ARRETEE (stop) ; go ensuite.\n"
end
document help_osmo
help_osmo - liste toutes les commandes du panneau osmocom, avec leur role et leur usage.
end

printf "[cmd.gdb] panneau osmocom charge : help_osmo\n"
GDB
# [2026-10-06] la demo « la main sur le firmware » (tools/demo-layer1.py, Python gdb) : comm
# [2026-10-19] LE CHEMIN DE LA DEMO N'EST PLUS FIGE. `source $HERE/demo-layer1.py`
# ecrivait le repertoire DE GENERATION : quand build-iso (52-qemu.sh) regenere
# cmd.gdb dans le rootfs de l'image, c'est /var/tmp/iso-build-NNN/rootfs/opt/
# GSM/qosmo/tools qui partait dans l'ISO - et sur la machine installee, chaque
# `telnet 0 44444` finissait par « demo-layer1.py: No such file or directory »,
# sans commande demo. cmd.gdb cherche désormais le fichier : la ou la console

```

```

# telnet dit que sont les outils (OSMO_GDB_TOOLS, pose par gdb-telnet.py), puis
# le chemin de generation debarrasse d'un eventuel prefixe de rootfs
# (GDB_CMD_HERE le force), puis l'arbre installe.
BAKED="${GDB_CMD_HERE:-$HERE}"
case "$BAKED" in */rootfs/opt/GSM/*) BAKED="/opt/GSM/${BAKED#*/rootfs/opt/GSM/}" ;; esac
cat >> "$OUT" <<GDB
python
import gdb, os
for _d in (os.environ.get("OSMO_GDB_TOOLS", ""), "$BAKED", "/opt/GSM/qosmo/tools"):
    _f = os.path.join(_d, "demo-layer1.py") if _d else ""
    if _f and os.path.isfile(_f):
        gdb.execute("source " + _f)
        break
else:
    gdb.write("[cmd.gdb] demo-layer1.py introuvable (OSMO_GDB_TOOLS=%r) : commande demo absente\n"
              % os.environ.get("OSMO_GDB_TOOLS", ""))
end
GDB

echo "ecrit : $OUT ($(wc -l < "$OUT") lignes, $(grep -c '^define ' "$OUT") commandes)"
if [ "$INSTALL" = 1 ]; then
    touch ~/.gdbinit
    grep -qF "source $OUT" ~/.gdbinit || printf 'source %s\n' "$OUT" >> ~/.gdbinit
    echo "~/.gdbinit : source $OUT"
fi

```

0.3.73 /opt/GSM/qosmo-overlay/tools/injecteur_bruit.py

32633 octets, 708 lignes → 708 lignes

```

#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# injecteur_bruit.py - proxy UDP qui degrade le sens DESCENDANT entre le pont
# (osmo-operator/pont) et la couche 1 du telephone emule.
#
# [2026-10-03] Cree pour le banc Calypso. Un seul fichier source :
#   /opt/GSM/qosmo/tools/injecteur_bruit.py
# (c54x_exe/run.sh le prend la, avec repli sur c54x_exe/tools/injecteur_bruit.py,
# un lien vers ce fichier ; qosmo/run_modules/38-bruit.sh le lance pour le plan
# de qosmo).
#

```

```

# OU IL S'INTERCALE
#   cible dsp      : pont_dsp.py --dsp-port 6702 -> [injecteur 127.0.0.1:6702]
#                   -> BSP de c54x_exe deplace sur CALYPSO_BSP_PORT (16702 par
#                   default). Format : 8 octets d'en-tete [tn, fn BE32, att, 0, 0]
#                   puis 148 octets de bits DURS 0/1 (pont/trx.py Trx.run_data ;
#                   calypso_bsp.c bsp_trxd_readable).
#                   Le BSP apprend l'adresse de son pair montant sur l'emetteur du
#                   descendant : une fois l'injecteur intercale, ce qu'il renvoie
#                   (bursts UL TRXDv0, s'il en emet) arrive ici et repart SANS
#                   MODIFICATION vers la source (le pont, 127.0.0.1:5702), depuis
#                   le port d'ecoute -- exactement l'adresse qu'il avait avant.
#   cible grgsm    : pont.py -> GSMTAP [injecteur 127.0.0.1:14730] -> L1 gr-gsm de
#                   QEMU (127.0.0.1:4730, port fige dans calypso_l1_grgsm.c).
#                   Le pont y envoie si PONT_GSMTAP_PORT=14730 (pont/config.py ;
#                   pont/pont.py le pose seul quand BRUIT_MODE vaut ber|relais).
#                   Format : en-tete GSMTAP v2 de 16 octets puis le bloc L2 DEJA
#                   DECODE (23 octets) -- PAS des bursts, PAS de valeurs souples.
#                   Le SCH (4731, « SCH2 ») et la parole TCH (/dev/shm/calypso_tch_dl)
#                   ne passent pas par ici et ne sont pas touches.
#
# MODES (sens descendant seulement ; le montant est toujours relaye tel quel)
#   relais         aucune alteration : sert a verifier que l'intercalation seule ne
#                   change rien.
#   ber            inversion aleatoire de bits, taux moyen --ber. --rafales L : erreurs
#                   groupees (Gilbert-Elliott, rafales de L bits en moyenne, taux
#                   --ber-rafales dans une rafale), meme taux moyen.
#                   dsp : bits durs inverses (0<->1). grgsm : bits des octets L2 inverses
#                   APRES decodage -- non physique (ni code correcteur ni CRC ne
#                   protegent plus le bloc) ; --perte est le modele realiste en grgsm.
#   souple         AWGN a --snr-db sur les valeurs souples. VERIFIE LE 2026-10-03 :
#                   aucun des deux flux n'en porte (dsp : bits durs 0/1 ; grgsm : octets
#                   L2). Sur des bits durs : modulation antipodale +-1, AWGN, decision
#                   dure -- BER = Q(sqrt(2 snr)), annonce au demarrage ; la sortie reste
#                   0/1, le seul format que le BSP sait lire. Si une source envoie un jour
#                   des valeurs souples int8 (convention TRXD : +127 = bit 1), elles sont
#                   bruitees et gardees souples. Refuse en grgsm.
#   iq             EXPERIMENTAL. Bits -> GMSK BT=0.3 a 4 echantillons/symbole (meme
#                   modulateur que calypso_gmsk.c, instant 0,5) + AWGN a --snr-db, envoie
#                   en I/Q int16 entrelaces (IQ_PASSTHROUGH du BSP, decimation par 4).
#                   CONTOURNE la modulation reglee du BSP (sb_moduler pour le SCH :
#                   instant 0,35, elargissement 1,0, bruit propre ; gmsk_elargir 0,3 sur
#                   les NB -- seul ce dernier est imite ici, --iq-elargir).

```

```

#          INCOMPATIBLE avec CALYPSO_BSP_STREAM=1 (le default de
#          start-native.sh --dsp) : en STREAM le BSP range les 148 premiers
#          octets du paquet comme des bits AVANT la branche IQ_PASSTHROUGH
#          (calypso_bsp.c bsp_trxd_readable -> bsp_ts0_stocker). Refuse dans ce
#          cas, sauf --forcer-iq. Necessite numpy.
#  --perte P (tous modes) : un burst/bloc sur 1/P est efface. dsp : la charge
#          est remplacee par des bits aleatoires (evanouissement : le BSP garde
#          sa cadence, un burst jete le ferait attendre CALYPSO_BSP_ATTENTE_MS).
#          grgsm : le bloc n'est pas transmis (= echec CRC cote L1).
#
# DETERMINISME : chaque paquet tire son alea d'une graine derivee de
# (--graine, fn, tn) : meme graine + memes trames = memes erreurs, quel que soit
# l'ordre d'arrivee (sauf --rafales, dont l'etat suit l'ordre d'arrivee). Sans
# --graine, une graine est tiree et ECRITE au demarrage pour pouvoir rejouer.
#
# Arret propre sur SIGTERM/SIGINT (statistiques finales), SIGUSR1 = statistiques
# tout de suite. --verifier : controle la configuration et sort (0 = bon).
import argparse
import logging
import math
import os
import random
import selectors
import signal
import socket
import struct
import sys
import time

VERSION = "2026-10-03"
log = logging.getLogger("bruit")

PROFILS = {
    # cible : (ecoute = la ou la source envoie, vers = la couche 1)
    "dsp": ("127.0.0.1:6702", "127.0.0.1:16702"),
    "grgsm": ("127.0.0.1:14730", "127.0.0.1:4730"),
}

MODES = ("relais", "ber", "souple", "iq")
MODES_GRGSM = ("relais", "ber")

DSP_HDR = 8          # [tn, fn BE32, att, 0, 0]
NBITS = 148

```

```

IQ_MIN_OCTETS = 4 * 146 # seuil de calypso_bsp.c pour lire la charge en I/Q
GSMTAP_TYPE_UM = 0x01
MASQUE64 = (1 << 64) - 1

def _adresse(txt, option):
    hote, sep, port = txt.rpartition(":")
    if not sep or not hote:
        raise argparse.ArgumentTypeError("%s : HOTE:PORT attendu, pas %r" % (option, txt))
    try:
        p = int(port)
    except ValueError:
        raise argparse.ArgumentTypeError("%s : port invalide %r" % (option, port))
    if not 0 < p < 65536:
        raise argparse.ArgumentTypeError("%s : port hors bornes %d" % (option, p))
    return (hote, p)

def _q_gauss(x):
    """Q(x) = P(N(0,1) > x)."""
    return 0.5 * math.erfc(x / math.sqrt(2.0))

def _geometrique(rng, p):
    """Nombre d'essais jusqu'au premier succes (>= 1), succes de probabilite p."""
    if p >= 1.0:
        return 1
    if p <= 0.0:
        return 1 << 62
    u = rng.random()
    return 1 + int(math.log1p(-u) / math.log1p(-p))

class Rafales:
    """Gilbert-Elliott : etat bon (aucune erreur) / mauvais (taux q), sejours
    geometriques. Longueur moyenne d'une rafale L bits, taux moyen p."""

    def __init__(self, p, longueur, q):
        self.q = q
        self.pi_m = p / q # fraction du temps en rafale
        self.p_mb = 1.0 / longueur # mauvais -> bon
        self.p_bm = self.p_mb * pi_m / (1.0 - pi_m) if pi_m < 1.0 else 1.0

```

```

        self.mauvais = False
        self.restant = None

    def positions(self, n, rng):
        if self.restant is None:
            self.restant = _geometrique(rng, self.p_bm)
        out = []
        pos = 0
        while pos < n:
            k = min(self.restant, n - pos)
            if self.mauvais:
                out.extend(i for i in range(pos, pos + k) if rng.random() < self.q)
            pos += k
            self.restant -= k
            if self.restant == 0:
                self.mauvais = not self.mauvais
                self.restant = _geometrique(rng, self.p_mb if self.mauvais else self.p_bm)
        return out

def positions_iid(n, p, rng):
    """Positions des bits inverses, independants de probabilite p (saut geometrique)."""
    if p <= 0.0:
        return []
    if p >= 1.0:
        return list(range(n))
    out = []
    lp = math.log1p(-p)
    i = -1
    while True:
        i += 1 + int(math.log1p(-rng.random()) / lp)
        if i >= n:
            return out
        out.append(i)

class ModulateurGmsk:
    """GMSK BT=0.3 (45.004) a `sps` echantillons par symbole, comme
    calypso_gmsk.c gmsk_moduler() : codage differentiel  $d_i = b_i \hat{b}_{i-1}$ 
    ( $b_{-1} = 1$ ),  $\alpha_i = 1 - 2 d_i$ ,  $\text{phase}(t) = \pi/2 * \sum \alpha_i q(t - i)$ ,
    echantillon j a l'instant  $t = j/\text{sps} + \text{decalage}$ . Avec  $\text{sps}=4$  et  $\text{decalage}=0,5$ ,
    les echantillons 4k (ceux que garde le BSP, decimation par 4) sont ceux de
    """

```

```

gmsk_moduler(..., decalage=0.5)."""

def __init__(self, np, sps=4, decalage=0.5):
    self.np = np
    self.sps = sps
    # La table q(t) est calculee EXACTEMENT comme calypso_gmsk.c q_init() :
    # somme de Riemann a gauche, pas T/16, normalisee, puis interpolation
    # lineaire (q_de). Une integration plus fine donnerait une autre forme
    # d'onde (q(0) = 0,500 au lieu de 0,477 : jusqu'a 0,05 rad d'ecart) et
    # le mode iq ne serait plus « le meme modulateur » que le BSP.
    bt, os_, lg = 0.3, 16, 3
    k = 2.0 * math.pi * bt / math.sqrt(math.log(2.0))
    dt = 1.0 / os_
    tab, acc = [], 0.0
    for i in range(lg * os_ + 1):
        t = -1.5 + i * dt
        tab.append(acc)
        acc += (_q_gauss(k * (t - 0.5)) - _q_gauss(k * (t + 0.5))) * dt
    tab = [v / acc for v in tab]

    def q(t):
        if t <= -1.5:
            return 0.0
        if t >= 1.5:
            return 1.0
        x = (t + 1.5) * os_
        i = int(x)
        f = x - i
        if i >= lg * os_:
            return 1.0
        return tab[i] * (1.0 - f) + tab[i + 1] * f

    # K(n) = q(n/sps + decalage) : 0 pour t <= -1,5, 1 pour t >= 1,5.
    n_lo = math.floor((-1.5 - decalage) * sps) + 1
    n_un = math.ceil((1.5 - decalage) * sps)
    self.n_lo = n_lo
    self.n_un = n_un
    self.k_trans = np.array([q(n / sps + decalage) for n in range(n_lo, n_un)])

def phase(self, bits):
    np = self.np
    b = np.asarray(bits, dtype=np.int8) & 1

```

```

prec = np.concatenate(([1], b[:-1]))
alpha = 1.0 - 2.0 * (b ^ prec)
n = len(b)
long_ = n * self.sps
d = np.zeros(long_)
d[:, self.sps] = alpha
conv = np.convolve(d, self.k_trans) # conv[k] <-> j = k + n_lo
j = np.arange(long_)
part_trans = conv[j - self.n_lo]
cumul = np.cumsum(d)
idx = j - self.n_un
part_marche = np.where(idx >= 0, cumul[np.clip(idx, 0, None)], 0.0)
return (math.pi / 2.0) * (part_trans + part_marche)

```

```
class Stats:
```

```

    def __init__(self):
        self.t0 = time.monotonic()
        self.t_prec = self.t0
        self.dl_prec = 0
        self.dl = 0 # paquets descendants recus
        self.relayes = 0 # paquets descendants envoyes vers la L1
        self.alteres = 0 # paquets dont la charge a ete degradee
        self.effaces = 0 # --perte
        self.hors_tn = 0 # hors --tn, relayes tels quels
        self.autres = 0 # format non reconnu, relayes tels quels
        self.iq_deja = 0 # charge deja en I/Q, relayee telle quelle
        self.iq_produits = 0 # bursts convertis en I/Q (mode iq)
        self.souples = 0 # paquets a valeurs souples vus
        self.bits = 0 # bits durs examines (ber, souple sur bits durs)
        self.inverses = 0 # bits durs changes
        self.ul = 0 # paquets montants relayes vers la source
        self.ul_sans_dest = 0 # montant reçu avant tout descendant
        self.err_envoi = 0

    def ligne(self, final=False):
        now = time.monotonic()
        dt = max(now - self.t_prec, 1e-9)
        pps = (self.dl - self.dl_prec) / dt
        self.t_prec, self.dl_prec = now, self.dl
        ber = (self.inverses / self.bits) if self.bits else 0.0
        return ("%s t=%.0fs DL recus=%d relayes=%d alteres=%d effaces=%d hors_tn=%d autres=%d" %

```

```

        " iq_produits=%d souples=%d | bits=%d inverses=%d BER=%.3e | UL relayes=%d s
        " | erreurs_envoi=%d | %.0f pq/s"
        % ("BILAN" if final else "STATS", now - self.t0, self.dl, self.relayes, self
        self.effaces, self.hors_tn, self.autres, self.iq_deja, self.iq_produits, s
        self.bits, self.inverses, ber, self.ul, self.ul_sans_dest, self.err_envoi

class Degradeur:
    def __init__(self, a, np=None):
        self.a = a
        self.cible = a.cible
        self.mode = a.mode
        self.stats = Stats()
        self.graine = a.graine
        self.tn = a.tn_set
        self.rafales = Rafales(a.ber, a.rafales, a.ber_rafale) if (a.mode == "ber" and a.rafales) else None
        self.snr_lin = 10.0 ** (a.snr_db / 10.0) if a.snr_db is not None else None
        self.p_souple = _q_gauss(math.sqrt(2.0 * self.snr_lin)) if self.snr_lin is not None else None
        self.np = np
        self.mod = ModulateurGmsk(np) if a.mode == "iq" else None

    def _rng(self, fn, tn, sel=0):
        x = (self.graine * 0x9E3779B97F4A7C15 + (fn + 1) * 0x100000001B3 + tn * 0x2545F491 + sel) & 0xFFFFFFFF
        return random.Random(x)

# -- cible dsp -----
def _dsp(self, pkt):
    st = self.stats
    if len(pkt) < DSP_HDR + NBITS:
        st.autres += 1
        return pkt
    charge = len(pkt) - DSP_HDR
    if charge >= IQ_MIN_OCTETS:
        st.iq_deja += 1
        return pkt
    tn = pkt[0] & 0x07
    if self.tn is not None and tn not in self.tn:
        st.hors_tn += 1
        return pkt
    fn = struct.unpack_from(">I", pkt, 1)[0]
    rng = self._rng(fn, tn)
    ba = bytearray(pkt)

```

```

dur = not bytes(ba[DSP_HDR:]).translate(None, b"\x00\x01") # que des 0/1 : bits dur
if not dur:
    st.souples += 1
efface = self.a.perte > 0.0 and rng.random() < self.a.perte
if efface:
    st.effaces += 1
    if dur:
        for i in range(DSP_HDR, len(ba)):
            ba[i] = rng.getrandbits(1)
    else:
        for i in range(DSP_HDR, len(ba)):
            ba[i] = 0 # souple : effacement = aucun
mode = self.mode
if mode == "relais":
    if efface:
        st.alteres += 1
    return bytes(ba)
if mode == "ber":
    n = len(ba) - DSP_HDR
    pos = self.rafales.positions(n, rng) if self.rafales else positions_iid(n, self.
    if dur:
        for i in pos:
            ba[DSP_HDR + i] ^= 1
    else:
        for i in pos: # souple : signe inverse
            v = ba[DSP_HDR + i]
            v = v - 256 if v > 127 else v
            ba[DSP_HDR + i] = min(127, -v) & 0xFF
    st.bits += n
    st.inverses += len(pos)
    st.alteres += 1
    return bytes(ba)
if mode == "souple":
    snr = self.snr_lin
    if dur:
        # +-1 + N(0, sigma^2), sigma^2 = 1/(2 snr), puis decision dure :
        # chaque bit change independamment avec la probabilite
        # Q(1/sigma) = Q(sqrt(2 snr)). On tire donc directement ces
        # positions -- meme loi, exactement, sans 148 gaussiennes.
        n = len(ba) - DSP_HDR
        pos = positions_iid(n, self.p_souple, rng)
        for i in pos:

```

```

        ba[DSP_HDR + i] ^= 1
        st.bits += n
        st.inverses += len(pos)
    else:
        vals = [(v - 256 if v > 127 else v) for v in ba[DSP_HDR:]]
        amp = (sum(abs(v) for v in vals) / len(vals)) or 127.0
        sigma = amp / math.sqrt(2.0 * snr)
        for k, v in enumerate(vals):
            w = int(round(v + rng.gauss(0.0, sigma)))
            ba[DSP_HDR + k] = max(-127, min(127, w)) & 0xFF
        st.alteres += 1
        return bytes(ba)
# mode iq
if dur:
    bits = list(ba[DSP_HDR:DSP_HDR + NBITS])
else:
    # souple int8 : decision dure
    bits = [1 if 0 < v <= 127 else 0 for v in ba[DSP_HDR:DSP_HDR + NBITS]]
return bytes(ba[:DSP_HDR]) + self._iq(bits, fn, tn)

def _iq(self, bits, fn, tn):
    np = self.np
    a = self.a
    ph = self.mod.phase(bits)
    i_ = a.amp * np.cos(ph)
    q_ = a.amp * np.sin(ph)
    if a.iq_elargir and any(bits):
        # Imite gmsk_elargir() du BSP sur les echantillons qu'il garde
        # (voisins a +-1 symbole = +-sps echantillons) ; FCCH (tout a zero)
        # laisse tel quel, comme le BSP.
        s, e = self.mod.sps, a.iq_elargir
        for v in (i_, q_):
            w = v.copy()
            w[s:] += e * v[:-s]
            w[:-s] += e * v[s:]
            v[:] = w / (1.0 + 2.0 * e)
    if self.snr_lin is not None:
        sigma = math.sqrt(a.amp * a.amp / (2.0 * self.snr_lin))
        g = np.random.Generator(np.random.PCG64(self._rng(fn, tn, 1).getrandbits(64)))
        i_ = i_ + g.normal(0.0, sigma, i_.shape)
        q_ = q_ + g.normal(0.0, sigma, q_.shape)
    out = np.empty(2 * len(i_), dtype="<i2")
    out[0::2] = np.clip(np rint(i_), -32768, 32767)

```

```

        out[1::2] = np.clip(np rint(q_), -32768, 32767)
        self.stats.iq_produits += 1
        self.stats.alteres += 1
        return out.tobytes()

# -- cible grgsm -----
def _grgsm(self, pkt):
    st = self.stats
    if len(pkt) < 16 or pkt[0] != 2 or pkt[2] != GSMTAP_TYPE_UM:
        st.autres += 1
        return pkt
    hdr = pkt[1] * 4
    if hdr < 16 or hdr >= len(pkt):
        st.autres += 1
        return pkt
    tn = pkt[3] & 0x07
    if self.tn is not None and tn not in self.tn:
        st.hors_tn += 1
        return pkt
    fn = struct.unpack_from(">I", pkt, 8)[0]
    rng = self._rng(fn, tn, pkt[12])
    if self.a.perte > 0.0 and rng.random() < self.a.perte:
        st.effaces += 1
        return None # bloc perdu = echec CRC cote
    if self.mode == "relais":
        return pkt
    ba = bytearray(pkt)
    n = (len(ba) - hdr) * 8
    pos = self.rafales.positions(n, rng) if self.rafales else positions_iid(n, self.a.be
    for i in pos:
        ba[hdr + (i >> 3)] ^= 0x80 >> (i & 7)
    st.bits += n
    st.inverses += len(pos)
    st.alteres += 1
    return bytes(ba)

def descendant(self, pkt):
    self.stats.dl += 1
    return self._dsp(pkt) if self.cible == "dsp" else self._grgsm(pkt)

def _parser():

```

```

p = argparse.ArgumentParser(
    prog="injecteur_bruit.py",
    description="Proxy UDP transparent pont -> couche 1 qui degrade le sens descendant "
        "(bits inverses, AWGN, I/Q bruite). Le montant est relaye sans modification."
p.add_argument("--cible", choices=sorted(PROFILS), default="dsp",
    help="format du flux : dsp (TRXD 8 o + 148 bits, BSP de c54x_exe) ou grgsm
        "(GSMTAP vers la L1 gr-gsm de QEMU) ; default dsp")
p.add_argument("--mode", choices=MODES, default="ber", help="default ber")
p.add_argument("--ecoute", help="HOTE:PORT ou la source envoie (dsp 127.0.0.1:6702, grgsm 127.0.0.1:6702) ; default None")
p.add_argument("--vers", help="HOTE:PORT de la couche 1 (dsp 127.0.0.1:16702, grgsm 127.0.0.1:16702) ; default None")
p.add_argument("--source", help="HOTE:PORT local d'emission vers la L1 (default : port local) ; default None")
p.add_argument("--retour", help="HOTE:PORT ou renvoyer le montant (default : l'emetteur de la couche 1) ; default None")
p.add_argument("--ber", type=float, default=0.0, help="mode ber : taux moyen d'inversion ; default 0.0")
p.add_argument("--rafales", type=float, default=0.0, metavar="L",
    help="mode ber : erreurs en rafales de L bits en moyenne (Gilbert-Elliott) ; default 0.0")
p.add_argument("--ber-rafales", type=float, default=0.5, help="taux d'erreur dans une rafale ; default 0.5")
p.add_argument("--snr-db", type=float, default=None,
    help="modes souple et iq : SNR en dB, P_signal / (2 sigma^2) par echantillon ; default None
        "(meme definition que CALYPSO_BSP_SNR_DB)")
p.add_argument("--perte", type=float, default=0.0,
    help="probabilite d'effacer un burst (dsp : bits aleatoires) ou un bloc (grgsm : blocs de 160 bits) ; default 0.0")
p.add_argument("--tn", default="tous", help="intervalles a degrader, ex. 0 ou 1,2 (default : tous) ; default tous")
p.add_argument("--graine", type=lambda s: int(s, 0), default=None, help="graine (deterministe) ; default None")
p.add_argument("--amp", type=int, default=30000, help="mode iq : amplitude (default 30000) ; default 30000")
p.add_argument("--iq-elargir", type=float, default=0.3,
    help="mode iq : elargissement des NB comme gmsk_elargir (default 0,3 ; 0 = pas d'elargissement) ; default 0.3")
p.add_argument("--forcer-iq", action="store_true",
    help="mode iq malgre CALYPSO_BSP_STREAM=1 ou CALYPSO_BSP_IQ_PASSTHROUGH=0 ; default False")
p.add_argument("--stats", type=float, default=10.0, help="periode des statistiques en s ; default 10.0")
p.add_argument("--journal", help="fichier journal (en plus de stderr) ; default None")
p.add_argument("--pidfile", help="ecrit le pid, efface a l'arret" ; default None")
p.add_argument("--etiquette", default="", help="libre, pour reconnaitre le processus (pk) ; default """)
p.add_argument("--verifier", action="store_true", help="controle la configuration et sort le resultat ; default False")
p.add_argument("--version", action="version", version="%(prog)s " + VERSION)
return p

```

```

def _valider(a, p):
    """Complete et controle les arguments ; rend numpy (mode iq) ou None."""
    ecoute, vers = PROFILS[a.cible]
    a.ecoute = _adresse(a.ecoute or ecoute, "--ecoute")
    a.vers = _adresse(a.vers or vers, "--vers")

```

```

a.source = _adresse(a.source, "--source") if a.source else None
a.retour = _adresse(a.retour, "--retour") if a.retour else None
if a.ecoute == a.vers:
    p.error("--ecoute et --vers designent la meme adresse %s:%d : l'injecteur s'enverrait")
if a.cible == "grgsm" and a.mode not in MODES_GRGSM:
    p.error("cible grgsm : mode %s impossible. Le flux GSMTAP vers QEMU porte des blocs 1
            "(16 o d'en-tete + 23 o), ni valeurs souples ni bursts a moduler (calypso_11.
            "gsmtap_readable). Modes possibles : %s (et --perte)." % (a.mode, ", ".join(MODES_GRGSM)))
for nom in ("ber", "perte", "ber_rafale"):
    v = getattr(a, nom)
    if not 0.0 <= v <= 1.0:
        p.error("--%s doit etre entre 0 et 1 (%g)" % (nom.replace("_", "-"), v))
if a.mode == "ber" and a.rafales:
    if a.rafales < 1.0:
        p.error("--rafales : longueur moyenne en bits >= 1 (%g)" % a.rafales)
    if not 0.0 < a.ber_rafale <= 1.0 or a.ber >= a.ber_rafale:
        p.error("--rafales : il faut --ber (%g) < --ber-rafale (%g) <= 1" % (a.ber, a.ber_rafale))
if a.mode == "souple" and a.snr_db is None:
    p.error("mode souple : --snr-db obligatoire")
if a.snr_db is not None and not -50.0 <= a.snr_db <= 100.0:
    p.error("--snr-db hors bornes raisonnables (%g)" % a.snr_db)
if a.tn in ("", "tous", "all"):
    a.tn_set = None
else:
    try:
        a.tn_set = {int(x) for x in a.tn.split(",") if x.strip() != ""}
    except ValueError:
        p.error("--tn : liste d'intervalles 0..7 attendue (%r)" % a.tn)
    if not a.tn_set or any(not 0 <= x <= 7 for x in a.tn_set):
        p.error("--tn : intervalles 0..7 (%r)" % a.tn)
if a.graine is None:
    a.graine = int.from_bytes(os.urandom(4), "little")
    a.graine_tiree = True
else:
    a.graine_tiree = False
a.graine &= MASQUE64
np = None
if a.mode == "iq":
    if a.cible != "dsp":
        p.error("mode iq : cible dsp seulement")
    try:
        import numpy as np # noqa: F811 - seul le mode iq en a besoin

```

```

except ImportError:
    p.error("mode iq : numpy est necessaire et introuvable pour %s "
            "(apt-get install python3-numpy, ou pip install numpy). "
            "Les modes relais, ber et souple n'en ont pas besoin." % sys.executable)
if not a.forcer_iq:
    if os.environ.get("CALYPSO_BSP_STREAM", "") == "1":
        p.error("mode iq refuse : CALYPSO_BSP_STREAM=1. En STREAM le BSP range les 1
                "de chaque paquet comme des BITS avant de regarder s'il s'agit d'I/Q
                "(calypso_bsp.c bsp_trxd_readable -> bsp_ts0_stocker) : l'I/Q y sera
                "de bits. Lancez c54x_exe sans STREAM, ou --forcer-iq (BRUIT_OPTS=--
        if os.environ.get("CALYPSO_BSP_IQ_PASSTHROUGH", "") == "0":
            p.error("mode iq refuse : CALYPSO_BSP_IQ_PASSTHROUGH=0, le BSP ne lit pas l'
if not 1000 <= a.amp <= 32767:
    p.error("--amp : 1000..32767 (%d)" % a.amp)
if not 0.0 <= a.iq_elargir <= 2.0:
    p.error("--iq-elargir : 0..2 (%g)" % a.iq_elargir)
return np

def _ouvrir(a):
    amont = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    try:
        amont.bind(a.ecoute)
    except OSError as e:
        amont.close()
        raise SystemExit("injecteur_bruit : impossible d'ecouter sur %s:%d (%s) -- un autre p
                        "tient-il deja ce port ? ss -lunp | grep :%d" % (a.ecoute[0], a.ecout
    aval = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    if a.source:
        loc = a.source
    else:
        loc = ("127.0.0.1", 0) if a.vers[0].startswith("127.") or a.vers[0] == "localhost" e
    try:
        aval.bind(loc)
    except OSError as e:
        amont.close()
        aval.close()
        raise SystemExit("injecteur_bruit : impossible de lier le port d'emission %s:%d (%s)
    for s in (amont, aval):
        s.setblocking(False)
        try:
            s.setsockopt(socket.SOL_SOCKET, socket.SO_RCVBUF, 1 << 20)

```

```

        except OSError:
            pass
    return montant, aval

def _journal(a):
    fmt = logging.Formatter("%(asctime)s [bruit] %(message)s", "%H:%M:%S")
    log.setLevel(logging.INFO)
    h = logging.StreamHandler(sys.stderr)
    h.setFormatter(fmt)
    log.addHandler(h)
    if a.journal:
        f = logging.FileHandler(a.journal)
        f.setFormatter(fmt)
        log.addHandler(f)

def _annoncer(a, aval, np=None):
    loc = aval.getsockname()
    log.info("injecteur de bruit %s : cible %s, mode %s%s", VERSION, a.cible, a.mode,
            " (EXPERIMENTAL)" if a.mode == "iq" else "")
    if a.mode == "ber":
        if a.rafales:
            log.info("BER moyen %.3e en rafales de %.1f bits (taux %.2f dans une rafale, Gilb
                    a.ber, a.rafales, a.ber_rafale)
        else:
            log.info("BER %.3e (bits independants)", a.ber)
        if a.cible == "grgsm" and a.ber > 0:
            log.info("ATTENTION grgsm : bits inverses dans le bloc L2 DEJA DECODE, rien ne l
                    "(ni FEC ni CRC) ; --perte modelise l'echec CRC")
    if a.mode == "souple":
        snr = 10.0 ** (a.snr_db / 10.0)
        log.info("SNR %.2f dB ; sur bits durs (le cas du pont) : +-1 + AWGN + decision dure,
                "BER attendu Q(sqrt(2 snr)) = %.3e", a.snr_db, _q_gauss(math.sqrt(2.0 * snr)
    if a.mode == "iq":
        log.info("I/Q GMSK 4 ech/symbole, amplitude %d, elargissement NB %.2f, %s ; "
                "contourne sb_moduler/gmsk_elargir du BSP", a.amp, a.iq_elargir,
                ("AWGN %.2f dB" % a.snr_db) if a.snr_db is not None else "SANS AWGN")
        if a.snr_db is not None and np is not None:
            # Ecretage int16 : a 30000 il ne reste que 2767 de marge. On l'estime
            # pour l'annoncer (le C, CALYPSO_BSP_SNR_DB, ecrete de la meme facon).
            g = np.random.default_rng(0)

```

```

        m = 20000
        s = a.amp * np.exp(1j * g.uniform(0.0, 2.0 * math.pi, m))
        sig = math.sqrt(a.amp * a.amp / (2.0 * 10.0 ** (a.snr_db / 10.0)))
        y = s + sig * (g.standard_normal(m) + 1j * g.standard_normal(m))
        ecr = float(np.mean((np.abs(y.real) > 32767) | (np.abs(y.imag) > 32767)))
        if ecr > 0.01:
            yc = np.clip(y.real, -32768, 32767) + 1j * np.clip(y.imag, -32768, 32767)
            eff = 10.0 * math.log10(a.amp * a.amp / float(np.mean(np.abs(yc - s) ** 2)))
            log.info("ATTENTION ecretage int16 : ~%.0f %% des echantillons, SNR effectif
                    "(bruit non gaussien) ; --amp plus bas pour l'eviter", 100.0 * ecr,
if a.perte:
    log.info("effacement : %.3e des %s", a.perte, "bursts (bits aleatoires)" if a.cible == "bits" else "frames")
if (a.mode == "relais" or (a.mode == "ber" and a.ber == 0.0)) and not a.perte:
    log.info("aucune degradation configuree : relais transparent")
log.info("intervalles : %s ; graine %d%s", "tous" if a.tn_set is None else ", ".join(map(lambda g: a.graine, a.graine_tiree)),
        a.graine, " (tiree : --graine %d pour rejouer)" % a.graine if a.graine_tiree else "")
log.info("PRET ecoute=%s:%d vers=%s:%d emission=%s:%d retour=%s", a.ecoute[0], a.ecoute[1], a.vers[0], a.vers[1],
        loc[0], loc[1], ("%s:%d" % a.retour) if a.retour else "emetteur du descendant")

def main(argv=None):
    p = _parser()
    a = p.parse_args(argv)
    np = _valider(a, p)
    if a.verifier:
        amont, aval = _ouvrir(a)
        amont.close()
        aval.close()
        print("injecteur_bruit : configuration valide (cible %s, mode %s, ecoute %s:%d -> %s
              % (a.cible, a.mode, a.ecoute[0], a.ecoute[1], a.vers[0], a.vers[1]))
        return 0
    _journal(a)
    amont, aval = _ouvrir(a)
    deg = Degradeur(a, np)
    st = deg.stats
    if a.pidfile:
        with open(a.pidfile, "w") as f:
            f.write("%d\n" % os.getpid())

    arret = []
    flash = []
    signal.signal(signal.SIGTERM, lambda s, f: arret.append(s))

```

```

signal.signal(signal.SIGINT, lambda s, f: arret.append(s))
signal.signal(signal.SIGUSR1, lambda s, f: flash.append(s))

_annoncer(a, aval, np)
sel = selectors.DefaultSelector()
sel.register(amont, selectors.EVENT_READ, "dl")
sel.register(aval, selectors.EVENT_READ, "ul")
source = None
prochain = time.monotonic() + a.stats if a.stats > 0 else None
try:
    while not arret:
        try:
            evts = sel.select(timeout=0.5)
        except InterruptedError:
            evts = []
        for cle, _ in evts:
            s = cle.fileobj
            for _ in range(512):
                try:
                    pkt, adr = s.recvfrom(65536)
                except (BlockingIOError, InterruptedError):
                    break
                except OSError:
                    st.err_envoi += 1 # ICMP remonte sur la socket : on continue
                    break
                if cle.data == "dl":
                    source = adr
                    sortie = deg.descendant(pkt)
                    if sortie is None:
                        continue
                    try:
                        aval.sendto(sortie, a.vers)
                        st.relayes += 1
                    except OSError:
                        st.err_envoi += 1
                else:
                    dest = a.retour or source
                    if dest is None:
                        st.ul_sans_dest += 1
                        continue
                    try:
                        amont.sendto(pkt, dest) # montant : tel quel

```

```

        st.ul += 1
    except OSError:
        st.err_envoi += 1
    now = time.monotonic()
    if flash or (prochain is not None and now >= prochain):
        del flash[:]
        log.info("%s", st.ligne())
        if prochain is not None:
            prochain = now + a.stats
    finally:
        log.info("%s", st.ligne(final=True))
        log.info("arret (%s)", "signal %d" % arret[0] if arret else "fin")
        sel.close()
        amont.close()
        aval.close()
        if a.pidfile:
            try:
                os.unlink(a.pidfile)
            except OSError:
                pass
    return 0

if __name__ == "__main__":
    sys.exit(main())

```

0.3.74 /opt/GSM/qosmo-overlay/tools/qosmo-couleurs.py

4922 octets, 89 lignes → 89 lignes

```

#!/usr/bin/env python3
"""qosmo-couleurs : colore a la volee les journaux du banc (stdin -> stdout).

tail -F qemu-asm.log | qosmo-couleurs          trace -d in_asm / exec de QEMU
tail -F qemu.log      | qosmo-couleurs          [trx] [sim] [dcch] [c54x] ...
tail -F mailbox.log   | qosmo-couleurs          ARM>WR ARM<RD DSP>WR DSP<RD
watch -n 0.5 --color "tail -n 40 qemu-asm.log | qosmo-couleurs"

```

Les lignes deja colorees (mobile.log) passent telles quelles. QOSMO_COULEURS=0 ou une sortie qui n'est pas un terminal (et pas --force) : copie sans couleur. Les panes tmux du banc (tmux_modules/_commun.sh) et ./run.sh --logs de c54x_exe

```

passent par ici. Rien n'est ecrit dans les fichiers : la couleur est a l'affichage."""
import os, re, sys, signal

def principal():
    force = '--force' in sys.argv[1:]
    actif = os.environ.get('QOSMO_COULEURS', '1') != '0' and (force or sys.stdout.isatty())
    out = sys.stdout
    if not actif:
        for l in sys.stdin:
            out.write(l); out.flush()
        return
    R = '\033[0m'
    def c(n, g=''): return f'\033[{g}38;5;{n}m'
    DIM, GRAS = '\033[2m', '\033[1m'
    ADR, OCT, SYM, PC, HOTE = c(81, '1;'), c(242), c(114), c(221, '1;'), c(240)
    TAGS = {'trx': c(75), 'sim': c(176), 'dcch': c(80), 'c54x': c(179), 'mbx': c(114),
            'mailbox': c(114), 'sb': c(215), 'a_sch': c(215), 'sbwin': c(215), 'jalon': c(20),
            'fb': c(215), 'pont': c(75), 'bsp': c(179), 'calypso-debug': c(242), 'run': c(25),
            'gdb-telnet': c(250), 'api': c(114), 'a5': c(176), 'rhea': c(179)}
    SENS = {'ARM>WR': c(221, '1;'), 'ARM<RD': c(81), 'DSP>WR': c(114, '1;'), 'DSP<RD': c(176),
            'ARM>RD': c(221, '1;'), 'ARM<WR': c(81), 'DSP>RD': c(114, '1;'), 'DSP<WR': c(176)}
    MAUVAIS = c(203, '1;'); BON = c(114, '1;'); ATTENTION = c(215)
    re_adr = re.compile(r'^([0-9a-f]{8}):(.*)$')
    re_objd = re.compile(r'^([OBJD-T:]\s*([0-9a-f]+)\s*$)')
    re_trace = re.compile(r'^([Trace \d+: ]([0-9a-f]+) \[([0-9a-f]+)/([0-9a-f]+)/([0-9a-f]+) |')
    re_stop = re.compile(r'^([Stopped execution of TB chain before|Linking TBs|cpu_io_recomp]')
    re_tag = re.compile(r'^([s*]\[([A-Za-z0-9_-]+)\])')
    re_mbx = re.compile(r'^([s*\d+\s+\d+\s+](ARM>WR|ARM<RD|DSP>WR|DSP<RD)\s+)([0-9a-f]+)')
    re_fn = re.compile(r'^([b(fn=)(\d+)]')
    re_hex = re.compile(r'^([b(0x[0-9A-Fa-f]{2,})b])')
    re_mal = re.compile(r'^([ERREUR|ECHEC|erreur|error|Error|ERROR|bKO\b|perdu[es]?|PERDUE?])')
    re_bon = re.compile(r'^([bOK\b|CRC_OK|bfinie\b|PLAUSIBLE|accept|ACTIF|tout tourne|Sync])')
    re_att = re.compile(r'^([attente|attend|retard|en retard|toujours pas|LOST|saute])')
    re_asmins = re.compile(r'^([s+[0-9a-f]+:s+)([0-9a-f]{8}|[0-9a-f]{4} [0-9a-f]{4}|[0-9a-f]{4} [0-9a-f]{4})')
    def generique(s):
        s = re_mal.sub(MAUVAIS + r'\1' + R, s)
        s = re_bon.sub(BON + r'\1' + R, s)
        s = re_att.sub(ATTENTION + r'\1' + R, s)
        s = re_fn.sub(r'\1' + GRAS + r'\2' + R, s)
        s = re_hex.sub(c(81) + r'\1' + R, s)
        return s
    for l in sys.stdin:
        if '\033[' in l:
            # deja colore (mobile.log) : tel quel

```

```

        out.write(l); out.flush(); continue
s = l.rstrip('\n'); t = s
m = re_adr.match(s)
if m: t = f'{ADR}{m.group(1)}{R}:{m.group(2)}{'
elif (m := re_objd.match(s)):
    t = f'{DIM}{m.group(1)}{R} {OCT}{m.group(2)}{R}'
elif (m := re_trace.match(s)):
    pc = m.group(4).lstrip('0') or '0'
    t = (f'{DIM}{m.group(1)}{R}{HOTE}{m.group(2)}{R} {DIM}[{m.group(3)}/{R}'
        f'{PC}{"0x" + pc.rjust(8, "0")}{R}{DIM}/{m.group(5)}/{m.group(6)}]{R} {SYM}'
    elif (m := re_stop.match(s)):
        t = f'{c(176)}{m.group(1)}{R}{DIM}{m.group(2)}{R}'
    elif s.startswith('IN:') or s.startswith('----'):
        t = f'{DIM}{s}{R}'
    elif (m := re_mbx.match(s)):
        t = (f'{DIM}{m.group(1)}{R}{SENS[m.group(2)]}{m.group(2)}{R}{m.group(3)}{'
            f'{ADR}{m.group(4)}{R}{SYM}{m.group(5)}{R}{generique(m.group(6))}'
    elif (m := re_asmins.match(s)):
        t = f'{DIM}{m.group(1)}{R}{OCT}{m.group(2)}{R}{m.group(3)}{PC}{m.group(4)}{R}{ge
    elif s.startswith('#'):
        t = f'{DIM}{s}{R}'
    else:
        m = re_tag.match(s)
        if m:
            col = TAGS.get(m.group(2).lower(), c(250))
            t = f'{m.group(1)}{col}[{m.group(2)}]{R}' + generique(s[m.end():])
        else:
            t = generique(s)
    out.write(t + '\n'); out.flush()

if __name__ == '__main__':
    signal.signal(signal.SIGPIPE, signal.SIG_DFL)
    try:
        principal()
    except (KeyboardInterrupt, BrokenPipeError):
        pass

```

0.3.75 /opt/GSM/qosmo-overlay/tools/rapport-run.sh

6838 octets, 131 lignes → 131 lignes

```

#!/bin/bash
set -u

_pid_qemu="$(pgrep -x qemu-system-arm 2>/dev/null | head -1)"
_live=""
if [ -n "$_pid_qemu" ] && [ -r "/proc/$_pid_qemu/environ" ]; then
    _live="$(tr '\0' '\n' < "/proc/$_pid_qemu/environ" 2>/dev/null | sed -n 's/^LOG_DIR=//p'
fi
# [2026-10-19] start-native.sh publie le chemin de mobile.log du run en cours
# (/run/osmo-banc/mobile-log, OSMO_MOBILE_LOG) : son dossier fait foi avant
# l'environnement de QEMU - en --dsp il n'y a pas de qemu-system-arm du fork,
# et les journaux sont sous /tmp/c54x-pont.
_pub="${OSMO_MOBILE_LOG:-$(cat /run/osmo-banc/mobile-log 2>/dev/null)}"
[ -n "$_pub" ] && [ -f "$_pub" ] && _live="$(dirname "$_pub")"
LIVE_DIR="${LOG_DIR:-${_live:-/tmp/calypso/logs}}"
L="${1:-$LIVE_DIR}"
SORTIE="${2:-}"
[ -n "$SORTIE" ] && exec > >(tee "$SORTIE")

q="$L/qemu.log"; o="$L/osmocon.log"; m="$L/mobile.log"
b="$L/bts.log"; p="${PONT_LOG:-/dev/shm/pont.log}"

n() { [ -r "${2:-}" ] && grep -icE -- "$1" "$2" 2>/dev/null | head -1 || echo 0; }
temoin() {
    [ -r "${2:-}" ] || return
    grep -iE -- "$1" "$2" 2>/dev/null | tail -1 \
        | sed -E 's/\x1b\[ [0-9;]*m//g; s/^[[:space:]]+//' | cut -c1-140
}
mes() {
    local lib="$1" mot="$2" fic="$3" v
    v="$(n "$mot" "$fic")"
    printf ' %-24s %7s' "$lib" "$v"
    if [ "$v" -gt 0 ] 2>/dev/null; then
        local t; t="$(temoin "$mot" "$fic)"; [ -n "$t" ] && printf ' | %s' "$t"
    else
        printf ' | motif jamais vu - absence NON prouvee'
    fi
    printf '\n'
}
titre() { printf '\n\033[1m== %s ==\033[0m\n' "$1"; }
info() { printf ' %-24s %s\n' "$1" "$2"; }

```

```

printf '\033[1m##### RAPPORT DE RUN - %s #####\033[0m\n' "$(date '+%Y-%m-%d %H:%M:
[ -r "$q" ] || { echo "ERREUR : $q illisible - mauvais repertoire ?"; exit 2; }
printf 'journaux : %s%s\n' "$L" "$([ "$L" = "$LIVE_DIR" ] && echo ' (run en cours)' || echo
printf 'derniere ecriture : %s\n' "$(stat -c %y "$q" 2>/dev/null | cut -d. -f1)"
if [ "$L" = "$LIVE_DIR" ] && [ -n "${_pid_qemu:-}" ]; then
    _t0="$(stat -c %s "$q" 2>/dev/null)"; sleep 2; _t1="$(stat -c %s "$q" 2>/dev/null)"
    if [ "${_t0:-0}" = "${_t1:-0}" ]; then
        printf '\033[1m    qemu.log NE GROSSIT PAS alors que qemu tourne (pid %s).\n' "$_pid_
        printf '        Journaux probablement perimes - verifier LOG_DIR du processus.\033[0m\n
    fi
fi

titre "0. IDENTITE DU RUN"
info "couche 1"      "$ (temoin '\[l1\] backend' "$q")"
info "pont"          "$ (temoin 'pont TRX démarré|pont TRX demarre' "$p")"
if [ "$L" = "$LIVE_DIR" ]; then
    P="$(pgrep -x qemu-system-arm 2>/dev/null | head -1)"
    [ -n "$P" ] && info "qemu vivant" "pid=$P age=$(ps -o etime= -p "$P" 2>/dev/null | tr -d
        || info "qemu vivant" "ARRETE"
    P="$(pgrep -f 'pont/pont(_dsp)?\.py' 2>/dev/null | head -1)"
    [ -n "$P" ] && info "pont vivant" "pid=$P age=$(ps -o etime= -p "$P" 2>/dev/null | tr -d
        || info "pont vivant" "ARRETE"
fi

titre "1. ACQUISITION L1 (osmocon.log)"
mes "L1CTL_FBSB_REQ"      'L1CTL_FBSB_REQ'          "$o"
mes "FB detectee"         '^FB[01] \('              "$o"
mes "Synchronize_TDMA"    'Synchronize_TDMA'         "$o"
mes "SB decodee"          '^SB[0-9] \(|=> SB 0x'       "$o"
mes "    BSIC non nul"     'BSIC=[1-9]'               "$o"
mes "RESET FULL"          'L1CTL_RESET_REQ: FULL'     "$o"
[ -r "$o" ] && info "TOA les plus vus" "$ (grep -oE 'TOA= *[0-9]+' "$o" | tr -s ' ' | sort |

titre "2. COUCHE 1 gr-gsm (qemu.log)"
mes "reset L1"            '\[l1\] reset L1'           "$q"
mes "canal dedie"         '\[l1\] canal dedie'        "$q"
mes "chiffrement actif"   '\[l1\] chiffrement A5'     "$q"
mes "canal TCH"           '\[l1\] canal TCH'          "$q"
mes "couche 1 en clair"   '\[l1\] couche 1 en clair'   "$q"

titre "3. PONT TRX (pont.log)"
mes "bursts DL"           'STATS dl_bursts='          "$p"

```

```

mes "RACH"                'rach=[1-9]'                "$p"
mes "erreurs"             'Traceback|error'            "$p"
[ -r "$p" ] && info "derniere STATS" "$(grep 'STATS' "$p" 2>/dev/null | tail -1 | cut -c1-14)

titre "4. COUCHES 2/3 (mobile.log) - jusqu ou va l abonne"
mes "sync ARFCN"          'Sync to ARFCN'              "$m"
mes "SYSTEM INFORMATION"  'SYSTEM INFORMATION [0-9]'    "$m"
mes " sans sysinfo"       'No sysinfo yet'             "$m"
mes "camp"                'camping \ (normal\)|camped normally' "$m"
mes "CHANNEL REQUEST"     'CHANNEL REQUEST'            "$m"
mes "IMMEDIATE ASSIGN"    'IMMEDIATE ASSIGNMENT'        "$m"
mes "LAPDm SABM"          'LAPD_STATE_SABM_SENT'        "$m"
mes "cipher negocie"      'cipher [1-9]'               "$m"
mes "LU REQUEST"          'LOCATION UPDATING REQUEST'     "$m"
mes "LU ACCEPT"           'LOCATION UPDATING ACCEPT'      "$m"
mes "LU REJECT"           'LOCATION UPDATING REJECT'      "$m"
mes "TMSI"                'TMSI'                      "$m"
mes "SMS"                 'SMS-DELIVER|new SMS|CP-DATA'  "$m"
mes "appel"               'CC_SETUP|MNCC_SETUP'         "$m"
[ -r "$m" ] && info "timers MM" "$(grep -oE 'T3[0-9]{3}' "$m" | sort | uniq -c | sort -rn | tr -s '\n' | cat)"
[ -r "$m" ] && info "ccch mode" "$(grep -oE 'ccch mode [A-Z]+' "$m" | sort | uniq -c | tr -s '\n' | cat)"

titre "5. RESEAU (bts / side-car)"
mes "BTS erreurs"         'ERROR'                      "$b"
[ -r "$b" ] && info "erreur dominante" "$(grep -oE 'send\(\\) failed on TRXD[^\(]*|ERROR [a-z0-9_]*' "$b" | sort | uniq -c | sort -rn | tr -s '\n' | cat)"
mes "sidecar mobile err"  'error|fail'                 "$L/sidecar-mobile.log"

titre "6. SERVICES"
mes "SMSC"                 'sms|deliver'                "$L/smsc-op1.log"
mes "asterisk warn/err"    'error|warning'              "$L/asterisk.log"

titre "7. VERDICT - jusqu ou va la chaine"
etape() {
    if [ "${2:-0}" -gt 0 ] 2>/dev/null; then printf ' [ OK ] %-22s (%s)\n' "$1" "$2"
    else printf ' [ .. ] %-22s (0)\n' "$1"; fi
}
etape "synchro FB"         "$(n '^FB[01] \(' '$o')"
```

```

etape "LU demande"          "$ (n 'LOCATION UPDATING REQUEST' "$m")"
etape "LU accepte"          "$ (n 'LOCATION UPDATING ACCEPT' "$m")"
etape "SMS"                  "$ (n 'SMS-DELIVER|new SMS|CP-DATA' "$m")"
printf '\n Methode : un 0 dit que le MOTIF n a pas ete vu. Verifier le motif\n'
printf '  avant de conclure a une absence.\n'

```

0.3.76 /opt/GSM/qosmo-overlay/tools/qosmo-tbring/tbring-dump.py

5007 octets, 110 lignes → 110 lignes

```

#!/usr/bin/env python3
"""tbring-dump : les N derniers blocs ARM executes, lus dans l'anneau du plugin tbring.

tbring-dump [-n N] [-f /dev/shm/qemu-tbring] [-e layer1.highram.elf] [-a qemu-asm.log] [-r]

-n N      nombre de blocs (defaut 200), les plus recents en dernier
-e ELF    symboles (nm) : fonction+decalage a cote de chaque PC
-a LOG    trace -d in_asm du meme run : les octets « OBJD-T » du bloc, desassemblees
          avec arm-none-eabi-objdump si present (ce build de QEMU n'a pas capstone)
-r        compte les blocs par PC sur toute la fenetre (histogramme) au lieu de la liste
-c/-C     couleurs forcees / coupees (defaut : si la sortie est un terminal)

Lecture a chaud, sans arreter QEMU : l'anneau est de la memoire partagee."""
import argparse, struct, subprocess, shutil, collections, os, sys, tempfile

def lire(path, n):
    with open(path, 'rb') as f:
        hdr = f.read(64)
        magic, entries, count, shift = struct.unpack('<8sQQQ', hdr[:32])
        if magic != b'TBRING01':
            sys.exit(f"{path}: pas un anneau tbring")
        n = min(n, entries, count)
        out = []
        for i in range(count - n, count):
            f.seek(64 + (i & (entries - 1)) * 8)
            pc, tour = struct.unpack('<II', f.read(8))
            if tour != ((i >> shift) & 0xffffffff):
                out.append((None, i))          # ecrase entre-temps (QEMU a fait un tour)
            else:
                out.append((pc, i))
        return entries, count, out

```

```

def symboles(elf):
    syms = []
    try:
        txt = subprocess.run(['nm', '-n', '-S', elf], capture_output=True, text=True, check=True)
    except Exception as e:
        print(f"nm {elf}: {e}", file=sys.stderr); return syms
    for l in txt.splitlines():
        p = l.split()
        if len(p) == 4 and p[2] in 'tTwW' and not p[3].startswith('$'):
            syms.append((int(p[0], 16), int(p[1], 16), p[3]))
    syms.sort()
    return syms

def resoudre(syms, pc):
    import bisect
    i = bisect.bisect_right([s[0] for s in syms], pc) - 1
    if i >= 0:
        a, sz, nom = syms[i]
        if pc < a + max(sz, 4):
            return f"{nom}+0x{pc - a:x}"
    return "?"

def in_asm(log):
    """pc -> (octets hex) pour chaque bloc traduit (premiere occurrence)."""
    table = {}
    pc = None
    with open(log, errors='replace') as f:
        for l in f:
            if l.startswith('0x') and l[10:11] == ':':
                try: pc = int(l[:10], 16)
                except ValueError: pc = None
            elif l.startswith('OBJD-T:') and pc is not None and pc not in table:
                table[pc] = l.split(':', 1)[1].strip()
                pc = None
    return table

def desasm(pc, hexs):
    od = shutil.which('arm-none-eabi-objdump')
    if not od:
        return [f".word {hexs}"]
    with tempfile.NamedTemporaryFile(suffix='.bin', delete=False) as t:
        t.write(bytes.fromhex(hexs)); tmp = t.name

```

```

try:
    r = subprocess.run([od, '-D', '-b', 'binary', '-m', 'arm', '-M', 'force-thumb' if pc
                        f'--adjust-vma=0x{pc & ~1:x}', tmp], capture_output=True, text=True)
    return [' ' + l.strip() for l in r.stdout.splitlines() if l[:1] == ' ' and ':' in l]
finally:
    os.unlink(tmp)

ap = argparse.ArgumentParser(description=__doc__, formatter_class=argparse.RawDescriptionHelpFormatter)
ap.add_argument('-n', type=int, default=200); ap.add_argument('-f', default='/dev/shm/qemu-tmp')
ap.add_argument('-e'); ap.add_argument('-a'); ap.add_argument('-r', action='store_true')
ap.add_argument('-c', action='store_true'); ap.add_argument('-C', action='store_true')
a = ap.parse_args()
COUL = (a.c or (sys.stdout.isatty() and os.environ.get('QOSMO_COULEURS', '1') != '0')) and not a.C
def K(n, s, g=''):
    return f"\033[{g}38;5;{n}m{s}\033[0m" if COUL else str(s)
entries, count, blocs = lire(a.f, (1 << 62) if a.r and a.n == 200 else a.n) # -r sans -n :
syms = symboles(a.e) if a.e else []
table = in_asm(a.a) if a.a else {}
print(K(242, f"# {a.f} : {count} blocs executes, anneau de {entries}, {len(blocs)} affiches"))
if a.r:
    h = collections.Counter(pc for pc, _ in blocs if pc is not None)
    for pc, k in h.most_common(60):
        print(f"{K(221, f'{k:10d}', '1;')} {K(81, f'0x{pc:08x}', '1;')} {K(114, resoudre(syms, pc))}")
else:
    for pc, i in blocs:
        if pc is None:
            print(K(203, f"{i:12d} (ecrase)")); continue
        print(f"{K(242, f'{i:12d}')} {K(81, f'0x{pc:08x}', '1;')} {K(114, resoudre(syms, pc))}")
        if pc in table:
            for l in desasm(pc, table[pc]):
                p = l.split(None, 2) # adresse: octets mnemonique operandes
                if len(p) == 3 and COUL:
                    m = p[2].split(None, 1)
                    print(f" {K(242, p[0])} {K(242, p[1])}\t{K(221, m[0], '1;')} {m[1] if m[1] != '' else ''}")
                else:
                    print(l)

```