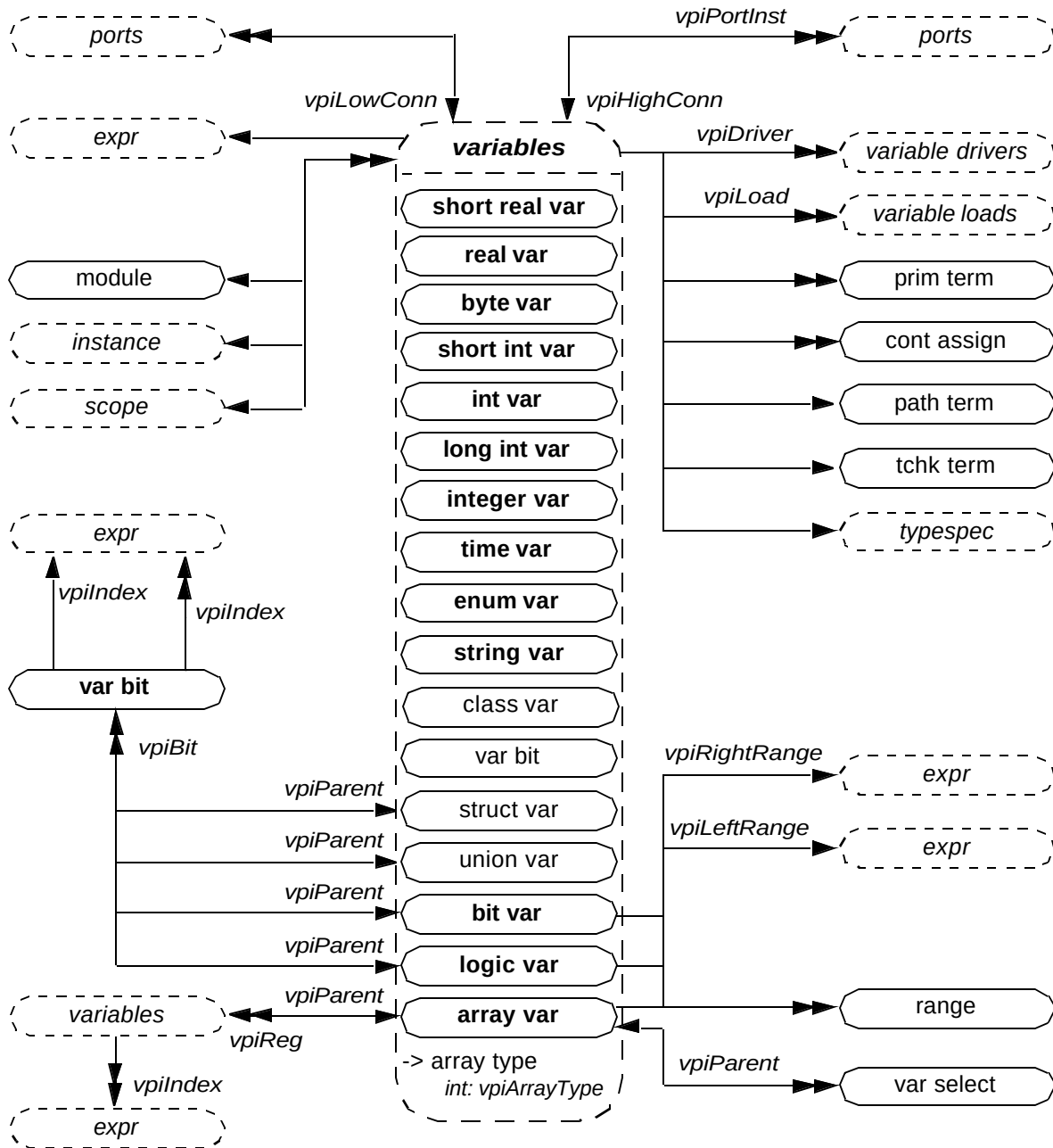


Section 32.14 Variables (supersedes P1364 26.6.7 and 26.6.8)

REPLACE



-> array member
 bool: vpiArray (deprecated)
 bool: vpiArrayMember

-> name
 str: vpiName
 str: vpiFullName

-> sign
 bool: vpiSigned

-> size
 int: vpiSize

-> array type
 int: vpiArrayType

-> lifetime
 bool: vpiAutomatic

-> constant variable
 bool: vpiConstantVariable

-> determine random availability
 bool: vpiIsRandomized

-> randomization type
 int: vpiRandType

-> member
 bool: vpiMember

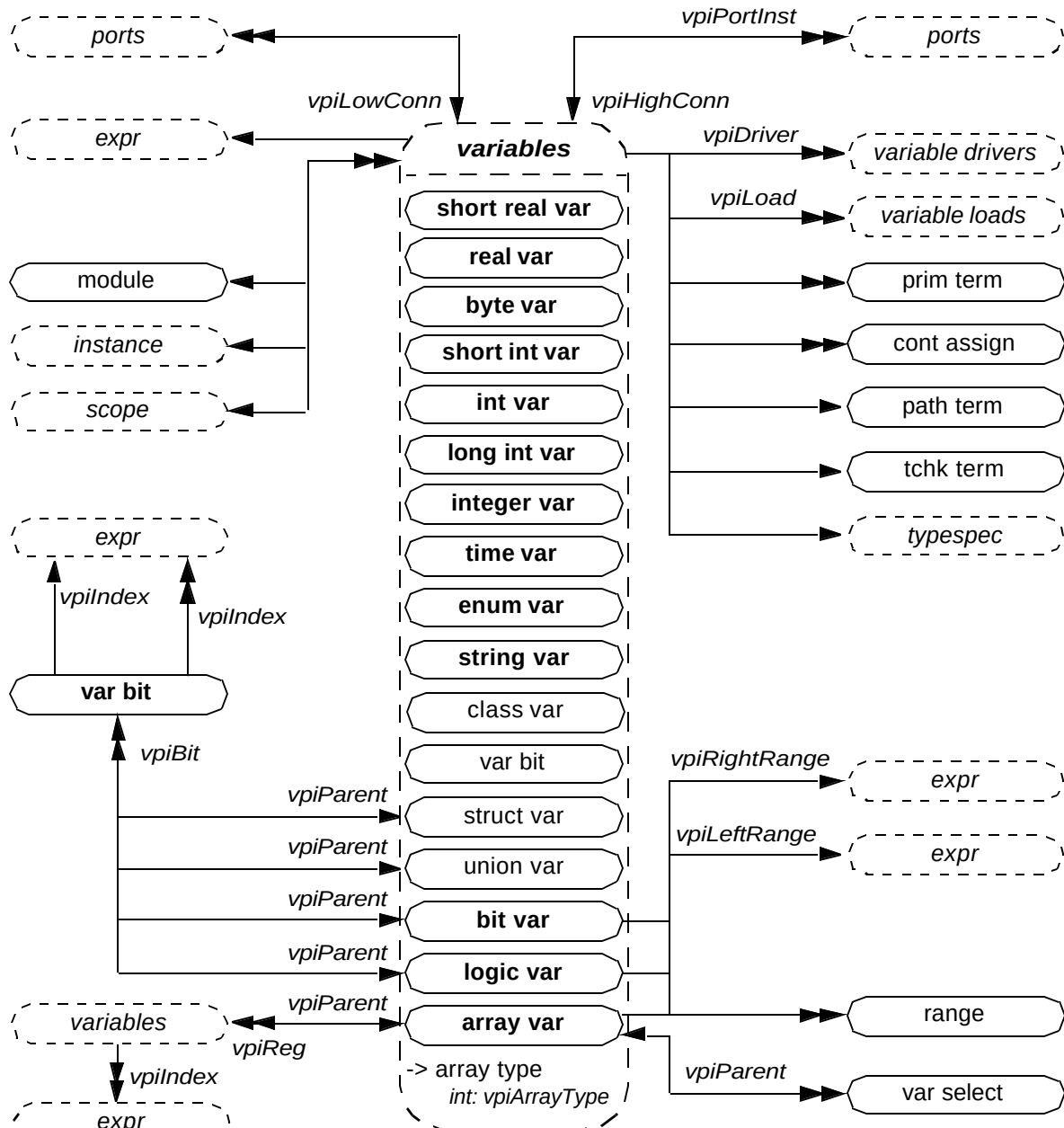
-> value
 vpi_get_value()
 vpi_put_value()

-> scalar
 bool: vpiScalar

-> visibility
 int: vpiVisibility

-> vector
 bool: vpiVector

WITH



-> array member
bool: vpiArray (deprecated)
bool: vpiArrayMember
-> name
str: vpiName
str: vpiFullName
-> sign
bool: vpiSigned
-> size
int: vpiSize
-> array type
int: vpiArrayType

-> lifetime
bool: vpiAutomatic
-> constant variable
bool: vpiConstantVariable
-> determine random availability
bool: vpiIsRandomized
-> randomization type
int: vpiRandType
-> member
bool: vpiMember

->value
vpi_get_value()
vpi_put_value()
-> scalar
bool: vpiScalar
-> visibility
int: vpiVisibility
-> vector
bool: vpiVector
-> valid
int: vpiValid

Note to editor: The new text should precede the original NOTES section as indicated below. (According to proposal 266, the original NOTES section will be renamed DETAILS.) The NOTE in the new text is informative rather than normative, and hence uses the modal auxiliary “will” rather than “shall”. The new NOTE clarifies the implications of the definitions of **vpiValidTrue** and **vpiValidFalse**.

REPLACE

NOTES

1)....

WITH

A value of **vpiValidTrue** for the property **vpiValid** shall indicate that the application may continue to access the properties, relationships, and value of the variable indicated by the reference handle. A value of **vpiValidFalse** shall indicate that the variable indicated by the reference handle can no longer be accessed. A variable may cease to exist, for example, if the scope terminates in which an automatic variable is declared (6.6), or because the object is reclaimed by automatic memory management (12.26).

If an implementation is unable to determine whether or not the associated variable still exists, it may cause **vpiValid** to return a value of **vpiValidUnknown**. In this case, the application may attempt to reacquire a handle to the variable by starting from a static handle or from a handle for which **vpiValid** returns **vpiValidTrue**.

NOTE -- An attempt to reacquire a handle to a variable for which **vpiValid** returns **vpiValidFalse** will always fail. An attempt to reacquire a handle to a variable for which **vpiValid** returns **vpiValidTrue** is unnecessary, but will always succeed.

It shall be an error for an application to attempt to obtain a property, relationship, or value of an invalid variable.

NOTES

1)...

Annex I (normative)

vpi_user.h

REPLACE

```
/****** task/function properties *****/  
#define vpiOtherFunc      6 /* returns other types; for property vpiFuncType */
```

WITH

```
/****** task/function properties *****/  
#define vpiOtherFunc      6 /* returns other types; for property vpiFuncType */  
  
/****** value for vpiValid *****/  
#define vpiValidUnknown   2 /* Validity of variable is unknown */
```