SNMP-BACNET INTEGRATION DOCUMENTATION

T3 ESP32 Controller

This document provides comprehensive technical documentation for the SNMP-BACnet integration implemented in the T3 ESP32 programmable controller.

info@electrobittech.com

Contents

Summary

This document provides comprehensive technical documentation for the SNMP-BACnet integration implemented in the T3 ESP32 programmable controller. The integration enables seamless communication between SNMP network management systems and BACnet building automation protocols, providing complete T3 specification compliance.

Key Achievements

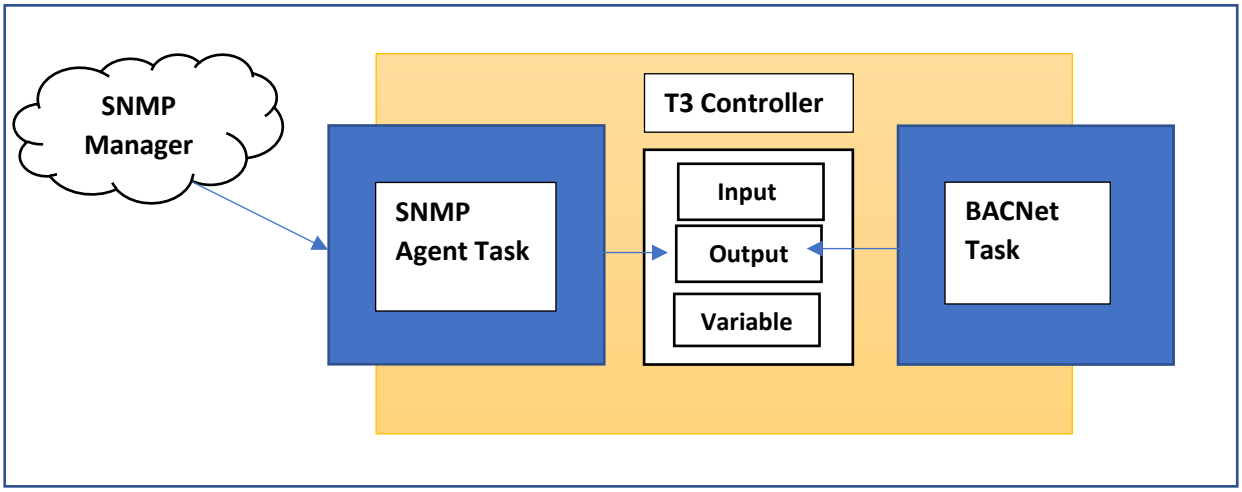
1. Full T3 specification compliance with enterprise OID `1.3.6.1.4.1.64991.1`
2. Complete mapping of 14 configuration types to BACnet objects
3. Real-time bidirectional data exchange between SNMP and BACnet
4. Production-ready implementation with comprehensive error handling
5. Support for 1,152 SNMP MIB entries (3 groups × 6 fields × 64 instances)

System Architecture

Overview

The T3 SNMP-BACnet integration acts as a bridge between two critical building automation protocols:

Component Architecture



T3 Specification Compliance

Enterprise OID Structure

1.3.6.1.4.1. 64991.<group>.<field>.<instance>

├── 1.3.6.1.4.1 = Enterprise OID (TEMCO)
├── 64991.1 = T3 Product Family
├── .Group = [1] Inputs Group / [2] Outputs Group / [3] Variables Group
├── .Field = Specific Field [0-4]
└── .instance = Instance [0-63]

Object Groups and Fields

Group	OID Base	Purpose	Instances
t3Inputs	1.3.6.1.4.1. 64991.1.1	Input Objects	0 - 63
t3Outputs	1.3.6.1.4.1. 64991.1.2	Output Objects	0 – 63
t3Variables	1.3.6.1.4.1. 64991.1.3	Variable Objects	0 - 63

Field Definitions

Field	OID Suffix	SNMP Type	Access	Description
index	.0	INTEGER	RD_ONLY	Instance number
cfgType	.1	INTEGER	RD_ONLY	Configuration type
analogVal	.2	GAUGE	RD/WR*	Analog value
binaryVal	.3	INTEGER	RD/WR*	Binary value
desc	.4	OCTET_STRING	RD/WR*	Description
units	.5	INTEGER	RD_ONLY	Engineering units

*Write access depends on object group (inputs are read-only)

SNMP-BACnet Object Mappings

Configuration Type Mapping (cfgType)

cfgType	Value	BACnet Object	Description	Range
T3_CFGTYPE_BI	1	BINARY_INPUT	Binary Input	0/1
T3_CFGTYPE_AI_0_100	2	ANALOG_INPUT	Analog Input 0-100%	0-100
T3_CFGTYPE_AI_0_10V	3	ANALOG_INPUT	Analog Input 0-10V	0-10V
T3_CFGTYPE_AI_4_20MA	4	ANALOG_INPUT	Analog Input 4-20mA	4-20mA
T3_CFGTYPE_AI_NEG10_10V	5	ANALOG_INPUT	Analog Input -10 to +10V	-10V to +10V
T3_CFGTYPE_AI_0_5V	6	ANALOG_INPUT	Analog Input 0-5V	0-5V
T3_CFGTYPE_BO	7	BINARY_OUTPUT	Binary Output	0/1
T3_CFGTYPE_AO_0_10V	8	ANALOG_OUTPUT	Analog Output 0-10V	0-10V
T3_CFGTYPE_AO_4_20MA	9	ANALOG_OUTPUT	Analog Output 4-20mA	4-20mA

T3_CFGTYPE_AO_0_100	10	ANALOG_OUTPUT	Analog Output 0-100%	0-100%
T3_CFGTYPE_VAR_INT	11	CHARACTERSTRING_VALUE	Integer Variable	-2^{31} to $2^{31}-1$
T3_CFGTYPE_VAR_FLOAT	12	CHARACTERSTRING_VALUE	Float Variable	$\pm 3.4\times10^{3}$
T3_CFGTYPE_VAR_STRING	13	CHARACTERSTRING_VALUE	String Variable	64 chars
T3_CFGTYPE_RESERVED	14	N/A	Reserved	N/A

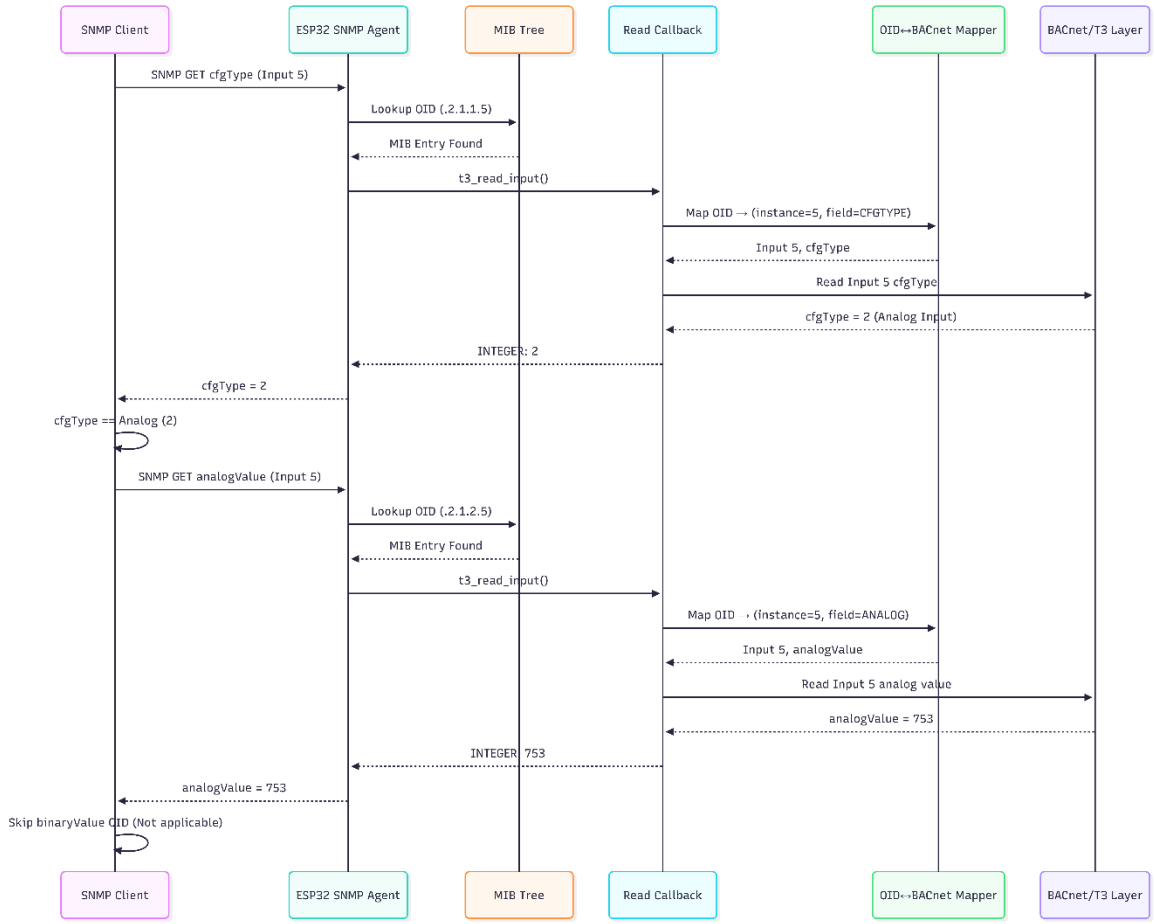
Data Type Conversion Matrix

SNMP Type	BACnet Type	Conversion	Precision
GAUGE	REAL	Float × 1000 → Integer	3 decimal places
INTEGER	INTEGER	Direct mapping	N/A
OCTET_STRING	CHARACTER_STRING	Direct mapping	N/A

Engineering Units Mapping

T3 Units	BACnet Units	Description
T3_UNITS_NONE	UNITS_NO_UNITS	No units
T3_UNITS_CELSIUS	UNITS_DEGREES_CELSIUS	Degrees Celsius
T3_UNITS_FAHRENHEIT	UNITS_DEGREES_FAHRENHEIT	Degrees Fahrenheit
T3_UNITS_PERCENT	UNITS_PERCENT	Percentage
T3_UNITS_PASCAL	UNITS_PASCALS	Pascals
T3_UNITS_PSI	UNITS_PASCALS	PSI (converted to Pa)
T3_UNITS_MILLIAMPS	UNITS_MILLIAMPERES	Milliamperes
T3_UNITS_VOLTS	UNITS_VOLTS	Volts
T3_UNITS_HERTZ	UNITS_HERTZ	Hertz
T3_UNITS_SECONDS	UNITS_SECONDS	Seconds

Implementation Details



File Structure

main/
- ├── snmp_interface.c # Main SNMP interface (T3-compliant)
- ├── t3_snmp_bacnet_mapping.h # Core mapping definitions
- ├── t3_snmp_bacnet_mapping.c # Mapping implementation

Components
- ├── **snmp_agent** ├── **include** (snmp agent core header files)
- ├── **src** (snmp agent core source files)
- ├── CMakefile.txt

snmp_agent.h --> **uSNMP core agent APIs**
snmp_interface.h --> **Public SNMP interface**
t3_snmp_bacnet_mapping.h --> **SNMP ↔ BACnet OID mapping**

Once application gets IP address, FreeRTOS application needs to only call `snmp_app_init()`. It will create a separate thread.

The agent exposes

- Standard MIB-II system objects
- Private enterprise MIBs mapped to T3 controller BACnet objects
- Support for SNMP GET / WALK / SET
- Cold-start and enterprise-specific traps
- Dynamic value handling via read/write callbacks
- The SNMP agent runs as a FreeRTOS task and integrates with the existing BACnet/T3 data layer.

Key Features

- SNMP v2c compatible (RO & RW communities)
- Dynamic MIB tree creation
- BACnet ↔ SNMP OID mapping
- Read-only and read-write objects
- Custom enterprise traps
- Modular callback-based design.

Object	OID	Description
sysDescr	1.3.6.1.2.1.1.1.0	Device description
sysObjectID	1.3.6.1.2.1.1.2.0	Enterprise OID
sysUpTime	1.3.6.1.2.1.1.3.0	System uptime
sysContact	1.3.6.1.2.1.1.4.0	Admin contact
sysName	1.3.6.1.2.1.1.5.0	Device name
sysLocation	1.3.6.1.2.1.1.6.0	Physical location

SNMP Agent Task

`static void snmp_agent_task(void *pvParameters)`

- Wait for network readiness.
- Initialize SNMP core.
- Build MIB tree.
- Send start up traps (**ColdStart**).
- Process SNMP requests continuously.

`initSnmpAgent(SNMP_PORT, ENTERPRISE_OID, RO_COMMUNITY, RW_COMMUNITY);`

- Binds SNMP agent to UDP port (161)
- Registers enterprise OID
- Configures community strings

`static void init_standard_mibs(void);`

- Uses prefix **B → 1.3.6.1.2.1**
- Implements standard SNMP system objects
- Uses callbacks for dynamic values (uptime, IP)

SNMP ↔ BACnet Mapping

- Instance number
- Field type
- BACnet object type
- Data representation.

`int t3_read_input(MIB *thismib)`

Supported Data Types:

- Convert OID to string
- Lookup BACnet mapping
- Read value from BACnet
- Populate SNMP response
- Return SNMP status

`int t3_write_output(MIB *thismib, void *ptr, int len)`

- Validate input
- Map OID → BACnet object
- Convert SNMP data to BACnet format
- Write value to controller
- Return SNMP status

Sending Trap command to manager

- **Cold Start Trap**

```
send_snmp_trap_cold_start(my_ip, manager_ip);
```

- **Enterprise Trap**

```
snmp_trap_enterprise(my_ip, manager_ip, "this is custom trap!!!");
```

Testing Procedures

All testing has been done using Net-SNMP CLI tool. This is available in both windows and linux OS.

https://github.com/net-snmp/net-snmp

https://www.net-snmp.org/tutorial/tutorial-5/commands/index.html

https://sourceforge.net/projects/net-snmp/files/net-snmp%20binaries/

Manual Testing

- Reading input object 0 (index, cfgType, analogVal, binaryVal, desc, units)

```
$ snmpget -v2c -L n -c public 192.168.1.15 1.3.6.1.4.1.2026.1.1.0.0
1.3.6.1.4.1.2026.1.1.1.0 1.3.6.1.4.1.2026.1.1.2.0 1.3.6.1.4.1.2026.1.1.3.0
1.3.6.1.4.1.2026.1.1.4.0 1.3.6.1.4.1.2026.1.1.5.0

iso.3.6.1.4.1.64991.1.1.0.0 = INTEGER: 0
iso.3.6.1.4.1.64991.1.1.1.0 = INTEGER: 1
iso.3.6.1.4.1.64991.1.1.2.0 = INTEGER: 0
iso.3.6.1.4.1.64991.1.1.3.0 = INTEGER: 0
iso.3.6.1.4.1.64991.1.1.4.0 = STRING: "INPUT_0"
iso.3.6.1.4.1.64991.1.1.5.0 = INTEGER: 1
```

1. Read the Configuration First:

```
# Ask the device: "How is Input 5 configured?"
snmpget -v2c -c public 192.168.1.100 .1.3.6.1.4.1.2026.2.1.1.5
```

Result: `iso.3.6.1.4.1.2026.2.1.1.5 = INTEGER: 2` (Meaning: It's an Analog Input)

2. Read the Correct Value Based on Configuration:

```
# Ask the device: "What is the analog value of Input 5?"
snmpget -v2c -c public 192.168.1.100 .1.3.6.1.4.1.2026.2.1.2.5
```

Result: `iso.3.6.1.4.1.2026.2.1.2.5 = INTEGER: 753`

The client would ignore the **inputBinaryValue** OID for this input.

- **Read/Write Value for OIDS**

```
snmpset -v2c -L n -c private 192.168.1.15 1.3.6.1.4.1.2026.1.2.3.0 i 30
iso.3.6.1.4.1.2026.1.2.3.0 = INTEGER: 30

snmpget -v2c -L n -c public 192.168.1.15 1.3.6.1.4.1.2026.1.2.3.0
iso.3.6.1.4.1.2026.1.2.3.0 = INTEGER: 30

snmpset -v2c -L n -c private 192.168.1.15 1.3.6.1.4.1.2026.1.3.3.1 i 50
iso.3.6.1.4.1.2026.1.3.3.1 = INTEGER: 50

nmpget -v2c -L n -c public 192.168.1.15 1.3.6.1.4.1.2026.1.3.3.1
iso.3.6.1.4.1.2026.1.3.3.1 = INTEGER: 50

snmpset -v2c -L n -c private 192.168.1.15 1.3.6.1.4.1.2026.1.2.4.0 s OP_T1
iso.3.6.1.4.1.2026.1.2.4.0 = STRING: "OP_T1"

snmpget -v2c -L n -c private 192.168.1.15 1.3.6.1.4.1.2026.1.2.4.0
iso.3.6.1.4.1.2026.1.2.4.0 = STRING: "OP_T1"

nmpget -v2c -L n -c private 192.168.1.15 1.3.6.1.4.1.2026.1.3.4.0
iso.3.6.1.4.1.2026.1.3.4.0 = STRING: "VAR_0"

snmpset -v2c -L n -c private 192.168.1.15 1.3.6.1.4.1.2026.1.3.4.0 s "VAR_XL"
iso.3.6.1.4.1.2026.1.3.4.0 = STRING: "VAR_XL"

snmpget -v2c -L n -c private 192.168.1.15 1.3.6.1.4.1.2026.1.3.4.0
iso.3.6.1.4.1.2026.1.3.4.0 = STRING: "VAR_XL"

snmpget -v2c -L n -c private 192.168.1.15 1.3.6.1.4.1.2026.1.3.2.0
iso.3.6.1.4.1.2026.1.3.2.0 = INTEGER: 0

snmpset -v2c -L n -c private 192.168.1.15 1.3.6.1.4.1.2026.1.3.2.0 i 77
iso.3.6.1.4.1.2026.1.3.2.0 = INTEGER: 77

snmpget -v2c -L n -c private 192.168.1.15 1.3.6.1.4.1.2026.1.3.2.0
iso.3.6.1.4.1.2026.1.3.2.0 = INTEGER: 77
```

- **Testing Trap**
  TRAP message will be sent to remote machine snmp manager. Remote machine needs to start ***snmptrapd*** and then restart esp device.

  There will also an INFORM command send to remote machine and also ACK will be received. In ***snmp_interface.h***, we can configure `TRAP_DST_ADDR` and `ENTERPRISE_OID` macros.

ESP Debug Logs

```
I (4408) esp_netif_handlers: sta ip: 192.168.1.12, mask: 255.255.255.0, gw: 192.168.1.1
I (4408) wifi station: got ip:192.168.1.12
I (4408) wifi station: Starting SNMP agent
I (4408) snmp_agent: T3 SNMP agent initialized
I (4418) wifi station: connected to ap SSID:FM360 password:novakdjokovic
I (7418) snmp_agent: Initializing T3 SNMP MIB tree (instances=64)
I (9338) snmp_agent: initMibTree complete
I (9338) SNMP_TRAP: SNMP TRAP sent to 192.168.1.10
I (9338) SNMP_TRAP: SNMP INFORM sent to 192.168.1.10
I (9528) wifi:<ba-add>idx:1 (ifx:0, 0a:aa:89:9a:33:aa), tid:6, ssn:2, winSize:64
I (9838) SNMP_TRAP: INFORM ACK received
```

SNMP NET CLI Logs

```
$ snmptrapd -f -Lo -n
NET-SNMP version 5.5.1
2026-01-27 20:57:59 UDP: [192.168.1.12]:57255->[0.0.0.0] [UDP: [192.168.1.12]:57255->[0.0.0.0]]:
iso.3.6.1.2.1.1.3.0 = Timeticks: (882) 0:00:08.82      iso.3.6.1.6.3.1.1.4.1.0 = OID: iso.3.6.1.6.3.1.1.5.1
2026-01-27 20:57:59 UDP: [192.168.1.12]:57255->[0.0.0.0] [UDP: [192.168.1.12]:57255->[0.0.0.0]]:
iso.3.6.1.2.1.1.3.0 = Timeticks: (882) 0:00:08.82      iso.3.6.1.6.3.1.1.4.1.0 = OID: iso.3.6.1.6.3.1.1.5.1
2026-01-27 20:57:59 UDP: [192.168.1.12]:57256->[0.0.0.0] [UDP: [192.168.1.12]:57256->[0.0.0.0]]:
iso.3.6.1.2.1.1.3.0 = Timeticks: (882) 0:00:08.82      iso.3.6.1.6.3.1.1.4.1.0 = OID: iso.3.6.1.4.1.999.1      iso.3.6.1.4.1.999.1.1 =
STRING: "this is custom trap!!!"
```

Testing Using Scripts

In script directory, there are three bat scripts which we can used to read all objects in bulk or individually. There scripts internally use SNMP Net CLI tools.

Below command reads all objects (0-63) of particular types.

- read_T3_input.bat 192.168.1.12
- read_T3_ouput.bat 192.168.1.12
- read_T3_variable.bat 192.168.1.12

Reading particular object by providing object index in argument.

read_T3_input.bat 192.168.1.12 1

```
==============================
Testing Object 1
==============================
iso.3.6.1.4.1.64991.1.1.0.1 = INTEGER: 1
iso.3.6.1.4.1.64991.1.1.1.1 = INTEGER: 1
iso.3.6.1.4.1.64991.1.1.2.1 = INTEGER: 100
iso.3.6.1.4.1.64991.1.1.3.1 = INTEGER: 100
iso.3.6.1.4.1.64991.1.1.4.1 = STRING: "INPUT_1"
iso.3.6.1.4.1.64991.1.1.5.1 = INTEGER: 1

Done.
```