DeviceTree vtechstudy : Lesson_2

Q.1) Conceptually, a common set of usage conventions, called _____, is defined for how data should appear in the tree to describe typical hardware characteristics.

A) mappings

B) protocols

C) bindings

D) groupings

Q.2) Use dtc on the .dts file in this folder to create . dtb file:

- **dtc -I dts -o sample-machine.dtb -O dtb sample-machine.dts**

The sample-machine.dts, when compiled using dtc, generates two warnings, the first message is:

Warning (unit_address_vs_reg): /external-bus: node has a reg or ranges property, but no unit name

What is a unit name? (select all that apply)

A) specifies the data type (data units) for the property, e.g. unsigned integer or string

B) a portion of the node-name that helps differentiate the node

C) a monotonically increasing number appended to some property names

D) a number that represents the base address of the node in memory

Q.3) The second message is:

Warning (i2c_bus_reg): /external-bus/i2c@1,0/rtc@58: I2C bus unit address format error, expected "3a"

Attempt to resolve the two warning messages by making two 1-line changes in the sample-machine.dts file.

Warning (unit_address_vs_reg): /external-bus: node has a reg or ranges property, but no unit name

<line number>, <corrected line>

Warning (i2c_bus_reg): /external-bus/i2c@1,0/rtc@58: I2C bus unit address format error, expected "3a"

<line number>, <corrected line>

Q.4) What is the default data type for DTS|DTB files?

A) uint8_t

B) uint16_t

C) uint32_t

D) uint64_t

Q.5) What syntax in .DTS file allows the data size to be specified as a byte?

A) {size} 8

B) /bytes/ 1

C) {memory} 1

D) /bits/ 8

In lesson_2/ folder, run make and then this command-line**: ./lesson_2 -f sample-machine.dtb**

./lesson_2 running...

[main] read dtb_blob[sample-machine.dtb]..size[1470]

config property

00 00 ab cd 00 00 12 34 00 00 fe 98 .....4....

example() returned [FDT_ERR_QC_NOERROR]...

A new property (config) has been added to the DeviceTree and its value is dumped as a string of hex digits, not unlike the output of hexdump. Use QDTE and confirm you can see this property and its values in the GUI.

Add **-t** (for trace) to the lesson_2 command-line and you will see additional data displayed. The additional output shows that the client program can, in fact, interpret the data in any way it sees fit. The rest of this module explores the DTB format and why it is possible for the client to have such flexibility. I leave it to the reader to decide if this flexibility is a good or bad thing.

Open fdt.h in an editor and study the structure definitions and defined macros. This file basically lays out everything you need to know about the DTB format. Look for this file in the same folder you found libfdt.h.

Assuming you still have QDTE open, use View | Raw to open a new window that displays a raw hex dump of the DTB. Scroll through this window to get a feel for what a DTB looks like in memory. As you scroll through you will notice there are strings of ascii data spread throughout the file and at the bottom of the file you will see a whole block of ascii data.

Q.6) What does the block of ascii data at the bottom of the file represent?

A) nodes with phandles

B) node names

C) property names

D) string property values

Q.7) What is the value for the off_dt_struct field of the header?

A) 00 00 00 28

B) 00 00 05 3C

C) 00 00 05 BE

D) 00 00 00 38

Q.8) If you treat the off_dt_struct value as an offset from the base of the file, what is the value at that offset?

A) FDT_MAGIC

B) FDT_BEGIN_NODE

C) FDT_PROP

D) FDT_END_NODE

Q.9) What is the offset for the first struct fdt_property in the file?

Q.10) The data portion of the first property ends at what offset?

A) 0x4C

B) 0x5F

C) 0x60

D) 0x61

E) 0x63

From fdt.h, this is structure definition for fdt_property:

```
struct fdt_property {
        fdt32_t tag;
        fdt32_t len;
        fdt32_t nameoff;
        char data[0];
};
```

Q.11) Match the fdt_property field with a valid description of its purpose

A) length of this fdt_property instance

B) offset of this fdt_property within DTB

C) identify start of fdt_property structure

D) type of data stored in this fdt_property struct

E) length of data in this fdt_property

F) location of property name

G) length of property name

H) start of data stored in this property

From fdt.h, this is structure definition for fdt_node_header:

```
struct fdt_node_header {
        fdt32_t tag;
        char name[0];
};
```

Q.12) Match the fdt_node_header field with a valid description of its purpose

A) start of data stored in this node

B) type of data stored in this node

C) identify start of fdt_node_header field structure

D) length of data in this node

E) start of node name

Q.13) At what offset is the end of the root node?

A) 0x14C

B) 0x260

C) 0x520

D) 0x534

A.14) Expressed as a macro, what value is in the next field in the DTB?

A) FDT_END_NODE

B) FDT_BEGIN_NODE

C) FDT_END

D) FDT_NOP

Extra Credit: Why are node names and property names handled differently by the DTB layout?

From QDTE, View | Values As… | Hexadecimal

You can now scroll through the raw window next to the GUI and walk through the raw listing, node by node and property by property.

The information in the fdt_property structure is the only information the FdtLib APIs have to work with when extracting data from the DTB at runtime. Specifically missing from this structure is a data type specifier. The library has a location for the data and a length (in bytes) for how much data is stored in that property. The client has complete freedom to interpret and use the data in any manner.