Test Plan Reference

PLT

0.7.1

2020-05-09

Revision History

Revision History

Revision 0.7.1   2020-05-09

- Expand `extractKey:` documentation
- `program`: Add PSoC4 target
- Document `webhook` command

Revision 0.7.0   2020-04-28

- Document `calibration:` sections
- `program`, `erase`: Add PSoC6 and CMSIS targets
- `measure`: Document unit suffixes, clarify reference argument

Revision 0.6.9   2020-03-30

- Remove `ESP32_HomeKit` target
- `program`: Allow any number of firmware elements
- `program`: Add PN7360, PN7362, PN7460 and PN7462 targets
- `serial set`: Set serial number
- `pin`: Can control `SWD_NRST` and `UARTx_RTS`

Revision 0.6.8   2020-02-24

New commands:

- `canCfg` - Configure CAN interface
- `canClear` - Configure CAN interface
- `canMatch` - Match received CAN message
- `canSend` - Send CAN message
- `ftdiCfg` - Configure external FTDI

Revision 0.6.5   2019-10-07

- Add STM32F2 and ST32F2_STLink targets

Revision 0.6.4   2019-10-08
No change
Revision 0.6.3   2019-10-07

- document `noflush` option for uart and uartExpect commands

- eval command

- uart: extract multiple keys

- ble test: Low-level BLE testing

Revision 0.6.2   2019-10-03

- document JLink/ST-Link target variants

- update initial example

- scan: ANY format

- serial request: user key extraction

Revision 0.5.7   2019-08-22

- `label` command: Document ZPL templates

- add Nordic nRF91 Cortex-M33 targets

Revision 0.5.6   2019-08-19

- Document retry mechanism for test items and test item steps

- Document GATT-level BLE commands

- Document `image` command test step to set background images

- Document setting Label substitutions with the `label` command test step

- add AVRATmega168P/PB targets

Revision 0.5.2   2019-05-13

- `measure`: Document `reference` argument

- Document `nfc` command

Revision 0.5.1   2019-04-16

- Add `CC1352` target

- Document `define` command

Revision 0.5.0   2019-03-31

- Document `freq` command

- Document `measure` command

- Document `mux` command

- Document `pin` command

- Document `power` command

- Document `short` command

- Document `uartCfg` command

Revision 0.4.9   2019-03-28

- Document `serial request` command

- Update UART port names

Revision 0.4.8   2019-03-08
Renamed to Test Plan Reference
Revision 0.4.7   2019-03-08
Split off from System Manual

Table of Contents

Preface

Document describing the test suite definition for use with the Production Line Tool.

1. Test Plan Reference

1.1 Test Plan YAML definition

Example: A minimal test suite, with a single test item, scanning a bar code.

```
title: "PLT demo: Scan"
suite:
- ident: SCAN-T1
  title: Scan MAC address
  steps:
  - command: scan ANY
```

1.1.1 Test Plan Structure

PLT test suites are encoded as YAML text files, starting with the test suite title.

```
title: "PLT demo: Scan"
```

The `title:` is followed by the `suite:` section, containing the test items in the test plan.

1.1.1.1 Test Items

Test items are identified with an `ident:` line, can contain a descriptive `title`, and should contain one or more test item steps.

The `ident:` field can be omitted, in which case test items will be automatically enumerated. A top-level `identPrefix:` field can be used to set common prefix for auto-generated test item identifiers.

During test execution, all test item steps need to complete successfully for the test item to succeed.

1.1.1.2 Test Item Steps

Test item steps consist of `command:` or `uartcmd:` blocks. A Test Item can contain multiple Test Item steps.

1.1.1.3 Retries

Retry counts can be set for at the test item or at the test item step level, by adding a `retry:` field.

```
title: "Retries"
identPrefix: "R"
suite:
- title: Test Item retries
  retry: 3
  steps:
  - command: sleepms 1000
  - command: operator Manual
- title: Test Item retries
  steps:
  - command: sleepms 1000
  - command: operator Manual
    retry: 3
```

1.1.2 Basic Example

A basic In-Circuit Test suite for the PLT demo board.

```
title: "v0.1.8 (Green)"
identPrefix: "ICT"
suite:
- title: Identify DUT
  steps:
  - command: identify nRF52
- title: Erase nRF52
  steps:
  - command: erase nRF52
- title: Program DEMO-BOARD FW nRF52
  steps:
  - command: program nRF52 s132_nrf52_6.0.0_softdevice.hex,ly10-demo-fw-0.1.8.hex
- title: BLE discovery
  steps:
  - command: bledis %BLEMAC% 30  # Wait up to 30 seconds for BLE discovery to complete
```

1.1.3 Calibration

Some test setups may require additional calibration, which can be specified in the YAML test plan with an optional top-level `calibration:` section.

```
calibration:
- [ppc: <ppcID>]
  [impedance: <impedanceOffset>]
  [voltage: <voltageOffset>]
  [voltageDATP07: <voltageOffset>]
  [voltageDATP08: <voltageOffset>]
  [voltageDATP09: <voltageOffset>]
  [voltageDATP10: <voltageOffset>]
  [voltageDATP11: <voltageOffset>]
  [voltageDATP12: <voltageOffset>]
  [voltageMUX0: <voltageOffset>]
  [voltageMUX1: <voltageOffset>]
  [voltageMUX2: <voltageOffset>]
  [voltageMUX3: <voltageOffset>]
  [current: <currentOffset>]
  [current3V3: <currentOffset>]
  [current5V: <currentOffset>]
  [currentVARV: <currentOffset>]
- ...
```

The following fields can be set in a calibration entry:

Field	Description
`ppc:`	Define the PPC for which this calibration entry should be applied. If omitted, the section defines the default offsets, to which a PPC-specific calibration section would be added.
`impedance:`	Specifies the offset for impedance measurements.
`voltage:`	Specifies the offset for voltage measurements.
`voltageDATP07:`	Specifies the offset for voltage measurements on the `voltageDATP07` measurement channel. Overrides the offset specified in the `voltage:` field.
`voltageDATP08:`	Specifies the offset for voltage measurements on the `voltageDATP08` measurement channel. Overrides the offset specified in the `voltage:` field.
`voltageDATP09:`	Specifies the offset for voltage measurements on the `voltageDATP09` measurement channel. Overrides the offset specified in the `voltage:` field.
`voltageDATP10:`	Specifies the offset for voltage measurements on the `voltageDATP10` measurement channel. Overrides the offset specified in the `voltage:` field.
`voltageDATP11:`	Specifies the offset for voltage measurements on the `voltageDATP11` measurement channel. Overrides the offset specified in the `voltage:` field.
`voltageDATP12:`	Specifies the offset for voltage measurements on the `voltageDATP12` measurement channel. Overrides the offset specified in the `voltage:` field.

Field	Description
voltageMUX0:	Specifies the offset for voltage measurements on the `voltageDATP09` measurement channel. Overrides the offset specified in the `voltage:` field.
voltageMUX1:	Specifies the offset for voltage measurements on the `voltageDATP10` measurement channel. Overrides the offset specified in the `voltage:` field.
voltageMUX2:	Specifies the offset for voltage measurements on the `voltageDATP11` measurement channel. Overrides the offset specified in the `voltage:` field.
voltageMUX3:	Specifies the offset for voltage measurements of the `voltageDATP12` measurement channel. Overrides the offset specified in the `voltage:` field.
current:	Specifies the offset for current measurements.
current3V3:	Specifies the offset for current measurements of the `current3V3` measurement channel. Overrides the offset specified in the `current:` field.
current5V:	Specifies the offset for current measurements of the `current5V` measurement channel. Overrides the offset specified in the `current:` field.
currentVARV:	Specifies the offset for current measurements of the `currentVARV` measurement channel. Overrides the offset specified in the `current:` field.

With the following argument values:

Argument	Description
ppcID	PPC identifier for a `ppc:` field, to define a calibration specific to a particular PPC.
impedanceOffset	Offset for impedance measurements. Can be specified with nOhm, uOhm/µOhm, mOhm, Ohm, kOhm/KOhm or MOhm suffixes. If no suffix is specified, the offset is assumed to be specified in Ohm.
voltageOffset	Offset for voltage measurements. Can be specified with nV, uV/µV, mV, V, kV/KV or MV suffixes. If no suffix is specified, the offset is assumed to be specified in Volt.
currentOffset	Offset for current measurements. Can be specified with nA, uA/µA, mA or A suffixes. If no suffix is specified, the offset is assumed to be specified in Ampere.

Example:

```
title: "test plan with calibration"
identPrefix: "ICT-T"
calibration:
# Default offsets
- impedance: -100 # 100 Ohm
  voltage: +20mV # Default for all voltage channels
  voltageDATP01: +40mV
  voltageMUX2: +20mV
  voltageMUX3: +30mV
  currentVARV: -20mA
# Additional offsets specific to a particular PPC
- ppc: 39c8db  # Additional offsets for this PPC
  impedance: +120Ohm
  voltageDATP08: -10mV
  current5V: +100mA
suite:
- title: Impedance Measurement
  steps:
  - command: mux 0 RATP02
  - command: mux 1 RATP17
  - command: mux 2 RA
  - command: mux 3 RVREF
  - command: short 0 2 set
  - command: short 1 3 set
  - command: measure impedance 10-20Ohm
```

1.2 Test Commands

1.2.1 ble gatt - GATT-level BLE Tests

Perform GATT-level BLE interactions with the DUT.

Usage:

```
- command: ble gatt connect addr:<address> [<timeout> [<minRSSI>]]
  [retry: <retryCount>]

- command: ble gatt connect name:<name> [<timeout> [<minRSSI>]]
  [retry: <retryCount>]

- command: ble gatt disconnect
  [retry: <retryCount>]

- command: ble gatt discover
  [retry: <retryCount>]

- command: ble gatt match char:<charUUID> <matchHex>
  [retry: <retryCount>]

- command: ble gatt read char:<charUUID> <matchHex>
  [retry: <retryCount>]

- command: ble gatt sub char:<charUUID>
  [retry: <retryCount>]

- command: ble gatt write char:<charUUID> <valueHex>
  [retry: <retryCount>]
```

Argument	Description
name	GAP name advertised by DUT.
address	BLE MAC address used by the DUT.
timeout	Timeout, in seconds
minRSSI	RSSI treshold (optional)
connect	Connect to GATT peripheral
disconnect	Disconnect from GATT peripheral
discover	Discover GATT services and characteristics
match	Match a value for a subscribed characteristic
read	Read a GATT characteristic
sub	Subscribe to notifications from a GATT characteristic
write	Write to a GATT characteristic
retryCount	Number of times to retry the test step (optional)

Example: Validate BLE Device Information Service

```
title: "DIS Validation"
suite:
- ident: ICT-T1
  title: Identify DUT
  steps:
  - command: identify nRF52
- ident: ICT-T2
  title: Program LY10-DEMO-BOARD FW
  steps:
  - command: program nRF52 s132_nrf52_6.0.0_softdevice.hex,ly10-demo-fw-0.1.7.hex
- ident: ICT-T3
  title: Validate BLE DIS
  steps:
  - command: ble gatt connect addr:%ble_mac% 10
  - command: ble gatt discover
  # GAP:Device Name
  - command: ble gatt read char:2a00 4c5931302d44454d4f5f424f415244 # "LY10-DEMO_BOARD"
  # GAP Appearance
  - command: ble gatt read char:2a01 0000
  # GAP:Peripheral Preferred Connection Parameters
  - command: ble gatt read char:2a04 0600080200009001
  # GAP:Central Address Resolution
  - command: ble gatt read char:2aa6 01
  # DIS:Manufacturer Name String
  - command: ble gatt read char:2a29 424344 # "BCD"
  # DIS:Model Number String
  - command: ble gatt read char:2a24 4c5931302d44454d4f5f424f415244 # "LY10-DEMO_BOARD"
  # DIS:Hardware Revision String
  - command: ble gatt read char:2a27 32303139 # "2019"
  # DIS:Firmware Revision String
  - command: ble gatt read char:2a26 302e312e37 # "0.1.7"
  # DIS:Software Revision String
  - command: ble gatt read char:2a28 312e322e33 # "1.2.3"
  - command: ble gatt disconnect
```

1.2.2 ble test - Low-level BLE Tests

Perform Low-level BLE tests.

Usage:

```
- command: ble test recv <channel>
  [retry: <retryCount>]

- command: ble test xmit <channel> [<length> [<payload>]]
  [retry: <retryCount>]

- command: ble test stop
  [retry: <retryCount>]
```

Argument	Description
channel	BLE channel
length	Test packet payload length
payload	Test packet payload type
retryCount	Number of times to retry the test step (optional)

1.2.3 bledis - Test BLE Discovery

Establishes a BLE connection to the DUT and discovers GATT services.

Usage:

```
- command: bledis %BLEMAC%|<name> [<timeout> [<minRSSI>]]
  [retry: <retryCount>]
```

Argument	Description
name	GAP name advertised by DUT. **%BLEMAC%** to specify the DUT's BLE MAC address instead
timeout	Timeout, in seconds
minRSSI	RSSI treshold (optional)
retryCount	Number of times to retry the test step (optional)

Example: BLE discovery of identified BLE HW MAC address.

```
- ident: ICT-T1
  title: Identify DUT
  steps:
  - command: identify nRF52
- ident: ICT-T2
  title: BLE Discovery
  steps:
  - command: bledis %BLEMAC% 30 -60
```

1.2.4 canCfg - Configure CAN interface

Configure the PLT CAN interface speed.

Usage:

```
- command: canCfg <port> <bitrate>
  [retry: <retryCount>]
```

Argument	Description
port	PLT CAN interface; currently only **CAN0** is available.
bitrate	CAN bit rate, in bits per second.
retryCount	Number of times to retry the test step (optional)

1.2.5 canClear - Configure CAN interface

Clear the buffer of received CAN frames.

Usage:

```
- command: canClear <port>
  [retry: <retryCount>]
```

Argument	Description
port	PLT CAN interface; currently only **CAN0** is available.
retryCount	Number of times to retry the test step (optional)

1.2.6 canMatch - Match received CAN message

Test if a particular CAN message has been received.

Usage:

```
- command: canMatch <port> <msgId> [<b> <b> ...]
  [extractKey: <key>]
  [retry: <retryCount>]
```

Argument	Description
port	PLT CAN interface; currently only **CAN0** is available.
msgId	CAN Message Arbitration ID
b	Payload bytes

Argument	Description
key	Key in which to store the matching CAN message (optional)
retryCount	Number of times to retry the test step (optional)

1.2.7 canSend - Send CAN message

Send a CAN data frame.

Usage:

```
- command: canSend <port> <msgId> [<b <b> ...]
  [extractKey: <key>]
  [retry: <retryCount>]
```

Argument	Description
port	PLT CAN interface; currently only `CAN0` is available.
msgId	CAN Message Arbitration ID
b	Payload bytes
key	Key in which to store the transmitted CAN message (optional)
retryCount	Number of times to retry the test step (optional)

1.2.8 define - Define user key

Manually defines a test plan key, which will be emedded in the test report and serial number requests performed as part of the current test plan.

Usage:

```
- command: define <key> <value>
```

Argument	Description
key	Name of the test plan key.
value	Value for the key; can be any kind of string value.

```
- ident: ICT-T0
  title: Set variables
  steps:
  - command: define work_order 1011X02
- ident: ICT-T1
  title: Invoke webhook with work_order key
  steps:
  - command: webhook request
```

1.2.9 erase - Erase DUT Flash

Erase DUT MCU on-board flash.

Usage:

```
- command: erase <target> [UART0|UART1]
  [retry: <retryCount>]
```

```
- command: erase CMSIS
  pack: <cmsis_pack>
  target: <cmsis_target>
  [retry: <retryCount>]
```

Argument	Description
target	Target to erase

Argument	Description
`UART0`, `UART1`	Port for UART targets
cmsis_pack	For `cmsis` target: CMSIS Pack file
cmsis_target	For `cmsis` target: Target name, as defined in CMSIS Pack file
retryCount	Number of times to retry the test step (optional)

Supported *target* values:

target	Description
`AVRATmega168P_ISP`	Atmel AVR ATmega168P(A) (JTAG:ISP)
`AVRATmega168P_XPm`	Atmel AVR ATmega168P(A) (USB, XPmini)
`AVRATmega168PB_ISP`	Atmel AVR ATmega168PB (JTAG:ISP)
`AVRATmega168PB_XPm`	Atmel AVR ATmega168PB (USB, XPmini)
`CC1352`	TI CC1352 RFSoC
`CMSIS`	CMSIS-DAP programmer, requires `pack:` and `target:` fields (USB)
`DA14580`	Dialog DA14580 RFSoC
`ESP32`	Espressif ESP32 RFSoC (JTAG)
`ESP32_JTAG`	Espressif ESP32 RFSoC (JTAG)
`ESP32_UART`	Espressif ESP32 RFSoC (UART)
`LPC11xx`	NXP LPC11xx MCU (SWD)
`nRF52`	Nordic nRF52 RFSoC (SWD)
`nRF52_JLink`	Nordic nRF52 RFSoC (USB, JLink)
`nRF91`	Nordic nRF9160 RFSoC:Cortex-M33 (SWD)
`nRF91_JLink`	Nordic nRF9160 RFSoC:Cortex-M33 (USB, JLink)
`PSoC4`	Cypress PSoC4 (SWD)
`PSoC6`	Cypress PSoC6 (SWD)
`PSoC6_KP3`	Cypress PSoC6 (USB, KitProg3)
`STM32F2`	ST STM32F2xx MCU (SWD)
`STM32F2_STLink`	ST STM32F2xx MCU (USB, ST-Link)
`STM32F4`	ST STM32F4xx MCU (SWD)
`STM32F4_STLink`	ST STM32F4xx MCU (USB, ST-Link)
`STM32L4`	ST STM32L4xx MCU (SWD)
`STM32L4_STLink`	ST STM32L4xx MCU (USB, ST-Link)

Example: Erase STM32L4 on-board flash.

```
- ident: ICT-T1
  title: Erase
  steps:
  - command: erase STM32L4
```

1.2.10 eval - Evaluate Expression

Evaluate an expression.

```
- command: eval <expression>
  [extractKey: <key>]
```

Argument	Description
expression	Expression to evaluate

Argument	Description
key	Key in which to store the evaluated expression value

Supported operators:

Operator	Description
+	Addition, concatenation
-	Subtraction, Negation
/	Division
*	Multiplication
**	Power
%	Modulo
&	Bitwise And
|	Bitwise Or
^	Bitwise Xor
~	Bitwise Not
>>	Right shift
<<	Left shift
!	Inversion
&&	Logical And
||	Logical Or
?	Ternary True
:	Ternary False
??	Null coalescence
>	Greater than
<=	Less than or equal
>=	Greater than or equal
=~	Regex match
!~	Regex mismatch

Example:

```
title: "Eval"
suite:
- ident: E0
  title: Eval
  steps:
  - command: define test "AC1D"
  - command: eval "test != 'FOOBAR'" # PASS
  - command: eval "test == 'AC1D'" # PASS
  - command: eval "test != 'AC1D'" # FAIL
```

1.2.11 freq - Frequency Monitor Control

Set the channel to use for frequency measurements.

Usage:

```
- command: freq 0|1
  [retry: <retryCount>]
```

Argument	Description
0	Use CLK0

Argument	Description
1	Use CLK1
retryCount	Number of times to retry the test step (optional)

Example: Measure CLK1 frequency

```
title: Measure frequency
suite:
- ident: ICT-T1
  title: Measure CLK1 frequency
  steps:
  - command: freq 1
  - command: sleepms 1000
  - command: measure frequency 7.90-8.10MHz
```

1.2.12 ftdiCfg - Configure external FTDI

Configure an externally connected FTDI FT4232H module.

Usage:

```
- command: ftdiCfg FTDI1 <target> [<UARTxx>][,<UARTxx>[,<UARTxx>[,<UARTxx>]]]
  [retry: <retryCount>]
```

Argument	Description
target	FTDI interface; only FTDI1 can be configured.
UARTxx	UART port to use for each FTDI port: UART2, UART3, .. , or _ to skip.
retryCount	Number of times to retry the test step (optional)

1.2.13 identify - Identify DUT

Identify DUT MCU and/or RF peripherals.

The MCU identifier is stored in the `mcu_id` key, and the BLE MAC address (if available) is stored in the `ble_mac` key.

Usage:

```
- command: identify <target> [UART0|UART1]
  [retry: <retryCount>]
```

Argument	Description
target	Target to identify
UART0, UART1	Port for UART targets
retryCount	Number of times to retry the test step (optional)

Supported *target* values:

target	Description
AVRATmega168P_ISP	Atmel AVR ATmega168P(A) (JTAG:ISP)
AVRATmega168P_XPm	Atmel AVR ATmega168P(A) (USB, XPmini)
AVRATmega168PB_ISP	Atmel AVR ATmega168PB (JTAG:ISP)
AVRATmega168PB_XPm	Atmel AVR ATmega168PB (USB, XPmini)
CC1352	TI CC1352 RFSoC
DA14580	Dialog DA14580 RFSoC
ESP32	Espressif ESP32 RFSoC (JTAG)
ESP32_JTAG	Espressif ESP32 RFSoC (JTAG)

target	Description
ESP32_UART	Espressif ESP32 RFSoC (UART)
nRF52	Nordic nRF52 RFSoC (SWD)
nRF52_JLink	Nordic nRF52 RFSoC (USB, JLink)
nRF91	Nordic nRF9160 RFSoC:Cortex-M33 (SWD)
nRF91_JLink	Nordic nRF9160 RFSoC:Cortex-M33 (USB, JLink)
STM32F2	ST STM32F2xx MCU (SWD)
STM32F2_STLink	ST STM32F2xx MCU (USB, ST-Link)
STM32F4	ST STM32F4xx MCU (SWD)
STM32F4_STLink	ST STM32F4xx MCU (USB, ST-Link)
STM32L4	ST STM32L4xx MCU (SWD)
STM32L4_STLink	ST STM32L4xx MCU (USB, ST-Link)

1.2.14 image - Set background image

Usage:

```
- command: image set <filename>
```

```
- command: image clear
```

Argument	Description
set	Set background image
clear	Clear background image
filename	Filename of the PNG or JPEG element to show

Example: Background for operator test

```
title: "OLED Image"
suite:
- ident: ICT-T1
  title: Show image
  steps:
  - command: image set fighter.png
  - command: operator "Manual test"
```

1.2.15 label - Set Label Substitutions

Usage:

```
- command: label keys <key> [<key>...]
```

The label command defines additional keys to substitute in the ZPL sent to the barcode printer, in addition to the default substitution keys.

Key	Description
DUT_PRODUCT	Product name
DUT_VERSION	Version
DUT_SERIAL	DUT Serial number
MAC_ADDRESS	MAC address
BLEMAC	BLE MAC address
FAILURE_MSG	Test failure

Example: Substitute CODE

```
title: "Custom barcode substitution"
suite:
- ident: ICT-T0
  title: Define label keys
  steps:
  - command: define CODE 12345
  - command: label keys CODE
```

The default ZPL generated by the PLT upon completion of a YAML test plan specification can be overridden by uploading `template-pass.zpl` and `template-fail.zpl` elements as part of the Release deployed through PLTcloud.

Example `template-fail.zpl` element:

```
^FX template-fail.zpl - ZPL Template for failing DUTs
^XA^LH40,30
^MD2
^FO0,10^ADN,30,8^FDCODE^FS
^FO0,80^ADN,30,8^FDN/G^FS
^XZ
```

Example `template-pass.zpl` element:

```
^FX template-pass.zpl - ZPL Template for DUTs that pass testing
^XA
~SD22
^CF0,30
^FO40,20^FDProduct:DUT_PRODUCT^FS
^FO40,30^FDVersion:DUT_VERSION^FS
^FO40,40^FDS/N:DUT_SERIAL^FS
^FO40,50^FDMAC Address:MAC_ADDRESS^FS
^FO40,60^FDBLE Address:BLEMAC^FS
^FO40,70^FDFailure:FAILURE_MSG^FS
^FO40,80^FDCode:CODE^FS
^FO40,170^BY2
^BCN,50,N,N,N
^FDMAC_ADDRESS^FS
^XZ
```

1.2.16 measure - Probe Measurement

Usage:

```
- command: measure <channel> [<signal>] <range> [<reference>]
  [extractKey: <key>]
  [retry: <retryCount>]
```

Argument	Description
channel	Measurement channel
signal	DDTPxx or RDTPxx probe for `pin` measurement
range	Acceptable range
reference	Reference voltage for impedance measurements (default: 5V)
key	Key in which to store the measurement results (optional)
retryCount	Number of times to retry the test step (optional)

The `measure` command supports the following channels:

channel	Description
`current3V3`	3V3 current draw
`current5V`	5V current draw
`currentVARV`	VARV current draw
`frequency`	Frequency (CLK0 or CLK1)

channel	Description
`impedance`	Impedance measurement (RVREF/RA)
`pin`	Digital pin measurement (DDTPxx/RDTPxx)
`voltageDATP07`	DATP07 voltage
`voltageDATP08`	DATP08 voltage
`voltageDATP09`	DATP09 voltage
`voltageDATP10`	DATP10 voltage
`voltageDATP11`	DATP11 voltage
`voltageDATP12`	DATP12 voltage
`voltageMUX0`	MUX0 voltage
`voltageMUX1`	MUX1 voltage
`voltageMUX2`	MUX2 voltage
`voltageMUX3`	MUX3 voltage

The following suffixes are supported for electrical measurement ranges:

Kind	Suffix	Description
Frequency	`Hz`	Hertz (default)
Frequency	`kHz`, `KHZ`	Kilohertz
Frequency	`MHz`	Megahertz
Frequency	`GHz`	Gigahertz
Voltage	`nV`	Nanovolt
Voltage	`uV`, `µV`	Microvolt
Voltage	`mV`	Millivolt
Voltage	`V`	Volt (default)
Voltage	`kV`, `KV`	Kilovolt
Voltage	`MV`	Megavolt
Current	`nA`	Nanoampere
Current	`uA`, `µA`	Microampere
Current	`mA`	Milliampere
Current	`A`	Ampere (default)
Impedance	`nOhm`	Nanoohm
Impedance	`uOhm`, `µOhm`	Microohm
Impedance	`mOhm`	Milliohm
Impedance	`Ohm`	Ohm (default)
Impedance	`kOhm`, `KOhm`	Kiloohm
Impedance	`MOhm`	Megaohm

Example: Electrical measurements

```
title: Electrical measurements
suite:
- ident: ICT-T1
  title: Measure CLK0 frequency
  steps:
  - command: freq 0
  - command: sleepms 1000
  - command: measure frequency 32.75-32.78kHz
- ident: ICT-T2
  title: Measure Impedance
  steps:
  - command: mux 0 RATP02
  - command: mux 1 RATP18
  - command: mux 2 RA
  - command: mux 3 RVREF
  - command: short 0 2 set
  - command: short 1 3 set
  - command: measure impedance 750-1000mOhm 3.3V
  - command: short 0 2 release
  - command: short 1 3 release
  - command: sleepms 1500
  - command: measure impedance 2-10Ohm 3300
- ident: ICT-T3
  title: Measure Currents
  steps:
  - command: measure current3V3 <1A
  - command: measure current5V 0.1-0.5A
  - command: measure currentVARV >100mA
- ident: ICT-T4
  title: Measure Voltages
  steps:
  - command: mux 0 DATP00
  - command: mux 1 DATP01
  - command: mux 2 RATP00
  - command: mux 3 RATP17
  - command: measure voltageMUX0 >1V
  - command: measure voltageMUX1 <3300mV
  - command: measure voltageMUX2 >100mV
  - command: measure voltageMUX3 1500-1800mV
  - command: measure voltageDATP07 200-6000mV
  - command: measure voltageDATP08 -0.2-0.1V
  - command: measure voltageDATP09 0-4V
  - command: measure voltageDATP10 -0.1-3.3V
  - command: measure voltageDATP11 0-3.4V
  - command: measure voltageDATP12 -0.1-3.4V
```

Example: Digital pin measurement

```
title: Digital pin measurements
suite:
- ident: ICT-T1
  title: Configure Digital pins as inputs
  steps:
  - command: pin RDTP21 input pullup
  - command: pin DDTP04 input
- ident: ICT-T2
  title: Read Digital pins
  steps:
  - command: measure pin RDTP21 low
  - command: measure pin DDTP04 high
```

1.2.17 mux - Multiplex Control

Select a probe or signal for a multiplex channel.

Usage:

```
- command: mux <channel> <signal>
```

Argument	Description
channel	Multiplex channel (0..3)
signal	Probe or signal

The following signals and probes can be assigned to a multiplex channel:

signal	MUX	Description
`DATP00`..`DATP06`	0, 1	Direct Analog Test Probes
`RATP00`..`RATP31`	0, 1	Routed Analog Test Probes
`DDTP00`..`DDTP08`	2, 3	Direct Digital Test Probes
`RVREF`	all	Impedance measurement, reference voltage
`RA`	all	Impedance measurement, test voltage
`GND`	all	Ground
`3V3OUT`	all	3.3V power rail
`5VOUT`	all	5.0V power rail
`VARVDIV`	all	VARV power rail, after 1/3 voltage divider

Example: Multiplex control

```
title: Multiplex control example
suite:
- ident: ICT-T1
  title: Setup MUX
  steps:
  - command: mux 0 DATP02
  - command: mux 1 DATP03
  - command: mux 2 RA
  - command: mux 3 RVREF
  - command: short 0 2 set
  - command: short 1 3 set
- ident: ICT-T2
  title: Measure impedance DATP02..DATP03
  steps:
  - command: measure impedance 10-200kOhm
```

1.2.18 nfc - NFC Handling

Manipulate NFC cards.

Usage:

```
- command: nfc write <TAGTYPE> <payload>...
  [retry: <retryCount>]
```

Argument	Description
tagtype	Tag type (`NTAG203`, `NTAG213` or `NTAG216`)
payload	NDEF payload (`text:....`)
retryCount	Number of times to retry the test step (optional)

Example: Program NDEF message with two text records.

```
- ident: ICT-T1
  title: Write NFC
  steps:
  - command: define CODE 123
  - command: nfc write NTAG213 text:"Sample Text" text:%CODE%
```

1.2.19 operator - Operator Test

Instruct operator to perform a manual test step.

Usage:

```
- command: operator <message>
  [retry: <retryCount>]
```

Argument	Description
message	Message to show on the PLT display
retryCount	Number of times to retry the test step (optional)

Example: Instruct operator to inspect housing.

```
- ident: ICT-T1
  title: Visual Inspection (manual)
  steps:
  - command: operator "Inspect Housing"
```

1.2.20 pin - Digital pin control

Configure a Digital probe.

Usage:

```
- command: pin <probe> input [pullup]
```

```
- command: pin <probe> output [low|high]
```

Argument	Description
probe	Probe (DDTPxx, RDTPxx, UARTx_RTS or SWD_NRST)
input	Configure as input
pullup	Enable pull-up (only for RDTPxx pins)
output	Configure as output
low	Set low
high	Set high

Example: Control digital pins

```
title: Control digital pins
suite:
- ident: ICT-T1
  title: Configure digital inputs
  steps:
  - command: pin DDTP05 input
  - command: pin DDTP03 input
  - command: pin RDTP04 input pullup
- ident: ICT-T2
  title: Control digital outputs
  steps:
  - command: pin DDTP00 output high
  - command: pin DDTP02 output low
  - command: pin RDTP01 output high
  - command: pin SWD_NRST output low
  - command: pin UART1_RTS output high
```

1.2.21 power - Power Control

Control power applied to the Device Under Test.

Usage:

```
- command: power off

- command: power <rail> [on|off|<level>]
```

Argument	Description
rail	Power rail
on	Turn on the specified power rail
off	Turn off all or specified power rail
level	Voltage level for **VARV**: 2..12.0

Supported *rail* values:

rail	Description
3V3	3.3V power rail
5V	5V power rail
VARV	Variable power rail
12V	12V power rail
SWD	SWD (3.3V) power rail
JTAG	JTAG (3.3V) power rail
UART0	UART0 (3.3V) power rail
UART1	UART1 (3.3V) power rail

Example: Apply power

```
title: Power control example
suite:
- ident: ICT-T1
  title: Apply power
  steps:
  - command: power 3V3 on
  - command: power 5V on
  - command: power VARV 10.2
- ident: ICT-T2
  title: Wait
  steps:
  - command: sleepms 1000
- ident: ICT-T3
  title: Power off
  steps:
  - command: power off
```

1.2.22 program - Program DUT

Erase and Program DUT MCU on-board flash.

Usage:

```
- command: program <target> [UART0|UART1] [offset:]<img>[,[offset:]<img>[,...]]
[noerase]
  [retry: <retryCount>]

- command: program CMSIS [offset:]<img>
  pack: <cmsis_pack>
  target: <cmsis_target>
  [retry: <retryCount>]
```

Argument	Description
target	Target to program
UART0, UART1	Port for UART targets

Argument	Description
offset	Optional: Offset image
img	Firmware Element
...	Optional: Additional offsets, images
cmsis_pack	For `cmsis` target: CMSIS Pack file
cmsis_target	For `cmsis` target: Target name, as defined in CMSIS Pack file
retryCount	Number of times to retry the test step (optional)

Firmware images can be provided as Intel HEX (`.hex`), ELF (`.elf` / `.axf`) or binary files (requiring an offset).

Supported *target* values:

target	Description
`AVRATmega168P_ISP`	Atmel AVR ATmega168P(A) (JTAG:ISP)
`AVRATmega168P_XPm`	Atmel AVR ATmega168P(A) (USB, XPmini)
`AVRATmega168PB_ISP`	Atmel AVR ATmega168PB (JTAG:ISP)
`AVRATmega168PB_XPm`	Atmel AVR ATmega168PB (USB, XPmini)
`CC1352`	TI CC1352 RFSoC
`CMSIS`	CMSIS-DAP programmer, requires `pack:` and `target:` fields (USB)
`DA14580`	Dialog DA14580 RFSoC
`ESP32`	Espressif ESP32 RFSoC (JTAG)
`ESP32_JTAG`	Espressif ESP32 RFSoC (JTAG)
`ESP32_UART`	Espressif ESP32 RFSoC (UART)
`LPC11xx`	NXP LPC11xx MCU (SWD)
`nRF52`	Nordic nRF52 RFSoC (SWD)
`nRF52_JLink`	Nordic nRF52 RFSoC (USB, JLink)
`nRF91`	Nordic nRF9160 RFSoC:Cortex-M33 (SWD)
`nRF91_JLink`	Nordic nRF9160 RFSoC:Cortex-M33 (USB, JLink)
`PN7360`	NXP PN7360 (SWD)
`PN7362`	NXP PN7362 (SWD)
`PN7460`	NXP PN7460 (SWD)
`PN7462`	NXP PN7462 (SWD)
`PSoC4`	Cypress PSoC4 (SWD)
`PSoC6`	Cypress PSoC6 (SWD)
`PSoC6_KP3`	Cypress PSoC6 (USB, KitProg3)
`STM32F2`	ST STM32F2xx MCU (SWD)
`STM32F2_STLink`	ST STM32F2xx MCU (USB, ST-Link)
`STM32F4`	ST STM32F4xx MCU (SWD)
`STM32F4_STLink`	ST STM32F4xx MCU (USB, ST-Link)
`STM32L4`	ST STM32L4xx MCU (SWD)
`STM32L4_STLink`	ST STM32L4xx MCU (USB, ST-Link)

Example: Program NXP LPC11C24 over SWD, using built-in PLT programmer.

```
title: "NXP LPC11C24 (SWD)"
identPrefix: ICT-
suite:
- title: Power via SWD_VDD
  steps:
  - command: power SWD on
- title: Program firmware
  steps:
  - command: program lpc11xx blinky.hex
```

Example: Program NXP LPC11C24 over a USB-attached CMSIS-DAP programmer, using the Keil `Keil.LPC1100_DFP.1.4.0.pack` CMSIS Pack file.

```
title: "NXP LPC11C24 (CMSIS-DAP)"
identPrefix: ICT-
suite:
- title: Power via 3V3
  steps:
  - command: power 3V3 on
- title: Program firmware
  steps:
  - command: program cmsis blinky.hex
    pack: Keil.LPC1100_DFP.1.4.0.pack
    target: LPC11C24FBD48/301
```

1.2.23 scan - Scan Barcode

Scan a barcode using USB-attached barcode scanner.

Usage:

```
- command: scan <format>
  [extractKey: <key>]
  [retry: <retryCount>]
```

Argument	Description
format	Format of code to scan
key	Key in which to store the barcode value (optional)
retryCount	Number of times to retry the test step (optional)

Supported *format* values:

format	Description
ANY	Any barcode
%MAC_ADDRESS%	48-bit MAC address (`XX:XX:XX:XX:XX:XX`)

```
title: Scan serial number
suite:
- steps:
  - command: scan ANY
    extractKey: MY_BARCODE
  - command: serial set %MY_BARCODE%
```

1.2.24 serial - Request or set serial number

Set serial number, from test plan or through PLTcloud.

Under Project Settings in PLTcloud, serial number allocation can be configured to use either monotonic counters, or through a webhook.

Usage:

```
- command: serial request [<KEY>:<jsonKey>]...
```

```
- command: serial set <value>
```

Argument	Description
request	Request serial number through PLTcloud backend
KEY	Test plan user key to extract from webhook response
jsonKey	Webhook response "extra" map key
set	Request serial number through PLTcloud backend

When using serial request, and a webhook is selected in PLTcloud, a JSON request will be sent containing:

```
{
    "dut_id": "<DUT IDENTIFIER>",
    "serial_number": "<SERIAL NUMBER>",
    "ble_mac": "<BLE MAC>",
    "mcu_id": "<MCU ID>",
    "MYCUSTOMKEY": "my custom value"
}
```

The API endpoint should return something like:

```
{
    "serial_number": "<SERIAL NUMBER>",
    "extra": {
        "name": "Ben"
    }
}
```

To extract the name field, use the serial request command as below:

```
title: Eventbrite
suite:
- ident: SCAN
  title: Scan
  steps:
  - command: scan ANY
    extractKey: BARCODE
- ident: WEBHOOK
  title: Lookup barcode
  steps:
  - command: serial request NAME:name
  - command: operator %NAME%
- ident: PRINT
  title: Print badge
  steps:
  - command: label keys NAME
```

Using serial set, the serial number can be set directly from the test plan, for example using a key extracted from a barcode scan:

```
title: Scan serial number
suite:
- steps:
  - command: scan ANY
    extractKey: BARCODE
  - command: serial set %BARCODE%
```

1.2.25 short - Connect Multiplex channels

Usage:

```
- command: short <muxA> <muxB> set|release
```

Argument	Description

Argument	Description
muxA	Multiplex channel (0..3)
muxB	Multiplex channel (0..3)
`set`	Short specified multiplex channels
`release`	Release short between specified multiplex channels

Example: connect multiplex channels

```
title: Multiplex example
suite:
- ident: ICT-T1
  title: Control shorts
  steps:
  - command: short 0 1 set
  - command: short 1 3 set
  - command: sleepms 1000
- ident: ICT-T1
  title: 0-2 instead of 0-1
  steps:
  - command: short 0 1 release
  - command: short 0 2 set
```

1.2.26 sleepms - Delay

Temporarily suspend test suite execution.

Usage:

```
- command: sleepms <duration>
```

Argument	Description
duration	Duration, in milliseconds

1.2.27 uart - Send and Extract UART response

Extract data from UART.

Usage:

```
- uartcmd: uart UART0|UART1 [noflush]
  [expect: <expectText>]
  [extract: <extractText>]
  [extractKey: <key>...]
  [send: <sendText>]
  [retry: <retryCount>]
```

Argument	Description
`UART0`, `UART1`	UART port
expectText	Text to expect, prior to extraction
extractText	Regular expression to extract
key	Key(s) in which to store extracted text
sendText	Text to send prior to extaction
`noflush`	Don't flush receive buffer before extraction
retryCount	Number of times to retry the test step (optional)

Example: extract ICCID from cellular modem, storing in the `ICCID` key.

```
- uartcmd: uart UART0
  expect: "+CCID:"
  extract: "CCID: (\\d{20})\r\n"
  extractKey: ICCID
  send: "AT+ICCID\r\n"
```

1.2.28 uartAwait - Await UART response

Wait for a specific UART response.

Usage:

```
- command: uartAwait UART0|UART1 <seconds>
  [retry: <retryCount>]
```

Argument	Description
UART0, UART1	UART port
seconds	Time to await response, in seconds
retryCount	Number of times to retry the test step (optional)

Example:

```
- command: uartExpect UART0 Pressed
- command: operator "Press button"
- command: uartAwait UART0 1
```

1.2.29 uartCfg - Configure UART

Configure a UART port.

Usage:

```
- command: uartCfg UART0|UART1 <speed> [<triplet>]
```

```
- command: uartCfg UART0 tp
```

Argument	Description
UART0, UART1	UART port
speed	Baud rate
triplet	UART configuration triplet: 8N1 or 7E1
tp	Use alternate test points instead of UART. (Only for UART0)

Example:

```
- command: uartCfg UART1 9600 8N1
- command: uartCfg UART0 tp
```

1.2.30 uartExpect - Set expectation for uartAwait

Set a UART response to wait for with the uartAwait command.

Usage:

```
- command: uartExpect UART0|UART1 <expect> [noflush]
  [retry: <retryCount>]
```

Argument	Description
UART0, UART1	UART port
expect	String to expect with subsequent uartAwait command
noflush	Don't flush receive buffer before extraction

Argument	Description
retryCount	Number of times to retry the test step (optional)

Example:

```
- command: uartExpect UART0 Pressed
- command: operator "Press button"
- command: uartAwait UART0 1
```

1.2.31 uartReadTimeout - Test if UART is not transmitting

Test if nothing is received from UART.

Usage:

```
- command: uartReadTimeout UART0|UART1 <seconds> [<sendText>]
  [retry: <retryCount>]
```

Argument	Description
`UART0`, `UART1`	UART port
seconds	Number of seconds to wait for incoming data
sendText	Text to send before waiting
retryCount	Number of times to retry the test step (optional)

Example: Test if modem is shut down.

```
- command: uartReadTimeout UART0 1 "AT"
```

1.2.32 webhook - Invoke webhook

Invoke a webhook through PLTcloud. The webhook HTTPS endpoint is configured in Project Settings on PLTcloud.

Usage:

```
- command: webhook request [<KEY>:<jsonKey>]...
  [retry: <retryCount>]
```

Argument	Description
`request`	Invoke webhook through PLTcloud backend
KEY	Test plan user key to extract from webhook response
jsonKey	Webhook response "extra" map key
retryCount	Number of times to retry the test step (optional)

When a webhook is selected in PLTcloud, a JSON request will be sent containing:

```
{
    "dut_id": "<DUT IDENTIFIER>",
    "serial_number": "<SERIAL NUMBER>",
    "ble_mac": "<BLE MAC>",
    "mcu_id": "<MCU ID>",
    "MYCUSTOMKEY": "my custom value"
}
```

The API endpoint should return something like:

```
{
    "extra": {
        "name": "Ben"
    }
}
```

To extract the name field, use the webhook request command as below:

```
title: Eventbrite
suite:
- ident: SCAN
  title: Scan
  steps:
  - command: scan ANY
    extractKey: BARCODE
- ident: WEBHOOK
  title: Lookup barcode
  steps:
  - command: webhook request NAME:name
  - command: operator %NAME%
- ident: PRINT
  title: Print badge
  steps:
  - command: label keys NAME
```