

BM1720 Engineering Notebook — 4-Chip Bring-Up Reference

Scope	Bare-metal bring-up of a 4-chip BM1720 board: PIC16F1704 power gating over I2C, chain enumeration over UART, PLL, open-core, work/nonce exchange, nonce verification, convoy stratum job construction.
Validation basis	2026-08-31. Live Antminer A3 hashboard, 60× BM1720, one chain per board, driven by a reconstructed driver over a hardware bridge (on-board bridge binary over SSH). Chains 0/1/2 exercised.
4-chip status	No 4-chip board has been exercised. Every 4-chip parameter in this notebook is an instantiation of a 60-chip-validated formula and carries the DERIVED tag (§14).

Tag	Meaning
SILICON ✓	Validated 2026-08-31 against a real A3 hashboard (60× BM1720) by driving a reconstructed driver over a hardware bridge.
FROM FW	Reverse-engineered from the stock A3 cgminer. Consistent with silicon behaviour; not independently silicon-exercised.
OPEN	Unresolved.
DERIVED	Computed in this notebook from tagged facts: a CRC from the stated algorithm, a formula instantiated for 4 chips, unit arithmetic. Not a new observation; inherits the weakest tag of its inputs. Every DERIVED CRC5/CRC16 below was produced by the §6 pseudocode after that pseudocode reproduced the three SILICON ✓ command frames and the CRC16 test vector.

- | | |
|---|---|
| 1. Device summary & validation status | 8. Open-core sequence |
| 2. Hashboard connector pinout (18-pin) | 9. Clock / PLL table |
| 3. Power architecture & PIC16F1704 | 10. Work frame & header layout |
| 4. PIC16F1704 I2C command recipe | 11. Nonce return frame & verified hash convention |
| 5. Chain UART protocol: framing, VIL frame, command set, register map | 12. Stratum, extranonce2, merkle root (convoy) |
| 6. CRC5 and CRC16 algorithms + test vectors | 13. Measured performance |
| 7. Chain bring-up sequence | 14. Host portability + 4-chip specifics |
| | 15. Open items |

1. Device summary & validation status

Item	Value
Part	BM1720 — Blake2b-256 proof-of-work ASIC.
Stock application	Bitmain Antminer A3: 3 hashboards, 60 chips per board, one private UART chain per board.
Target here	4 chips, one chain, one PIC16F1704 (or a directly-enabled DC-DC), one UART.
Chain link	UART 115200 8N1 raw (57600 also accepted). Host → chain frames prefixed 55 AA; chain → host frames prefixed AA 55.
Power gating	On-board PIC16F1704 over I2C, 7-bit 0x50 for chain 0. ASICs are silent on UART until the PIC enables the DC-DC.
Hash computed by chip	Blake2b-256 over exactly the 80 bytes on the wire (per-32-bit-word byte-swapped header). 17,702 of 17,702 returned nonces reproduced (§11).
Nonce field	Chip enumerates the 32-bit value at logical header [32:36]; host half [36:40] held at 0.
PLL reference	25 MHz; word written big-endian to reg 0x0c; encoding checked on silicon at 500 MHz.

Validation status

Item	Evidence	Tag
PIC16F1704 power-up over I2C (jump → get_version → set_voltage → enable DC-DC)	Enable reply 15 2a (byte 0 = 0x15 echo = OK). Version read-back 05 0a 14 28 50.	SILICON ✓
Chain enumeration (CHAIN_INACTIVE + SET_ADDRESS)	60 chips each answered READ_REG(CHIP_ADDR) at their assigned address (0, 4, 8, ...).	SILICON ✓
Register read (READ_REG)	60 valid-CRC5 response frames.	SILICON ✓
Set frequency (WRITE_REG 0x0c)	Accepted; chain stable.	SILICON ✓
Open-core	Byte-exact from a live stock-cgminer UART capture; chips return nonces after it.	SILICON ✓
Send work → return nonces	Returned nonces are genuine Blake2b solutions: 17,702 / 17,702 reproduced by the verifier; best observed depth 46 leading zero bits.	SILICON ✓
Convoy stratum (subscribe / authorize / notify / target parse, 80-byte header, Blake2b pipeline)	Live jobs mined.	SILICON ✓
Multi-chain	Chains 0/1/2 (tty01/tty02/tty04, PICs 0x50/0x51/0x52) brought up and hashed in parallel.	SILICON ✓
Accepted share on a specific pool	Out of scope for chip/driver bring-up. Share difficulty and submit-field serialization are pool policy, not device behaviour (§12).	—
CRC5 algorithm	Reproduces every captured command frame and validates the 60 response frames.	FROM FW silicon-consistent
CRC16 algorithm	CCITT-FALSE; work frames built with it are accepted (nonces returned).	FROM FW
PLL table (125 entries)	Encoding silicon-checked at 500 MHz only.	FROM FW
Registers 0x28 / 0x2c function	Written 0x00000000 during bring-up; function unconfirmed.	OPEN
Host nonce half [36:40]	Silicon did not reliably accept a nonzero host half in the tested profile.	OPEN
4-chip board	Not exercised. All 4-chip values are formula instantiations.	DERIVED

2. Hashboard connector pinout (18-pin) SILICON ✓

2×9, 2.00 mm pitch. Same signal connector as the Bitmain S9/L3+/D3/A3 generation.

Pin	Signal	Pin	Signal
1	GND	2	GND
3	SDA (I2C to PIC)	4	SCL (I2C to PIC)
5	PLUG0 (board-present)	6	A2 (PIC I2C address strap)
7	A1 (PIC I2C address strap)	8	A0 (PIC I2C address strap)
9	GND	10	GND
11	TXD (UART to chain)	12	RXD (UART from chain)
13	GND	14	GND
15	RST (chain reset, active low)	16	D3V3 (3.3 V in; powers the PIC)
17	NC	18	NC

- Pin 16: 3.3 V is supplied *by the controller into the board* to power the PIC. It is not derived from the board's 12 V rail.
- A0/A1/A2 strap the PIC I2C address. A3 chains 0–3 sit at 7-bit 0x50 / 0x51 / 0x52 / 0x53 (8-bit write addresses 0xA0 / 0xA2 / 0xA4 / 0xA6).
- SDA/SCL require pull-ups. The OEM control board provides them; a retrofit adapter must add them (§14).

3. Power architecture & PIC16F1704

Rail / element	Behaviour	Tag
12 V from PSU	Feeds the board's per-domain DC-DC converters (ASIC core voltage).	SILICON ✓
DC-DC enable	Gated by the on-board Microchip PIC16F1704, commanded over I2C (§4). Until the PIC enables the DC-DC the ASICs have no core voltage and are silent on UART.	SILICON ✓
PIC supply	3.3 V supplied by the controller on connector pin 16.	SILICON ✓
PIC I2C address	7-bit 0x50 + strap value (A0/A1/A2, pins 8/7/6). Chain 0 = 0x50.	SILICON ✓
Power hold	Heartbeat command every ≤10 s, otherwise the PIC latches power off.	SILICON ✓
Fan interlock	Stock firmware's software fan-fault check disables the PIC on low fan RPM. Keep the fan running. A retrofit host with its own PIC firmware may not enforce this; the OEM board does.	FROM FW
Core voltage default	Code 0x78 ≈ 815 mV (factory default).	SILICON ✓

4. PIC16F1704 I2C command recipe SILICON ✓

Frame format

55 AA <len> <cmd> <data...> <chk>

len = byte count following 55 AA, including len and chk itself
(4 for frames with one data byte, 5 for frames with two) [DERIVED: matches all seven frames]

chk = (len + cmd + data[0] + data[1] + ...) & 0xFF # 55 AA excluded

Transaction: I2C write of the whole frame to 7-bit 0x50 (chain 0) → wait ~100 ms → I2C read-back. Byte 0 of the reply echoes the command byte on success (observed on enable DC-DC: reply 15 2a). The observed get_version reply 05 0a 14 28 50 does not begin with 0x17; it is recorded here exactly as read.

Op	Frame (hex, on the wire)	Reply observed	Notes
jump loader → app	55 AA 04 06 00 0A	—	Send first, before anything else.
reset	55 AA 04 07 00 0B	—	
get_version	55 AA 04 17 00 1B	05 0a 14 28 50	5-byte read-back.
set_voltage	55 AA 05 10 78 00 8D	—	Code 0x78 ≈ 815 mV, factory default. Higher code = lower mV.
enable DC-DC	55 AA 05 15 01 00 1B	15 2a	Data byte 01 = ON. Reply byte 0 0x15 = OK.
disable DC-DC	55 AA 05 15 00 00 1A	—	Data byte 00 = OFF. Safe shutdown.
heartbeat	55 AA 04 16 00 1A	—	Every ≤10 s or the PIC latches power off.

Power-up order

1. jump loader → app (55 AA 04 06 00 0A)
 2. get_version (55 AA 04 17 00 1B), read 5 bytes; expect 05 0a 14 28 50
 3. set_voltage (55 AA 05 10 78 00 8D)
 4. enable DC-DC (55 AA 05 15 01 00 1B), read back; byte 0 must be 0x15
 5. heartbeat (55 AA 04 16 00 1A) every ≤10 s for as long as the board must stay powered
- ~100 ms between each command and its read-back.

Voltage code

mV = (1609.927 – code) / 1.827 # code = data byte of set_voltage

code = 1609.927 – 1.827 × mV # [DERIVED] inverse

0x78 (120) → 815.5 mV # factory default; only code exercised in the source

Enable and disable differ only in the data byte (01 vs 00). Sending 55 AA 05 15 00 00 1A powers the board down.

5. Chain UART protocol

Physical / serial SILICON ✓

Parameter	Value
Baud	115200 (57600 also accepted). 8N1, raw.
Topology	One private point-to-point UART per chain. TXD pin 11 (host → chain), RXD pin 12 (chain → host).
Host → chain prefix	55 AA before every command frame and every work frame.
Chain → host prefix	AA 55 before every return frame (§11).
A3 control board devices	Chains 0–3 = /dev/tty01, /dev/tty02, /dev/tty04, /dev/tty05. O1/O2/O4 verified; O5 not exercised.
Inter-frame timing	Not captured in the source. The validated artefact is the frame order (§7, §8).

Command ("VIL") frame — host → chain

Offsets are counted after the 55 AA prefix. All multi-byte fields big-endian.

Offset	Width	Field	Value
0	1	byte0	0x40 (0x10 if broadcast else 0) (cmd & 0x0F)
1	1	flen	Total frame length excluding the prefix = 4 + len(payload) + 1. 0x05 without payload, 0x09 with a 32-bit register value.
2	1	chip_address	Target chip address. 0x00 when broadcast.
3	1	reg	Register address for WRITE_REG / READ_REG. 0x00 in the captured SET_ADDRESS and CHAIN_INACTIVE frames.
4 ... flen-2	0 or 4	payload	Register value, big-endian (WRITE_REG only).
flen-1	1	crc5	§6 CRC5 over the first (flen-1)×8 bits, starting at byte0. The 55 AA prefix is not covered.

Command set

cmd nibble	Name	byte0 unicast	byte0 broadcast	Payload	flen
0x0	SET_ADDRESS	0x40	—	none; chip_address = address being assigned	0x05
0x1	WRITE_REG	0x41	0x51	4 bytes, big-endian register value	0x09
0x2	READ_REG	0x42	0x52	none	0x05
0x3	CHAIN_INACTIVE	—	0x53	none	0x05

Frame examples (CRC5 included, after 55 AA)

Meaning	Frame	Tag
SET_ADDRESS, assign address 0x00	40 05 00 00 1C	SILICON ✓
CHAIN_INACTIVE (broadcast)	53 05 00 00 03	SILICON ✓
WRITE_REG 0x0c = 0x00500221 (500 MHz PLL, broadcast)	51 09 00 0C 00 50 02 21 0C	SILICON ✓
READ_REG 0x00 (broadcast probe)	52 05 00 00 0A	SILICON ✓ body; DERIVED CRC
READ_REG 0x00 unicast to chip 0x00	42 05 00 00 0E	DERIVED
READ_REG 0x00 unicast to chip 0x40	42 05 40 00 08	DERIVED
WRITE_REG 0x14 = 0x00000020 (broadcast)	51 09 00 14 00 00 00 20 0D	DERIVED
WRITE_REG 0x1c = 0x07007A61 unicast to chip 0x10	41 09 10 1C 07 00 7A 61 14	DERIVED CRC; value/target SILICON ✓

Register map

Reg	Name	Use	Tag
0x00	CHIP_ADDR	Per-chip address. READ_REG(0x00) enumerates the chain.	SILICON ✓
0x08	HASH_RATE	Status.	FROM FW
0x0c	PLL_PARAM	Frequency. Value = vilpll word zero-extended to 32 bits, big-endian (§9).	SILICON ✓
0x14	TICKET_MASK	Nonce-return threshold. Stock bring-up writes 0x00000020.	SILICON ✓ (value as captured)
0x1c	MISC_CONTROL	Baud/config + core-gate enable bits. Bring-up value 0x07007A61 (§7 step 8).	SILICON ✓
0x20	GENERAL_I2C	Temp-sensor / on-chip I2C passthrough.	FROM FW
0x28	—	Written 0x00000000 during bring-up. Function unconfirmed.	OPEN
0x2c	—	Written 0x00000000 during bring-up. Function unconfirmed.	OPEN
0x40	CORE_MUX_SELECT	Temp / VDD mux.	FROM FW
0x50 0x54 0x58 0x5c	Open-core gate	Four gate registers. Written with the growing 32-bit mask during open-core (§8).	SILICON ✓

6. CRC5 and CRC16 algorithms + test vectors

CRC5 FROM FW silicon-consistent

Polynomial x^5+x^2+1 , initial state all ones, MSB-first over `bit_len` bits. Command frames: `bit_len = (flen-1)×8`. Nonce-return frames: `bit_len = 51`.

```
st = [1,1,1,1,1] # st[0..4]
for i in 0 .. bit_len-1:
    din = (data[i//8] >> (7 - (i%8))) & 1
    fb = din ^ st[4]
    st[4], st[3], st[2], st[1], st[0] = st[3], st[2], st[1]^fb, st[0], fb
crc5 = (st[4]<<4) | (st[3]<<3) | (st[2]<<2) | (st[1]<<1) | st[0]
```

Input bits (hex, MSB-first)	bit_len	CRC5	Tag
40 05 00 00	32	0x1C	SILICON ✓ (SET_ADDRESS 0x00)
53 05 00 00	32	0x03	SILICON ✓ (CHAIN_INACTIVE)
51 09 00 0C 00 50 02 21	64	0x0C	SILICON ✓ (PLL 500 MHz)
52 05 00 00	32	0x0A	DERIVED
12 34 56 78 00 05 00	51	0x0A	DERIVED synthetic nonce-frame body; bits 51–55 ignored
12 34 56 78 00 05 80	51	0x1E	DERIVED same body with the bit-7 flag set (bit 48 is inside the span)

CRC16 FROM FW — CRC-16/CCITT-FALSE

Init 0xFFFF, polynomial 0x1021, no input reflection, no output reflection, no final XOR. Work frames: over bytes 0–81 inclusive (opcode, `work_id`, 80 header bytes), result stored big-endian at bytes 82–83.

```
crc = 0xFFFF
for b in data:
    crc ^= b << 8
    repeat 8:
        crc = ((crc << 1) ^ 0x1021) & 0xFFFF if (crc & 0x8000)
            else (crc << 1) & 0xFFFF
```

Input	CRC16	Tag
"123456789" (ASCII, 9 bytes)	0x29B1	FROM FW standard check value
20 01 + 80 × 00 (open-core dummy work, pass 1)	0xF593	DERIVED
20 20 + 80 × 00 (open-core dummy work, pass 32)	0xC89B	DERIVED

7. Chain bring-up sequence SILICON ✓ order, captured live from stock cgminer via UART man-in-the-middle

Precondition: PIC power-up complete (§4), heartbeat running. Frames below are shown after 55 AA. Step 3 and step 8 address modes are explicit in the capture (broadcast, unicast to 0x10); steps 2, 6, 7 are shown broadcast — the source does not state their address mode.

Step	Command	Frame(s), 4-chip board	Tag / notes
1	READ_REG 0x00, broadcast (probe)	52 05 00 00 0A	Any valid-CRC5 reply (§11) proves the chain is powered and the UART is wired.
2	WRITE_REG 0x14 TICKET_MASK = 0x00000020	51 09 00 14 00 00 00 20 0D	Value SILICON ✓ ; CRC DERIVED .
3	WRITE_REG 0x0c PLL_PARAM = vilpll(target MHz), broadcast	51 09 00 0C 00 50 02 21 0C	500 MHz shown, SILICON ✓ . Other frequencies: §9, recompute CRC5.
4	CHAIN_INACTIVE, broadcast	53 05 00 00 03	SILICON ✓
5	SET_ADDRESS × chip_count, interval 256 / next_pow2(chip_count); k-th frame assigns address k × interval	40 05 00 00 1C 40 05 40 00 1A 40 05 80 00 10 40 05 C0 00 16	4 chips → interval 64 DERIVED . 60 chips → interval 4: 40 05 00 00 1C, 40 05 04 00 03, 40 05 08 00 07, ... SILICON ✓ (all 60 answered at 0, 4, 8, ...).
6	WRITE_REG 0x28 = 0x00000000	51 09 00 28 00 00 00 00 1D	Register function OPEN . Write it anyway; it is in the working capture.
7	WRITE_REG 0x2c = 0x00000000	51 09 00 2C 00 00 00 00 10	Register function OPEN . Same.
8	WRITE_REG 0x1c MISC_CONTROL = 0x07007A61, unicast to chip 0x10	41 09 10 1C 07 00 7A 61 14	Value and target SILICON ✓ on the 60-chip board, where 0x10 is chip index 4. A 4-chip board has no chip at 0x10 (addresses 0x00/0x40/0x80/0xC0): correct target for 4 chips is OPEN (§15).
9	Temp-sensor init: writes to CORE_MUX_SELECT 0x40 and GENERAL_I2C 0x20	values not in source	Present in the stock sequence; the written values were not included in the source data. Whether hashing depends on it is not established (§15).
10	OPEN CORE	§8	SILICON ✓ byte-exact.
11	Real work	§10	Nonces return after step 10 only.

MISC_CONTROL must be 0x07007A61. The value is from the silicon capture and is the one that opens the cores. A prior static-analysis guess, 0x07003A01, selects a different code path and does *not* open cores — do not use it.

8. Open-core sequence SILICON ✓ byte-exact

Enumerated cores boot gated OFF. Ungate with 32 passes of a growing bitmask written to the four gate registers, plus one dummy work frame per pass.

```
mask = 0
for pass in 1 .. 32:
    mask = ((mask << 1) | 1) & 0xFFFFFFFF      # 0x1, 0x3, 0x7, ..., 0xFFFFFFFF
    WRITE_REG 0x50 = mask      (broadcast)      # 55 AA 51 09 00 50 <mask BE> <crc5>
    WRITE_REG 0x54 = mask      (broadcast)      # 55 AA 51 09 00 54 <mask BE> <crc5>
    WRITE_REG 0x58 = mask      (broadcast)      # 55 AA 51 09 00 58 <mask BE> <crc5>
    WRITE_REG 0x5c = mask      (broadcast)      # 55 AA 51 09 00 5C <mask BE> <crc5>
    send dummy work: opcode 0x20, work_id = pass & 0x7F, 80 header bytes = 0x00, CRC16 (§10)
    read and discard anything the chain returns
```

Per-pass CRCs DERIVED

Register frame per pass: 51 09 00 <reg> <mask, 4 bytes BE> <CRC5>. Dummy work frame per pass: 20 <pass> + 80 × 00 + <CRC16 hi> <CRC16 lo>. Each frame on the wire is preceded by 55 AA.

Pass	mask (BE bytes)	CRC5 @0x50	CRC5 @0x54	CRC5 @0x58	CRC5 @0x5C	work_id	CRC16 dummy
1	00 00 00 01	1A	17	00	0D	01	F5 93
2	00 00 00 03	10	1D	0A	07	02	41 9C
3	00 00 00 07	04	09	1E	13	03	DD 86
4	00 00 00 0F	09	04	13	1E	04	39 A3
5	00 00 00 1F	13	1E	09	04	05	A5 B9
6	00 00 00 3F	02	0F	18	15	06	11 B6
7	00 00 00 7F	05	08	1F	12	07	8D AC
8	00 00 00 FF	0B	06	11	1C	08	C9 DD
9	00 00 01 FF	17	1A	0D	00	09	55 C7
10	00 00 03 FF	0A	07	10	1D	0A	E1 C8
11	00 00 07 FF	15	18	0F	02	0B	7D D2
12	00 00 0F FF	0E	03	14	19	0C	99 F7
13	00 00 1F FF	1D	10	07	0A	0D	05 ED
14	00 00 3F FF	1E	13	04	09	0E	B1 E2
15	00 00 7F FF	18	15	02	0F	0F	2D F8
16	00 00 FF FF	14	19	0E	03	10	39 00
17	00 01 FF FF	0C	01	16	1B	11	A5 1A
18	00 03 FF FF	19	14	03	0E	12	11 15
19	00 07 FF FF	16	1B	0C	01	13	8D 0F
20	00 0F FF FF	08	05	12	1F	14	69 2A
21	00 1F FF FF	11	1C	0B	06	15	F5 30
22	00 3F FF FF	06	0B	1C	11	16	41 3F
23	00 7F FF FF	0D	00	17	1A	17	DD 25
24	00 FF FF FF	1B	16	01	0C	18	99 54
25	01 FF FF FF	12	1F	08	05	19	05 4E
26	03 FF FF FF	00	0D	1A	17	1A	B1 41
27	07 FF FF FF	01	0C	1B	16	1B	2D 5B
28	0F FF FF FF	03	0E	19	14	1C	C9 7E
29	1F FF FF FF	07	0A	1D	10	1D	55 64
30	3F FF FF FF	0F	02	15	18	1E	E1 6B
31	7F FF FF FF	1F	12	05	08	1F	7D 71
32	FF FF FF FF	1A	17	00	0D	20	C8 9B

Pass 1, complete wire bytes (each line is one write):

[illegible]

Pass 32 register frames: 55 AA 51 09 00 50 FF FF FF FF 1A, ... 54 FF FF FF FF 17, ... 58 FF FF FF FF 00, ... 5C FF FF FF FF 0D; dummy work 20 20 + 80×00 + C8 9B.

9. Clock / PLL table FROM FW — encoding silicon-checked at 500 MHz

Encoding

Bits of vilpll	Field	Value
23:16	fbdiv	8-bit feedback divider
15:8	refdiv	Always 0x02 in this table
7:4	postdiv1	4-bit
3:0	postdiv2	4-bit

$$\text{freq MHz} \approx 25 \times \text{fbdiv} / (\text{refdiv} \times \text{postdiv1} \times \text{postdiv2}) \quad \# 25 \text{ MHz reference}$$

```

register 0x0c value (32-bit, big-endian on the wire) = vilpll zero-extended: 00 <fbdiv> <refdiv> <p
500 MHz: vilpll 0x500221 → 00 50 02 21 → frame 51 09 00 0C 00 50 02 21 0C [SILICON ✓]

```

MHz	vilpll	fbdiv	refdiv	postdiv1	postdiv2	25 × fbdiv / (refdiv × pd1 × pd2)	Tag
100	0x400242	64	2	4	2	100.00	DERIVED decode
294	0x2f0221	47	2	2	1	293.75	DERIVED decode
500	0x500221	80	2	2	1	500.00	SILICON ✓ accepted, chain stable
825	0x420211	66	2	1	1	825.00	DERIVED decode

Full table, 125 entries, 100–825 MHz (MHz → vilpll)

Column-major: read down each MHz/vilpll pair. Pairs 437/438, 587/588 and 637/638 share a word in the source and are kept as listed. All 125 entries decode with $\text{refdiv} = 2$ to within 1 MHz of the stated value DERIVED check.

MHz	vilpll	MHz	vilpll	MHz	vilpll	MHz	vilpll	MHz	vilpll
100	0x400242	248	0x770232	381	0x3d0221	512	0x520221	656	0x690221
106	0x440242	254	0x7a0232	384	0x7b0241	519	0x530221	662	0x6a0221
112	0x480242	260	0x7d0232	387	0x5d0231	525	0x540221	668	0x6b0221
118	0x420271	267	0x6b0251	391	0x7d0241	531	0x550221	675	0x6c0221
125	0x460271	273	0x6d0251	394	0x3f0221	537	0x560221	681	0x6d0221
131	0x3f0261	279	0x430231	397	0x7f0241	544	0x570221	687	0x6e0221
137	0x420232	285	0x720251	400	0x600231	550	0x580221	693	0x6f0221
142	0x440261	294	0x2f0221	406	0x410221	556	0x590221	700	0x700221
148	0x470261	300	0x600241	412	0x420221	562	0x5a0221	706	0x710221
154	0x4a0261	306	0x310221	419	0x430221	569	0x5b0221	712	0x720221
160	0x4d0232	312	0x640241	425	0x440221	575	0x5c0221	718	0x730221
166	0x5d0271	319	0x660241	431	0x450221	581	0x5d0221	725	0x740221
172	0x450251	325	0x4e0231	437	0x460221	587	0x5e0221	731	0x750221
178	0x470251	331	0x6a0241	438	0x460221	588	0x5e0221	737	0x760221
184	0x670271	338	0x510231	444	0x470221	594	0x5f0221	743	0x770221
190	0x5b0232	344	0x6e0241	450	0x480221	600	0x600221	750	0x780221
196	0x5e0261	350	0x540231	456	0x490221	606	0x610221	756	0x790221
200	0x400241	353	0x710241	462	0x4a0221	612	0x620221	762	0x7a0221
206	0x420241	356	0x720241	469	0x4b0221	619	0x630221	768	0x7b0221
212	0x440241	359	0x730241	475	0x4c0221	625	0x640221	775	0x7c0221
217	0x570251	362	0x570231	481	0x4d0221	631	0x650221	781	0x7d0221
223	0x6b0232	366	0x750241	487	0x4e0221	637	0x660221	787	0x7e0221
229	0x6e0261	369	0x760241	494	0x4f0221	638	0x660221	793	0x7f0221
235	0x710232	375	0x5a0231	500	0x500221	644	0x670221	800	0x800221
242	0x610251	378	0x790241	506	0x510221	650	0x680221	825	0x420211

The `fildiv1` / `fildiv2` fields of the legacy PLL struct are AntRouter values the A3 does not use. Ignore them; transmit only the `vilpll` word.

10. Work frame & header layout SILICON ✓ (nonce return proves acceptance)

Work frame — host → chain

Offsets after the 55 AA prefix. 84 bytes after the prefix, 86 on the wire.

Offset	Width	Field	Value
0	1	opcode	0x20
1	1	work_id	work_id & 0x7F. Returned in the nonce frame as wc (§11); it is the only work identifier the chip returns, so keep every in-flight work_id's header until its nonces are no longer expected.
2 ... 81	80	header	The 80-byte logical header with every 32-bit word byte-reversed (wireswap, below).
82	1	CRC16 high byte	CRC-16/CCITT-FALSE (§6) over bytes 0–81 inclusive (82 bytes: opcode, work_id, 80 swapped header bytes). Big-endian.
83	1	CRC16 low byte	

Logical header — 80 bytes, before the per-word byte swap

Offset	Width	Field	Content
0 ... 31	32	prevhash	From the job (§12).
32 ... 39	8	nonce	[32:36]: 32-bit value the chip enumerates; send as 00 00 00 00. [36:40]: host half; send as 00 00 00 00 — the silicon did not reliably accept a nonzero host half in the tested profile (OPEN).
40 ... 47	8	ntime	The job's ntime, unchanged per work.
48 ... 79	32	merkle root	Fresh per work (§12).

```
wireswap(h): for each 4-byte word w = h[4k:4k+4], k = 0..19: emit w[3] w[2] w[1] w[0]
frame = 0x20 | (work_id & 0x7F) | wireswap(header80) | CRC16_hi | CRC16_lo
```

Work generation practice SILICON ✓ (as stock cgminer does it)

- nonce[32:40] = 0 in every work sent.
- ntime = the job's ntime, not rolled.
- The merkle root is rolled every work: a fresh extranonce2 → fresh coinbase → fresh root (§12). This is how the search space is extended.
- Observed work rate on a 60-chip chain: ~44–55 works/s; each work is a full 2^{32} sweep in ~18 ms at stock clock (§13).

Complete example frame (all-zero header, work_id 1 — the open-core dummy): §8, pass 1.

11. Nonce return frame & verified hash convention

Return frame — chain → host SILICON ✓

9 bytes. Offsets from the start of the frame.

Offset	Width	Field	Value
0 ... 1	2	prefix	AA 55
2 ... 5	4	nonce	32-bit, big-endian. nonce = int.from_bytes(frame[2:6], 'big').
6	1	diff	Byte as named in the stock driver. Semantics not further specified in the source.
7	1	wc	work_id & 0x7F of the work that produced the nonce.
8	1	flags crc5	Bit 7 = 1: non-data (register) response; bit 7 = 0: nonce. Bits 4:0 = CRC5 (§6) over the first 51 bits of the 7-byte body (bytes 2–8). Bits 6:5: not assigned in the source.

CRC5 span: body = bytes 2–8 = 56 bits; the first 51 bits are bytes 2–7 in full plus bits 7:5 of byte 8. The flag bit is inside the span (synthetic KATs in §6). A frame whose CRC5 does not match is discarded. The field layout of a register response (bit 7 set) beyond the flag and CRC is not captured in the source; the enumeration check in §7 step 5 relied on each chip answering READ_REG(0x00) with a valid-CRC5 frame carrying its assigned address.

Hash convention SILICON ✓ — 17,702 / 17,702 returned nonces reproduced

The chip computes Blake2b-256 over exactly the 80 bytes that were on the wire, i.e. the per-32-bit-word byte-swapped header. Blake2b-256 = BLAKE2b with the 32-byte digest parameter, unkeyed, no salt or personalization (Python hashlib.blake2b(data, digest_size=32)).

```
wireswap(h)      = concat over 4-byte words of that word reversed          # same function
header80        = prevhash(32) | nonce32(4) | 00 00 00 00 | ntime(8) | merkle-root(32)
                  # nonce32 = return-frame bytes [2:6] placed verbatim at header[32:36]
                  # (identical to the big-endian encoding of the parsed integer)
solution_int     = int_from_big_endian( blake2b_256( wireswap(header80) ) )
leading_zero_bits = 256 - solution_int.bit_length()
```

```
import hashlib

def wireswap(h: bytes) -> bytes:
    return b''.join(h[i:i+4][::-1] for i in range(0, 80, 4))

def blake2b_256(b: bytes) -> bytes:
    return hashlib.blake2b(b, digest_size=32).digest()

def depth(prevhash: bytes, ntime8: bytes, merkle-root: bytes, nonce32: int) -> int:
    header = prevhash + nonce32.to_bytes(4, 'big') + bytes(4) + ntime8 + merkle-root # 80 bytes
    sol = int.from_bytes(blake2b_256(wireswap(header)), 'big')
    return 256 - sol.bit_length() # leading zero bits of the solution
```

Item	Value	Tag
Nonces reproduced by this verifier	17,702 of 17,702	SILICON ✓
Best observed depth	46 leading zero bits	SILICON ✓
Library self-check: blake2b_256(80 × 0x00)	143aa0da2b6a4ca39eee3ee50a6536d75eedff3b5ef0229a6d603afa7854d5b8	DERIVED algorithm check only; not a silicon value

12. Stratum, extranonce2, and merkle-root construction (pool: convoy) **SILICON** ✓ live jobs mined

Item	Value
Pool	mine.convoy.xyz:3333
Chain	Convoy: Bitcoin-Knots Blake2b-PoW fork. The pool computes the outer header stages server-side and hands the ASIC a Sia-style 80-byte job over ordinary Bitcoin-stratum framing.
mining.subscribe	Returns extranonce1 and extranonce2_size. On convoy extranonce2_size = 8.
mining.authorize	Standard.
mining.notify	Supplies job_id, prevhash, coinb1, coinb2, merkle branch list (hex entries), ntime.
Difficulty / target	target = diff1 / diff, diff1 = 0xFFFF << 208. diff is set by the pool (usually vardiff); it is pool policy, not a device constraint. Convoy example: diff 16383.75 → target ≈ 2 ²¹⁰ .
mining.submit params	[worker, job_id, extranonce2_hex, ntime_hex, nonce_hex]. The byte order of these fields, and any per-word swap applied to the reconstructed header, are pool-specific: validate against the target pool.

extranonce2

- Host-chosen, extranonce2_size bytes (8 on convoy), unique per work.
- Rolled per work: each value yields a different coinbase → different merkle root → different header, without touching the nonce field the chip enumerates.
- Multiple chains: allocate extranonce2 from one shared counter so no two works collide.

Coinbase, merkle root, header

```

coinbase = coinb1 || extranonce1 || extranonce2 || coinb2

root = blake2b_256( 0x00 || coinbase )
for h in mining.notify.merkle_branch:           # each h hex-decoded to bytes
    root = blake2b_256( 0x01 || h || root )

header80 = prevhash(32) || nonce(8) || ntime(8) || root(32)    # nonce(8) = 0 when sent to the chip

```

Per-work host loop

```
work_id = (work_id + 1) & 0x7F
extranonce2 = next_from_shared_counter()           # 8 bytes on convoy
root = merkle_root(job, extranonce1, extranonce2)
header = prevhash || 0x00x8 || ntime || root
send 55 AA | 0x20 | work_id | wireswap(header) | CRC16           # §10
store (work_id → job_id, extranonce2, header) for nonce lookup
on return frame with wc == work_id: verify depth (§11) against target; submit if it meets the pool t
```

13. Measured performance SILICON ✓ bridge-limited

Configuration	Rate	Notes
One chain (60 chips), driven over an SSH-proxied hardware bridge with per-work pacing	~51 works/s ≈ 219 GH/s	Each work is a full 2^{32} sweep in ~18 ms at stock clock.
Three chains in parallel	≈ 660 GH/s	

Both figures are limited by the SSH bridge round-trip, not by the silicon. A native on-board driver removes that ceiling.

14. Host portability + 4-chip specifics

The driver command layer is transport-agnostic (a write/read vtable). Any host with (a) I2C to the PIC and (b) a UART to the chain can run it. Frames are identical on every host.

Host	I2C	UART	Status / requirements
OEM A3 control board (AM335x Linux)	/dev/i2c-0	/dev/tty01, tty02, tty04, tty05	Proven via a small on-board bridge binary driven over SSH. SILICON ✓
Raspberry Pi	native /dev/i2c-1	a UART, or USB-serial	Add SDA/SCL pull-ups and a 3.3 V feed to pin 16.
ESP32 (esp-miner style)	HW or bit-bang I2C	UART	Identical frames.
aditBoard v2 (RP2040 USB)	control-port I2C	data-port UART pass-through	Two blockers, both must be fixed (below).

aditBoard v2 blockers

1. No I2C pull-ups on SDA/SCL. Add 2.2 kΩ–10 kΩ from pin 3 (SDA) and from pin 4 (SCL) to pin 16 (3.3 V).
2. The RP2040 emberone-usbserial-fw leaves the DW_apb_i2c peripheral holding SCL low after any aborted or timed-out transfer. A transfer timeout alone does not clear it. On timeout the firmware must reset/re-initialise the I2C peripheral (or bit-bang I2C) so the pad releases; otherwise a stalled bus latches SCL low until power-cycle.

4-chip board specifics DERIVED — no 4-chip board has been exercised

Item	4-chip value	Basis
chip_count	4	—
SET_ADDRESS interval	$256 / \text{next_pow2}(4) = 64$	Formula SILICON ✓ at 60 chips (interval 4).
Chip addresses	0x00 0x40 0x80 0xC0	Frames in §7 step 5.
Power	One PIC16F1704 (§4), or a directly-enabled DC-DC. With a directly-enabled DC-DC, §4 does not apply and no heartbeat is needed; the chain protocol is unchanged.	—
UART	One chain, one UART.	—
MISC_CONTROL write target (§7 step 8)	Unicast to 0x10 is the captured 60-chip form; no chip sits at 0x10 with interval 64.	OPEN — §15.
Work pacing	Not measured. Estimate below.	DERIVED

Pacing estimate, not a measurement. Per-chip rate at the §13 clock = $219 \text{ GH/s} \div 60 \approx 3.65 \text{ GH/s}$; 4 chips $\approx 14.6 \text{ GH/s}$. The interval formula together with the 60-chip "full 2^{32} sweep in $\sim 18 \text{ ms}$ " figure indicates the 2^{32} space is split across chips by address; with interval 64 each chip would cover 2^{30} nonces per work $\rightarrow \approx 0.29 \text{ s per work} \rightarrow \approx 3.4 \text{ works/s}$. Size work timeouts and the in-flight work_id window from a measurement on the 4-chip board, not from the 60-chip 18 ms figure.

15. Open items

Unresolved on silicon OPEN

Item	State
Registers 0x28 / 0x2c	Written 0x00000000 during bring-up (§7 steps 6–7); exact function unconfirmed.
Host nonce half, header [36:40]	Left zero because the silicon did not reliably accept a nonzero host half in the tested profile. The mechanism to use it, if any, is unconfirmed. Until resolved, the only search-space extension is rolling the merkle root per work (§10, §12).
MISC_CONTROL (0x1c) target on a 4-chip board	Captured as unicast to chip 0x10 on the 60-chip board (§7 step 8). With interval 64 there is no chip at 0x10. Untested candidates: unicast to 0x00 (41 09 00 1C 07 00 7A 61 + CRC5) or broadcast 51 09 00 1C 07 00 7A 61 1F DERIVED . Confirm by checking that nonces return after open-core.

Not captured in the source data (capture needed; not chip unknowns)

Item	What is missing
Temp-sensor init (§7 step 9)	Values written to CORE_MUX_SELECT 0x40 and GENERAL_I2C 0x20, and whether hashing depends on them.
Address mode of §7 steps 2, 6, 7	Source states the mode only for step 3 (broadcast) and step 8 (unicast 0x10). Steps 2, 6, 7 are presented broadcast.
Inter-frame timing on the chain UART	No delays recorded; only frame order.
Register-response frame layout	Only the bit-7 flag and the CRC5 span are known (§11). Field positions of the returned register value / chip address are not recorded.
Nonce-return diff byte	Name only; semantics not recorded.
PIC reply lengths	Recorded for enable (2 bytes) and get_version (5 bytes) only.
Chain 3	/dev/tty05 and PIC 0x53 not exercised.
PLL entries other than 500 MHz	FROM FW only; no other frequency was run on silicon.
4-chip board	Nothing in §14 has been run on a 4-chip board.