N4X Reference Manual

Scannex Electronics Limited

2023-03-10

Scannex Electronics Ltd, UK
t: +44(0)1273 715460
https://www.scannex.co.uk
info@scannex.co.uk

Scannex LLC, USA
t: 1-866-BUFFER (1-866-428-3337)
https://www.scannex.com
info@scannex.com

Date	Author	Release
2023-03-10	MP	Initial document

Table of Contents

N4X Overview

Description

The ip.buffer N4X combines the ip.buffer with a special Power Control Board inside an N4X class enclosure.

The Power Control Board (PCB) contains a small CPU that provides monitoring and timing functions, and can be controlled by the ip.buffer through an I2C/TWI interface.

Part Numbers

Part Number	Description
ip.1-32.4g.pc.N4X	single-port, 32MByte, 4G Cellular
ip.4-256.4g.pc.N4X	four port, 256MByte, 4G Cellular
ip.4-456.4g.pc.2a.N4X	four port, 256MByte, 4G Cellular, dual antenna

Features

The PCB includes these features:

- Controlling the power of the ip.buffer
- Waking the ip.buffer up:
  - WAKEUP inputs (pulse or level)
  - Interval or periodic times
  - Magnet/push-button activation
  - Cold power cycle (for engineer site visits)
- Forcing output power on the COM ports
  - Allows the instrumentation to be kept alive while the ip.buffer is sleeping
- Output switches for COM1 to COM4
  - Regulated 12V (See Electrical Characteristics)
  - or VIN
- Power input 11.5V to 30V
  - Approx 150µA quiescent current when ip.buffer off, and power outputs off.
- Future expansion options for ModBus RS485 and SDI-12

ip.buffer Requirements

To gain the full functionality of the PCB, the ip.buffer must be running firmware v3.03.328 or above.

If the ip.buffer is running 3.02 or prior, then some basic, fixed, functionality is provided:

- ip.buffer kept alive for 60 seconds on any WAKEUP
- ip.buffer woken up every 12 hours
- ip.buffer alive for 15 minutes on a cold power cycle (to allow easy setup of a blank ip.buffer)

Version History

Power Control Board

- v1.00.0002, 2023-02-22
  - Initial released version

ip.buffer

- v3.03.328, 2023-02-14
  - Correct I2C internals
  - Prebuilt Lua APIs for accessing PowerControlBoard
  - Prebuilt LuaDebug terminal handler code

Serial Numbers

N4X Version	ip.4.N4X	ip.1.N4X
PCB (*full processor control*)	≥ `00-02-ae-20-06-fe`	≥ `00-02-ae-10-3a-01`
PCB (*no processor control*)		≥ `00-02-ae-10-3a-d8` ≤ `00-02-ae-10-3a-dc`
Original (*Screw terminal, PTH*)	≤ `00-02-ae-20-06-e3`	≤ `00-02-ae-10-3a-e1`

Manual for the "Original":
https://scannex.com/files/ipbuffer/ip.buffer_N4X_manual_20210413.pdf

Upgrading

The ip.buffer can be upgraded through the normal methods.

The PCB has to be upgraded on-site, with access to the 10-way IDC connector (HD2), using a standard CMSIS-DAP Cortex programming adapter.

This requires access to the inside of the N4X enclosure, and cannot be done remotely.

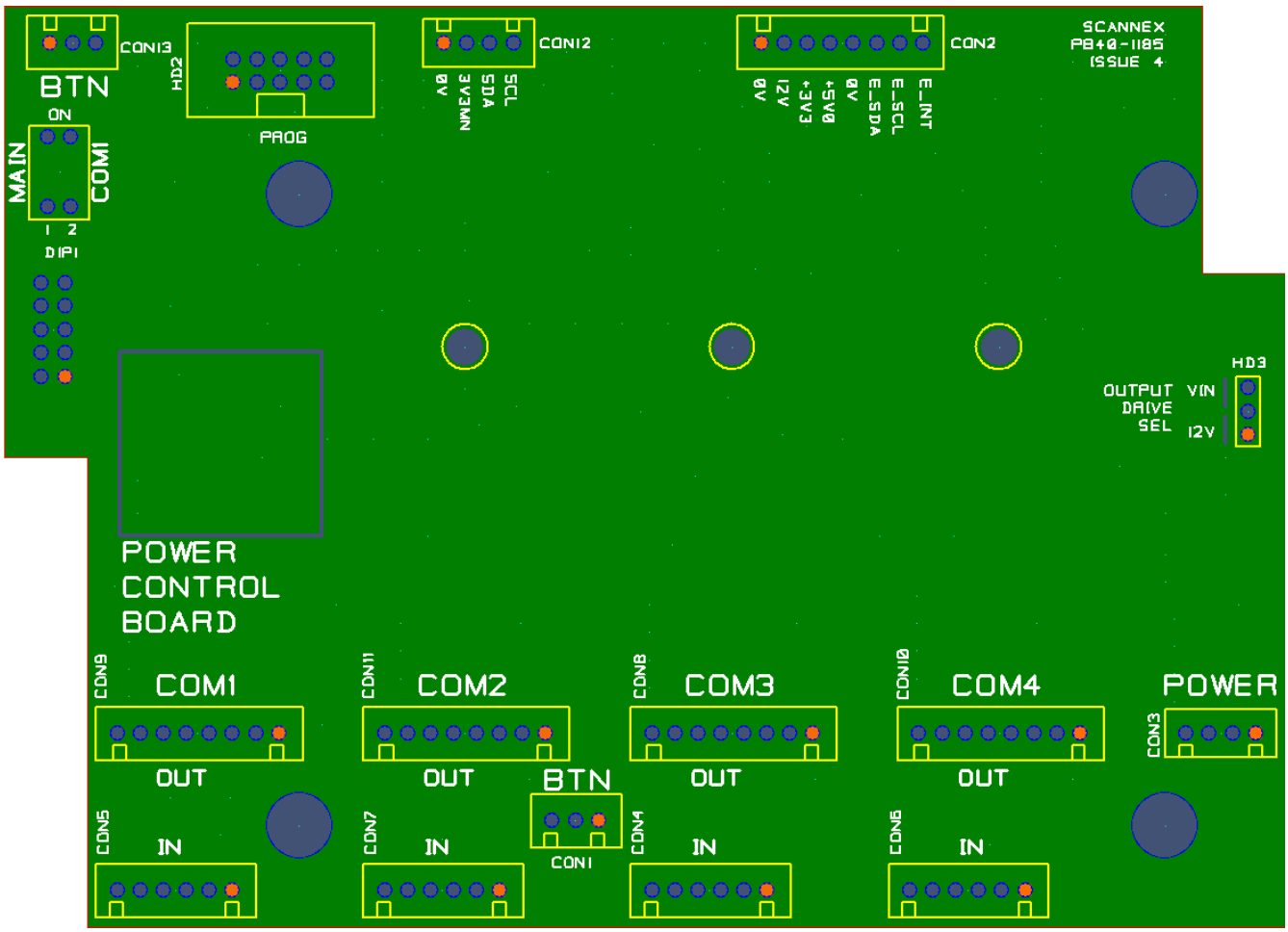
A Raspberry Pi Pico RP2040 running Free-DAP can be used with the open source EDBG project on a laptop.

In time, Scannex will be able to provide a small, self-contained, programming adapter to upgrade the PCB.

Connections & Cables

Physical connections and electrial information.

PCB Layout



Pin 1 on each connector highlighted in red/orange.

Configuration

DIP Switch

The DIP1 switch (top left) provides power overrides.

DIP#	Description
1	Forces ip.buffer power when "ON" (default "OFF")
2	Forces COM1 POWER OUT when "ON" (default "OFF")

Output Drive Select

HD3 (middle right) controls the power output voltage to the COM1 to COM4 power outputs.

Option	Pins	Description
VIN	2+3	Direct from POWER input
12V	1+2	From the 12V regulated supply (default)

Magnet Activation

The ip.buffer can be woken with a strong magnet.

The PCB detects with a reed switch that is placed underneath the LED window. A magnet can be placed above the window, or on the top case side (opposite the connector panel).

The blue LED will flash 2-5 to show the PCB has sensed the magnet.

Connections

COM1-4

- Mating Connector: Souriau UTS6JC10E7P
- Internal Cable
  - Panel Connector: Souriau UTS710E7S
  - 8-way JST XHP-8 (CON9, CON11, CON8, CON10)

Souriau	JST Pin	Description
A	1	POWER OUT#
B	2	GND
C	3	COM#.TX
D	4	COM#.RX
E	5	GND
-	6	COM#.DTR
G	7	WAKEUP#
-	8	(COM1) OUT_CNT digital
-	8	(COM2-4) DSR#

POWER

- Mating Connector: Souriau UTS6JC10E7S
- Internal Cable
  - Panel Connector: Souriau UTS710E7P
  - 4-way JST XHP-4 (CON3)

Souriau	JST Pin	Description
A	1	VIN
B	2	GND
-	3	GND
G	4	WAKEUP PWR

VIN is fused and reverse polarity protected.
Case is tied to GND.
See Input Supply for specifications.

Ethernet

- Mating cable:
  - Metz Connect 142M4D15020
  - Farnell 2442817
- Panel Connector: M12 D-Code

M12 D-Code	RJ45
1	1
2	2
3	3
4	6

Antenna

- Panel Connector: TNC

Internal Connections

ip.buffer COM Ports

D9F ←→ 6-way JST XHP-6 (CON5, CON7, CON4, CON6)

D9F	JST Pin	Description
4	1	DSR
8	2	RTS
3	3	RXD
7	4	CTS
2	5	TXD
6	6	DTR

BTN

3-way JST XHP-3 (CON13 + CON1)

JST Pin	Description
1	LED (blue)
2	GND
3	MAGNET (pull to GND for active)

The Blue LED is controlled by the PCB.
The Amber LED is the ip.buffer's "M" (MODEM) LED.
The Green LED is the ip.buffer's "S" (STATUS) LED.

Expansion and Debug

QUICC/I2C

4-way JST XHP-4 (CON12)

JST Pin	Description
1	GND
2	3V3
3	SDA

JST Pin	Description
4	SCL

For adding I2C/TWI devices to the ip.buffer bus.

Manufacturing Connections

PROG

10-way IDC (HD2)

Standard CMSIS-DAP / SWD connection for programming the Atmel ATSAML10 CPU.

TEST

8-way SIL 0.1" (CON2)

JST Pin	Description
1	GND
2	12V
3	3V3
4	5V
5	GND
6	E.SDA (expansion)
7	E.SCL (expansion)
8	E.INT (expansion)

Electrical Characteristics

Input Supply

- 11.5-30V input VIN
- Protection:
  - 1.8A slow blow PTC thermal fuse
  - Reverse Battery Protected
  - Power output switches turn off when VIN < 10.5V

Power Outputs

- COM POWER individual ratings:
  - 0.7A minimum
  - 1.5A typical
  - 700mJ max inductive load switch-off energy
  - COM#.CTS is asserted when output voltage > 3V approx.
- 12V combined load between all outputs (HD3=12V)
  - 1.7A @ VIN=12V
  - 2.5A @ VIN=24V
  - 3A transient peak for a few seconds
- VIN combined load between all outputs (HD3=VIN)
  - 1.7A
  - > 3A for a few seconds

Wakeup Inputs

- 3-18V
- Input impedance
  - 40k, ≤ 3.3V
  - 10k, > 3.3V

Lua API

The Lua API for the ip.buffer.

pcb : Power Control Board API

```
v = pcb.led()
v = pcb.led(seq)
v = lcd.led(seq, enable)
```

Control LED Sequence Bits

Parameters

- seq: The sequence number, e.g. 0x42
- enable: Boolean whether to enable the flag

Returns

- v: The bit field of the 25-bits

Description Controls individual LED flash sequences. The PCB will show the highest flash sequence that's enabled.

```
t = pcb.alive()
```

Read Time Alive

Parameters None

Returns

- t.power: Time, in seconds, the N4X has had power
- t.main: Time, in seconds, the ip.buffer has been powered

Description Reads the timer registers.

If the PCB is rebooted, these counters will be reset.

pcb.debug()

Enable Debug Timings

Parameters None

Returns None

Description Some timing parameters have enforced minimum values. This API method allows for shorter times that are more useful for testing.

t = pcb.info()

Query `PCB` Settings

Parameters None

Returns

- `t.power`: The SecondsPowered register
- `t.main`: The SecondsMain register
- `t.counter`: The WakeupCounter register
- `t.interval`: The WakeupInterval register
- `t.wake.bits`: The WakeupFlags register
- `t.wake.text`: The text names of why woken
- `t.live.bits`: The WakeupInputs register
- `t.live.text`: The text name of the currently active inputs
- `t.leds`: The LedBits field register
- `t.mv`: The millivolt reading at VIN
- `t.onfor`: The PowerTimer down counter register
- `t.duration`: The <<WakeupTime` register
- `t.outs`: The PowerControl register
- `t.reset.bits`: The ControllerStatus register
- `t.reset.text`: The decoded string values
- `t.temp`: The temperature in steps of 0.125°C.

Description Reads the I2C memory block and decodes.

Debug function.

`pcb.led_clear()`

Clear LED Patterns

Parameters None

Returns None

Description Clears all LED pattern bits.

`pcb.led_set(bits)`

Set LED Patterns

Parameters
- `bits`: The LED sequence bits

Returns None

Description Sets the LED sequence bits.

There should be no need to use this function.

This register allows emulation of the rt.buffer LED patterns. However, since the window on the N4X shows green/amber/blue, it is confusing to flash the PCB's blue LED while the ip.buffer is alive. Consequently, by default, the PCB will only flash the blue LED while the ip.buffer is asleep (or when the magnet/user is triggered).

Triggering the magnet will produce a 2-5 flash pattern on the blue LED.

`mv = pcb.mv()`

Get Input Voltage

Parameters None

Returns
- `mv`: Millivolt reading from the POWER input

Description Reads the current voltage.

```
pcb.off()
pcb.off(int)
Power Off the ip.buffer
```

Parameters • `int`: True means interval. False means periodic (i.e. do not reset the timer)

Returns None

Description Turns off the ip.buffer, and optionally modifies the wakeup counter.

When calling `pcb.off()` the wakeup counter is not reset. This enables a regular, periodic, wakeup.

When calling `pcb.off(true)` the wakeup counter is reset to zero, so the ip.buffer will wakeup again after the wakeup timer - making the delay an interval.

If COM1.DTR is asserted, or if any WAKEUP hardware input is asserted, the ip.buffer will not turn off (because active hardware controls override the firmware controls).

this API call will reset the Reasons flags. Use `pcb.onfor(0)` to release the PCB control without clearing the wakeup reason flags.

```
ok = pcb.ok()
Check if PCB present
```

Parameters None

Returns • `ok`: True if the `PCB` is present on the I2C bus

Description Queries the I2C/TWI bus for the PCB address 42.

```
seconds = pcb.onfor(sec)
seconds = pcb.onfor()
```
Keep ip.buffer Alive

Parameters
- `sec`: The number of seconds to keep the ip.buffer alive.

Returns
- `seconds`: The number of seconds remaining.

Description Reads, or updates the number of seconds the `PCB` should keep the ip.buffer alive.

This can be repeatedly called to keep 'topping up' the timer, so the ip.buffer stays alive.

If COM1.DTR is asserted, or if any WAKEUP hardware input is asserted, the ip.buffer will not turn off even when this timer reaches zero (because active hardware controls override the firmware controls).

```
pcb.power_clear()
```
Turns all power off

Parameters None

Returns None

Description Sets all COM port outputs (and OUT_CNT) to disabled.

```
v, t = pcb.power_out()
v, t = pcb.power_out(chnl, enable)
v, t = pcb.power_on(chnl)
v, t = pcb.power_off(chnl)
```
Control Power Outputs

Parameters
- `chnl`: Channel number 1-4, 8 (for OUT_CNT)
- `enable`: Boolean flag

Returns
- `v`: Integer bit field
- `t`: Table representation of the enabled bits.

Description Reads, controls, turns on/off the power output for each COM port.

The value for the table `t` contains the bits that are enabled.

```
function FindInTable(t, lookfor)
  local i,v
  for i,v in pairs(t)
```

```
  do
    if v == lookfor
    then return i
    end
  end
  return nil
end

local v,t = pcb.power_out()

if FindInTable(t, 4)
then
  -- COM4 POWER is enabled
end
```

```
r1,r2,r3,r4 = pcb.random()
```

Get Random Bytes

Parameters None

Returns
- `r1,r2,r3,r4`: Four 32-bit integer values

Description Uses the TRNG (True Random Number Generator) in the SAML10 CPU.

```
t = pcb.reasons()
t = pcb.reasons(clr)
```

Get Reasons

Parameters
- `clr`: Which flags to clear (either number or comma separated string)

Returns
- `t.wake.bits`: The bit field of why woken
- `t.wake.txt`: The text names of why woken
- `t.live.bits`: The bit field of the currently active inputs
- `t.live.txt`: The text name of the currently active inputs

Description Reads the reason the ip.buffer was powered, and why it is alive now.

Bit field names:

Bit	Text	Description
0	WAKEUP1	COM1 WAKEUP pin triggered
1	WAKEUP2	COM2 WAKEUP pin triggered
2	WAKEUP3	COM3 WAKEUP pin triggered

Bit	Text	Description
3	WAKEUP4	COM4 WAKEUP pin triggered
4	WAKEUPPWR	POWER WAKEUP pin triggered
6	DTR1	The ip.buffer COM1.DTR is asserted
7	DIP	The DIP1 (MAIN) switch is set to 'On'
8	USER	The magnet sensor has been triggered
13	INTERVAL	The time interval/period has expired
14	PCB	The PCB is keeping the ip.buffer running
15	COLD	The whole system has had a cold boot / power cycle

```
t = pcb.reset_cause()
```

Query SAML CPU Reset Cause

Parameters None

Returns

- `t.bits`: The bit field values
- `t.text`: String representation of reasons

Description Queries the SAML10 reset cause register.

Bit-field names:

Bit	Text	Description
0	COLD	The system has had a cold boot power cycle
1	BOD12	Brown out 1.2V
2	BOD33	Brown out 3.3V
4	RST	External Reset on the SAML10
5	WDT	Watchdog Timer inside the SAML10
6	SWD	Hardware debug probe requested reset

```
degc = pcb.temp()
```
Read Temperature

Parameters None

Returns
- `degc`: Temperature in °C, with resolution of 0.125°C

Description The `PCB` has a temperature sensor that allows the ip.1 to read temperature, and gives the ip.4 better resolution.

```
t = pcb.version()
```
Get Version Strings

Parameters None

Returns
- `t.vers`: Firmware version
- `t.date`: Firmware date
- `t.part`: Part number
- `t.name`: Product name

Description Reads the firmware version information from the `PCB`.

```
t = pcb.wakeup()
t = pcb.wakeup(wt)
```
Read or Set Wakeup Timers

Parameters
- `wt.interval`: The interval, in seconds, to wakeup the ip.buffer
- `wt.duration`: The time, in seconds, to keep the ip.buffer alive after a WAKEUP pulse input

Returns
- `t.counter`: Incrementing seconds counter
- `t.interval`: Interval, in seconds, to wakeup the ip.buffer when counter >= interval
- `t.duration`: Time, in seconds, to keep ip.buffer alive after a WAKEUP pulse input
- `t.onfor`: The number of seconds the PCB will keep the ip.buffer alive

Description Read, or set-and-read, the various timer values.

The `wt` fields are optional. e.g. you can set just the `duration` without updating the `interval`.

pcb.user : Access User Data

The User Data Register in the PCB CPU is RAM-based. All the while the N4X has power, the data will persist.

If the PCB power cycles or reboots, the contents will be erased.

```
data = pcb.user.read()
data = pcb.user.read(ofs, len)
```

Read User Data

Parameters
- ofs: The offset, in bytes, to the user internal register.
- len: The length, in bytes. If not present, the data is assumed to be ASCIIZ

Returns
- data: The returned binary string

Description Reads either the whole user data area, or a subsection.

```
pcb.user.write(data, ofs)
```

Write User Data

Parameters
- data: The (binary) string data to write to the user data area.
- ofs: The offset, in bytes, to the user internal register.

Returns None

Description Writes data to the user data register.

pcb : Helper API Methods

```
txt = pcb._pk(pt,v)
```
Pack a Value

Parameters
- `pt`: Packet format
- `v`: Value

Returns
- `txt`: Return string

Description Wrapper abbreviation for `struct.pack(pt,v)`

```
txt = pcb.bits2string(tbl, bits)
```
Convert Bit-Field to String

Parameters
- `tbl`: Table of bit-field names
- `bits`: Integer value

Returns
- `txt`: Comma separated string of bit names

Description Uses a table to convert a bit field into a string representation.

```
pcb._reasons = { [0]='WAKEUP1', [1]='WAKEUP2'... [15]='COLD'}
txt = pcb.bits2string(pcb._reasons, 123)
```

```
pcb.cmd(c)
```
Send Command

Parameters • `c`: Command byte

Returns None

Description Writes the command byte into register 0x20.

```
d = pcb.rd(addr, ···)
```
Read I2C Register

Parameters • `addr`: Register address

Returns • `d`: Data value

Description Wraps up the call to `twi.rd(pcb.t(addr), ···)`

```
d = pcb.rd1(addr)
d = pcb.rd2(addr)
d = pcb.rd4(addr)
```
Read Unsigned Integer Register

Parameters • `addr`: Register address

Returns • `d`: Data

Description Reads single byte (uint8_t), double byte (uint16_t), or quad-byte (uint32_t)

```
txt = pcb.rdvar()
```
Read 16-byte Variable Return Register

Parameters None

Returns
- `txt`: String data

Description Reads the 16-byte variable return register, after the command has been issued.

```
txt = pcb.reasons2string(v)
```
Convert Reason Bits to String

Parameters
- `v`: Integer value

Returns
- `txt`: String representation

Description Abbreviation for `pcb.bit2string(pcb._reasons, bits)`

```
txt = pcb.rndb()
```
Read Random Bytes

Parameters None

Returns
- `txt`: Binary string (16 byte)

Description Calls the command to request random data, and pulls back with `pcb.rdvar()` using locking.

```
v = pcb.string2bits(tbl, txt)
```
Convert String to Bit-Field

Parameters
- `tbl`: Table of bit-field names
- `txt`: String to convert

Returns
- `v`: Integer value

Description Converts from a string to a value, using a table of names.

```
v = pcb.string2reasons(txt)
```
Convert Reason String to Bit-Field

Parameters
- `txt`: String to convert

Returns
- `v`: Integer value

Description Abbreviation for `pcb.string2bits(pcb._reasons, txt)`

```
t = pcb.t(addr)
```
TWI Table

Parameters
- `addr`: Register address

Returns
- `t`: Table suitable for the `twi` library calls

Description Generates a temporary table for use with the `twi` library read/write calls.

`pcb.dev=42` is used as the I2C device address

```
txt = pcb.var(n)
```
Request and Fetch Variable

Parameters
- `n`: Variable number

Returns
- `txt`: String data

Description Wraps up a call to request `pcb.cmd(n)` and read `pcb.rdvar()`. Locks the Lua state while issuing the two commands to avoid contentions.

```
pcb.wr(addr, ···)
```
Write I2C Register

Parameters
- `addr`: Register address

Returns None

Description Wraps up the call to `twi.wr(pcb.t(addr), ···)`

```
pcb.wr1(addr, v)
pcb.wr2(addr, v)
pcb.wr4(addr, v)
```
Write Unsigned Integer Register

Parameters
- `addr`: Register address
- `v`: Value (uint8_t, uint16_t, or uint32_t)

Returns None

Description Writes one-byte (uint8_t), two-byte (uint16_t), or four-byte (uint32_t).

Internal Registers

The internal I2C registers of the PCB.

Register Map

Address	Size	Description
0x00	4	SecondsPowered
0x04	4	SecondsMain
0x08	4	WakeupCounter
0x0c	4	WakeupInterval
0x10	2	WakeupFlags
0x12	2	WakeupInputs
0x14	4	LedBits
0x18	2	InputVoltage
0x1a	2	PowerTimer
0x1c	2	WakeupTime
0x1e	1	PowerControl
0x1f	1	ControllerStatus
0x20	1	Command
0x21	15	Spare (for future expansion)
0x30	16	Variable
0x40	160	Data

SecondsPowered

Address	Size	Description
0x00	4	SecondsPowered

The number of seconds that power has been applied to the PCB.

See also SecondsMain

SecondsMain

Address	Size	Description
0x04	4	`SecondsMain`

The number of seconds the ip.buffer has been powered.

See also SecondsPowered

WakeupCounter

Address	Size	Description
0x08	4	`WakeupCounter`

The incrementing counter that's used to determine when to wake up the ip.buffer.

When this counter reaches WakeupInterval, the ip.buffer will be powered up, and have the flag `INTERVAL` set in WakeupFlags.

See also: WakeupInterval

WakeupInterval

Address	Size	Description
0x0c	4	`WakeupInterval`

The time, in seconds, for when the ip.buffer should be woken.

See also: WakeupCounter

WakeupFlags

Address	Size	Description
0x10	2	`WakeupFlags`

A bit-field showing the reasons the ip.buffer was woken.

To clear one of the flags, a `1` value should be written. e.g. to clear the MAGNET and WAKEUP1 flags, write the value `1 + 256 = 257`

WakeupFlag Bits

The flags are the same for WakeupFlags and WakeupInputs.

Bit	Decimal	Description
0	1	WAKEUP1
1	2	WAKEUP2
2	4	WAKEUP3
3	8	WAKEUP4
4	16	WAKEUP POWER
6	64	DTR1
7	128	DIP-MAIN
8	256	MAGNET
13	8192	INTERVAL (WakeupCounter & WakeupInterval)
14	16384	PowerControlBoard
15	32768	Cold power cycle

WakeupInputs

Address	Size	Description
0x12	2	`WakeupInputs`

The current, live, state of the wakeup values.

See WakeupFlag Bits for bit-field description.

LedBits

Address	Size	Description
0x14	4	`LedBits`

The LED pattern bit field for the blue LED.

The following are special patterns: - 11 - idle, when the ip.buffer is off. Happens every 8 seconds - 12 - auto-clearing 'data collecting' pattern. - 25 - auto-clearing 'magnet/button has been triggered' pattern.

Writing

You can update the bit-field by writing values between `0x0000.0000` and `0x01ff.ffff` (all 25 bits set)

There are some shortcuts by using the unused higher bits: - `0x6000.00xx` sets the flashing parameter `xx` (in HEX) — e.g. `0x6000.0012` sets the data collection flash sequence (which auto-resets) — e.g. `0x6000.0031` enables bit 10 for the 31 flash sequence - `0x4000.00xx` clears the flashing parameter `xx` (in HEX) — e.g. `0x4000.0031` clears bit 10 for the 31 flash sequence — e.g. `0x4000.003f` clears 31, 32, 33, 34, 35 (bits 10 to 14)

See the bit fields below...

Reading

The 32-bit field represent the 11, 12, 13, 14, 15, 21, 22, etc flash patterns. The highest bit number that is set will denote which blue LED flash pattern is shown.

Bit	Flash pattern
0	11
1	12
2	13
3	14
4	15
5	21
6	22
7	23
8	24
9	25

Bit	Flash pattern
10	31
11	32
12	33
13	34
14	35
15	41
16	42
17	43
18	44
19	45
20	51
21	52
22	53
23	54
24	55

InputVoltage

Address	Size	Description
0x18	2	`InputVoltage`

The voltage, in millivolts.

PowerTimer

Address	Size	Description
0x1a	2	`PowerTimer`

The number of seconds the PCB will keep the ip.buffer running.

When this value decrements to zero, the PCB will allow the ip.buffer to sleep, assuming the ip.buffer's COM1.DTR is unasserted (as reported in WakeupInputs & WakeupFlag Bits.

WakeupTime

Address	Size	Description
0x1c	2	`WakeupTime`

The number of seconds the ip.buffer should be kept alive when woken up.

Defaults to 60.

PowerControl

Address	Size	Description
0x1e	1	`PowerControl`

A bit field that indicates which outputs should be held active while the ip.buffer is sleeping.

If the corresponding bit is 1 then the RTS output of the COM port will not be able to control the power output. i.e. The PowerControl bit fields is OR'd with the RTS lines.

Bit field

Bit	Decimal	Output
0	1	COM1 PWR
1	2	COM2 PWR
2	4	COM3 PWR
3	8	COM4 PWR
7	128	OUT_CNT digital output

ControllerStatus

Address	Size	Description
0x1f	1	`ControllerStatus`

A bit-field that shows the reasons why the PCB was reset.

Controller Status Bits

Bit	Decimal	Reason
0	1	Power On Reset
1	2	BOD12
2	4	BOD33
3	8	x
4	16	External Reset
5	32	Watchdog Timer
6	64	System Reset Request via SWD
7	128	x

See ATSAML10 datasheet, section 21.8.1 "Reset Cause"

Command

Address	Size	Description
0x20	1	`Command`

Write-only register to invoke specific commands.

Command List

- Each of the following place the data in `Variable`
  - 0x00 - fetch part number
  - 0x01 - fetch application name
  - 0x02 - fetch version string
  - 0x03 - fetch date string
  - 0x0f - fetch random bytes (using the True Random Number Generator in the SAML)
- Requests to stay running
  - 0x10 - power on forever, until an off command
  - 0x11 - keep power on for 1 minute
  - ...
  - 0x1f - keep power on for 15 minutes
- Debug

    - 0xee - enable extended values for diagnostics
- Power off commands
    - 0xfa - crash (reboots SAML via WDT) test command
    - 0xfc - power off and reset the `WakeupCounter`, making the wakeup an interval
    - 0xfd - power off now, making the `WakeupCounter` wakeup a periodic time.

Variable

Address	Size	Description
0x30	16	`Variable`

The register that returns values requested by `Command`.

Data

Address	Size	Description
0x40	160	`Data`

A block of RAM that can be used by the ip.buffer when it sleeps.

The values will be cleared when the PCB reboots or is power cycled. This RAM area is only useful for between wakeups of the ip.buffer.

Approvals

See the ip.buffer manual for EMC, safety, environmental, and cellular approvals.

EU/UK Declaration of Conformity

See online: https://www.scannex.co.uk/products/ipbuffer/downloads.html

European Union Waste Electrical and Electronic Equipment (WEEE) Statement.

UK Users

In the UK Scannex is registered as a WEEE producer and has responsibility for the recycling of Scannex products and any products returned to Scannex, postage paid, will be recycled at Scannex's cost.

European Users (outside the UK)

Where the supplier of Scannex products is resident in your country then the supplier acts as the importer of the equipment. Thus the supplier has the legal responsibility to deal with recycling:

If the supplier of Scannex products is not resident in your country then the business end-user acts as the importer of the product. It is Scannex understanding that in this situation:

- No organisation is required to register as the WEEE producer
- No organisation is required to provide WEEE collection and recycling arrangements.

Manufacturer/Responsible Party

Scannex Electronics Ltd
Unit 8 English Business Park
English Close
Hove, East Sussex
BN3 7ET
UK
Tel: +44 (0)1273 715460
https://www.scannex.co.uk

Scannex LLC
7400 Beaufont Springs Drive
Suite 300
Richmond, VA 23225
USA
Tel: +1-866-428-3337
https://www.scannex.com

Index

Q

R

S