



CPAL bytecode formatı (.cpalb): v1.0

Axion Robot *CircuitPal V2*

Özet—Bu belge CircuitPal V2 eğitim robotunun teknik sözleşmelerinden birini IEEE dergi biçiminde sunar. İçerik depodaki Markdown kaynağından üretilmiştir; kaynak ile bu çıktı arasında bir fark görülürse depo kazanır.

Index Terms—CircuitPal, ESP32-S3, eğitim robotu, gömülü yazılım, Axion Robot

Sahibi: WS-B · Faz 3 · Yürüten: firmware/
components/cpal_vm · Üreten: tools/cpalc
Opcode anlamları: tools/opcodes/opcodes.
yaml ((2)). Bu doküman **kabı** tanımlar.

I. DOSYA YERLEŞİMİ (LITTLE-ENDIAN, HIZALAMA YOK)

Toplam uzunluk `header_len + pool_len + code_len + 4` olmalı; tutmazsa `BAD_LAYOUT`.
`setup_off`, `loop_off` ve tüm sıçrama hedefleri `< code_len` ve bir komut başlangıcı.

TABLO 1

Ofset	Boyut	Alan	Değer / kural
0	4	magic	"CPAL"
4	1	format_major	1, uyuşmazsa BAD_FORMAT
5	1	format_minor	0, geriye uyumlu eklemeler için
6	2	flags	0 (rezerve)
8	4	opcode_table_hash	opcodes.yaml kanonik hash'i (§3), uyuşmazsa BAD_HASH ("robotun yazılımı editörden eski")
12	2	pool_len	sabit havuzu bayt sayısı ($\leq$ CPAL_VM_MAX_POOL_BYTES)
14	2	code_len	kod bayt sayısı ($\leq$ CPAL_VM_MAX_CODE_BYTES)
16	2	setup_off	cpal_setup giriş ofseti (kod içinde)
18	2	loop_off	cpal_loop giriş ofseti (kod içinde)
20	1	n_vars	kullanılan değişken sayısı ($\leq$ CPAL_VM_MAX_VARS)
21	1	max_stack	beyan edilen yığın derinliği ($\leq$ CPAL_VM_MAX_STACK)
22	2	header_len	24, ileride minor sürümde alan eklenirse büyür; VM header_len $\geq$ 24 kabul eder, fazlasını atlar
header_len	pool_len	sabit havuzu	\0 sonlu metinler art arda; pool_len > 0 $\Rightarrow$ pool[pool_len-1] == 0 (zorunlu); PUSH_STR imm = havuz bayt ofseti
header_len+pool	code_len	kod	opcode akışı; son komut HALT, LOOP_END ya da JMP olmalı
son 4	4	crc32	IEEE CRC-32, başlık+havuz+kod üstünden (kendisi hariç)

II. KOD AKIŞI

- Her komut 1 bayt opcode + (varsa) immediate: i32 4 bayt, u16/i16 2 bayt, u8 1 bayt.
- Yığın: 32-bit işaretli tamsayı. bool 0/1. Enum'lar tamsayı değeri.
- API opcode'ları argümanları **sondan başa** çeker (son itilen = son parametre), dönüş varsa iter. Aralık dışı argüman **klemplenir** (çocuk API'si; hata değil). blocking opcode'lar ilgili cpal_* fonksiyonunu **doğrudan** çağırır; bekleme fonksiyonun içindedir (çağrı geçişi, ADR-027). halts_vm (SHUTDOWN) çağrı dönerse VM HALTED.
- JMP/JZ/JNZ imm:i16, hedef = (komutun başlangıç ofseti) + imm. Geriye sıçrama (**imm** $\leq$ 0, kendine sıçrama dahil, cpalc'nin boş gövdeli while true'su JMP 0'dır) alındığında K1'in geri-sıçrama sayacını artırır.
- LOOP_END: bir **zamanlama yuvası** biter (= C'de cpal_loop() dönüşü): heartbeat + tick'i bırak; sürücü CPAL_TICK_MS iletir, cpal_runtime_tick() çağırır, VM loop_off'tan sürer. Bütçe/geri-sıçrama bırakmaları saat **ilerletmez**. setup bölümü loop_off'a **akarak** girer ya da JMP ile atlar; cpalc her zaman setup ... JMP loop üretir.
- HALT: program biter (VM HALTED).

III. OPCODE_TABLE_HASH: KANONİK HESAP

opcodes_gen.py: {format, enums: {tip: {AD: değer}}, ops: [...]} ops = core+api'nin **id'ye göre** sıralı listesi, her biri yalnız **anlam taşıyan** alanlar (id, name, imm, jump, halts_vm, deprecated, args[name,type,min,max], returns, blocking). json.dumps(sort_keys=True, separators=(",", ":"), ensure_ascii=True) $\rightarrow$ **FNV-1a 32-bit**. Hash'e **girmeyenler**: blok etiketi/tooltip/renk/kategori/unit/since/doc, enum label/ui_hidden, yaml'daki satır sırası ve biçim. Hash'e **girenler**: enum ad $\rightarrow$ değer tabloları (yeniden numaralama

yüklü programı kırar, hash yakalar), argüman aralıkları (klemp değişimi anlamdır). Aynı sayı cpal_opcodes.h'te CPAL_OPCODE_TABLE_HASH, opcodes.json'da hash; VM cpal_vm_opcode_table_hash() ile döndürür, cpal_link HELLO (Faz 4) taşır.

IV. YÜKLEME DOĞRULAMASI (VM, CPAL_VM_LOAD)

Sırayla: magic $\rightarrow$ format_major $\rightarrow$ header_len $\geq$ 24 + uzunluk tutarlılığı $\rightarrow$ CRC $\rightarrow$ hash $\rightarrow$ sınırlar $\rightarrow$ havuz \0 sonu $\rightarrow$ **statik tarama**: her opcode tanımlı ve yaşıyor mu (CPAL_OP_INFO[.valid), immediate kod içinde mi, her sıçrama hedefi (is_jump) < code_len ve komut başlangıcı mı, setup_off/loop_off aynı şart, son komut HALT/LOOP_END/JMP mı, PUSH_STR imm < pool_len mı, LOAD/STORE imm < n_vars mı. Herhangi biri düşerse yükleme **sebebiyle** reddedilir; program hiç koşmaz.

Hata kodu eşlemesi (VM'in uyguladığı): len < 4 (magic okunamaz) $\rightarrow$ BAD_LAYOUT; magic tutmuyorsa $\rightarrow$ BAD_MAGIC; magic geçtikten **sonra** len < 28 (başlık+CRC okunamaz) $\rightarrow$ BAD_LAYOUT; LOAD/STORE imm $\geq$ n_vars $\rightarrow$ BAD_LAYOUT; setup_off/loop_off komut ortası $\rightarrow$ BAD_LAYOUT; tanımsız/deprecated/rezerve opcode ya da immediate kod sonunu aşıyor $\rightarrow$ BAD_OPCODE; sıçrama hedefi kod dışı/komut ortası $\rightarrow$ BAD_JUMP; CPAL_VM_MAX_* aşımı $\rightarrow$ TOO_BIG. cpalc'nin negatif testleri aynı kodları bekler.

Koşum zamanı ağı (statik tarama yetmez): havuz ofseti yığından da gelebilir (PUSH_I32 x; PRINT). Üretilen dispatch str argümanını çağrıdan **önce** VM_STR() ile doğrular; sınır dışı $\rightarrow$ FAULT_BAD_POOL_INDEX, cpal_* **hiç çağrılmaz**. Yığın alt/üst taşması, LOAD/STORE indeks (yükleyici zateneledi ama VM yine kontrol eder), pc $\geq$ code_len $\rightarrow$ HALTED (güvenlik ağı).

V. KOŞUM SINIRLARI (K1 ÜÇ KATMAN)

TABLO 2

Katman	Sınır	Davranış
Komut bütçesi	CPAL_VM_BUDGET_PER_TICK = 200 (ölçülecek)	tick biter, sonraki tick sürer
Geri sıçrama	CPAL_VM_MAX_BACK_JUMPS = 32 / tick	tick zorla bırakılır, cpal_vm_runaway_count() artar, program sürer (FAULT değil), while(true){} sistemi kilitlemez
Yığın/boyut	başlıkta beyan, yüklemede sınır	koşumda taşma FAULT_STACK_* → VM FAULT, motorlar kesilir (runtime)

VI. SÜRÜMLEME

- format_major artışı = eski VM'ler reddeder.
format_minor artışı = header_len büyür, eski VM bilmediği başlık alanlarını atlar (flags bitleri de minor kapsamı).
- Opcode ekleme table_version + hash değiştirir;
robot güncellenmeden yeni program yüklenemez, editör HELLO'da fark edip kullanıcıya söyler (K1).