CrossWorks CoreBASIC Library

Version: 3.1

Contents

CrossWorks CoreBASIC Library

CrossWorks CoreBASIC Library

About the CrossWorks CoreBASIC Library

The *CrossWorks CoreBASIC Library* is an application that makes extensive use of the software components in the CrossWorks Target Library.

The components that CoreBASIC Library uses are:

- *CrossWorks Platform Library*: provides base platform services.
- *CrossWorks Device Library*: provides drivers for common digital sensors, such as accelerometers, gyroscopes, magnetometers, and so on.
- *CrossWorks Shield Library*: provides drivers for a range of Arduino-style shields.
- *CrossWorks Graphics Library*: is a library of simple graphics functions for readily-available LCD controllers.

Architecture

The *CrossWorks CoreBASIC Library* is one part of the *CrossWorks Target Library*. Many of the low-level functions provided by the target library are built using features of the *CrossWorks Tasking Library* for multi-threaded operation.

Delivery format

The *CrossWorks CoreBASIC Library* is delivered in source form.

License

The source files in this package are not public domain and are not open source. They represent a substantial investment undertaken by Rowley Associates to assist CrossWorks customers in prototyping solutions using well-written, tested drivers.

CrossWorks CoreBASIC Library

Should you wish to incorporate CoreBASIC in a product, you will need to purchase a Commercial Use license for CoreBASIC.

Feedback

This facility is a work in progress and may undergo rapid change. If you have comments, observations, suggestions, or problems, please feel free to air them on the **CrossWorks Target and Platform API** discussion forum.

CrossWorks CoreBASIC Library

Parsing

CoreBASIC uses a tokenized form of a program and the process by which user input is converted to tokens is called *parsing*.

Any keywords from the **nb_core_keywords** list are converted to single-byte tokens. Integers and floating point values are converted from ASCII to binary form and stored in the token stream.

CrossWorks CoreBASIC Library

Memory

Cells

Memory in the interpreter is divided into 8-byte *cells* which can hold a single variable and its value, a value, or some program text. The type **nb_cell_t** describes the layout of each cell. Cells are combined to hold longer, more complex objects such as the tokenized program text, strings, and arrays.

Scratch memory

The interpreter stores variables and the expression stack in the *scratch area* at the beginning of memory: variables are allocated from the beginning of scratch memory in increasing address order, whilst expressions are placed onto the expression stack at the end of the scratch area in decreasing address order. When these two meet, an out of memory exception is raised (and the interpreter fully checks this).

Program memory

Immediately following the scratch area is the program text; it's a single object and never moves from its fixed place, although it does change in size. When the program size changes, because a user has edited the program, all variable values are lost as the program could overwrite objects held in free memory (the *heap*) as described later.

Heap

Following the program area is the *heap*, an area of free memory which is used to hold strings and arrays which are dynamically sized. These objects are created when required and allocated from free memory.

Dynamic objects are allocated from the heap using a first-fit algorithm. The free list head is kept in **nb_first_free_index** which is an index into the cell array. Each entry in the free list has type NB_TYPE_FREE and its length in cells is kept in the member length of free.

Both strings and arrays can be intermediate objects created during the evaluation of an expression, used, and then become unreferenced because they're not used again. For instance, consider the statement PRINT "A" & "B". In this case, the strings "A" and "B" are created, concatenated, printed, and are then orphaned with no references to them, and in doing so use six cells in the heap. As more and more intermediate objects are created and the heap consumed, it's inevitable that at some stage we'll hit the end of the heap, and when we do we trigger a *garbage collection*.

Garbage collection

About garbage collection

Garbage collection is triggered when **nb_create_string** or **nb_create_array** are called (and only those two functions, either directly or indirectly) and there are not enough free cells to satisfy the request in any block in the free list. As we've seen, some strings or arrays held in the heap may be orphaned and considered *garbage*. In this case, a garbage collection is triggered in order to try to return some unused objects to the heap, and this is done in two phases.

Phase 1, Mark and sweep.

The first phase is to mark all accessible objects in the heap and then reconstruct the free list from all unreferenced (and therefore free) objects. Adjacent free objects are coalesced into single blocks. Once the free list is reconstructed, another attempt is made to see whether there is a block that can satisfy the request. If there is, that is all well and good, and the block of cells is then removed from the free list and put to use. This phase is particularly efficient and doesn't require any costly movement of objects in memory.

Phase 2, Compact.

Following mark and sweep, if there is no single free block large enough to satisfy the request, the garbage collector tries even harder by *compacting* the used cells. In the second phase, all *used* blocks are known (from the first phase) and the garbage collector slides them to the beginning of object memory adjusting all references to moved blocks as it does so. In this way it creates a contiguous run of cells beyond the last used cell. These unused cells are now returned to the free list as a single block and are then used to allocate the new object (or fail if there is still not enough room for the object). Thus, the CoreBASIC garbage collector is a *compacting* garbage collector.

Of course, there is a down side to this. For a start, garbage collection takes time, so your program is halted whilst garbage is collected, but this isn't a big problem as the first phase of collection is very fast. Only when there is severe pressure on object memory with many live references to objects will the second compacting phase be triggered, and this is slightly slower.

The biggest down side of the compacting algorithm is that you need to take special care when creating an array or reference that all arrays and strings you want to keep around are rooted. The best place to keep objects alive so they're not collected is to push a reference to them onto the expression stack.

In a similar vein, it's imperative that you *do not* dereference an array or string and keep that pointer locally if there is a possibility that a garbage collection will happen before you are done with the object. This is because the garbage collector may move objects in memory, and so the string or array elements will move in memory also. If you need to refer to an object where there is the possibility of a garbage collection, you must use the reference and dereference when you need to look at the string contents or array elements.

Some places in the interpreter are marked with a comment "*GC*" which indicates that the code following is written in a specific way so that it doesn't get tangled up with the garbage collector.

CrossWorks CoreBASIC Library

<nb_core.h>

API Summary

***** UNASSIGNED GROUP *****	
nb_array_data	Returns a pointer to the array elements for a pointer to an object
nb_clear_flags	Clear interpreter flags
nb_error_line_number	The current line number that had an error in it.
nb_matrix_data	Returns a pointer to the array elements for a pointer to an object
nb_string_data	Returns a pointer to the string for a pointer to an object
nb_token_length_inline	Returns the number of bytes occupied by the token at
Syntax	
NB_CORE_TOKENS	Single-byte CORE module tokens
NB_PROPERTIES	Property enumeration
nb_check_optional_token	Checks an optional token at the token pointer
nb_check_optional_token_2	Checks an optional token at the token pointer
nb_is_end_of_line	Return whether X is logically the end of the line
nb_is_end_of_statement	Does token mark an end of statement?
Program	
NB_LINE_HEADER_t	Line header
nb_line_t	The structure of a CoreBASIC line
nb_next_line	Get pointer to next line
nb_program_address	Construct token pointer from program offset
nb_program_offset	Construct program offset from token pointer
Utility	
nb_cell_index	Return the index into the object array of the cell pointer X
nb_dyadic_index	Convert token to dyadic index
nb_is_zero	Check-zero predicate
nb_monadic_index	Converts token into monadic index
nb_signon	Display CoreBASIC sign on
nb_truth_value	Turns a C truth value into a CoreBASIC truth value
Modules	

CrossWorks CoreBASIC Library

NB_CORE_MODULE_INDEX	The index of the core module in the module array
nb_core_module	The core module
nb_module_t	CoreBASIC extension module definition
nb_modules	The list of CoreBASIC modules
Configuration	
NB_INPUT_BUFFER_SIZE	Maximum number of characters in CoreBASIC input buffer
NB_USE_OWN_ACOSH	Choose own implementation of acosh
NB_USE_OWN_ASINH	Choose own implementation of asinh
NB_USE_OWN_ATANH	Choose own implementation of atanh
nb_input_buffer	The CoreBASIC scratch input buffer
Memory	
NB_SCRATCH_CELLS	The number of objects held in scratch memory
nb_cell_index_t	Cell index within CoreBASIC memory array
nb_cell_type_t	Classifies an object held on the stack or in memory
nb_execute_quick_recycle	Recycle unused memory
nb_execute_recycle	Recycle unused memory
nb_sentinel_index	The sentinel cell index
Exceptions	
NB_TRY_CONTEXT_t	Exception context
nb_current_ctx	Topmost try context
nb_try	Create a try block
nb_unwind_try	Unwind the topmost try
Control	
nb_control_stack_item_t	Control stack entry
Variables	
nb_sp	Expression stack pointer
nb_var_front_index	Variable front index
State	
nb_check_immediate_mode	Ensure CoreBASIC in immediate mode
nb_check_program_mode	Ensure CoreBASIC in program mode
nb_immediate_mode	Interpreter mode
Core	
NB_ERROR_t	CoreBASIC runtime errors
nb_boolean_t	A Boolean value used by the interpreter

CrossWorks CoreBASIC Library

nb_flag_t	Interpreter action flags
nb_flags	Interpreter flags
Tokens	
NB_TOKEN_t	CORE module token enumeration
nb_property_t	Global properties
nb_token_auto	Configurable AUTO token
nb_token_catalog	Configurable CATALOG token
nb_token_cd	Configurable CD token
nb_token_chain	Configurable CHAIN token
nb_token_chdir	Configurable CHDIR token
nb_token_check	Configurable CHECK token
nb_token_dir	Configurable DIR token
nb_token_example	Configurable EXAMPLE token
nb_token_flush	Configurable FLUSH token
nb_token_get	Configurable GET token
nb_token_history	Configurable HISTORY token
nb_token_kill	Configurable KILL token
nb_token_load	Configurable LOAD token
nb_token_reboot	Configurable REBOOT token
nb_token_save	Configurable SAVE token
nb_tokenize	Tokenize text
Events	
nb_broadcast_event	Broadcast event to all modules
nb_event_t	CoreBASIC event notification
Checks	
nb_throw_exception	Throw a CoreBASIC exception
I/O	
nb_print_line	Print a tokenized line
nb_print_tokens	Print a sequence of tokens
Runtime	
nb_check_stack	Check available stack space
Stack	
nb_push_integer	Push integer to expression stack
nb_replace_float	Replace top of stack with integer
nb_replace_integer	Replace the top item on the stack with an integer

CrossWorks CoreBASIC Library

Execute	
nb_assign_variable	Assign value to a variable
Evaluation	
nb_stringize	Convert top of stack to string
Editing	
nb_delete_line	Delete program line
nb_delete_line_number	Delete numbered line
nb_delete_token	Delete single token in line
nb_find_line	Finds numbered line
nb_first_line	Return first line in program
nb_fix_program_object	Fix up program object after editing
nb_join_lines	Join lines
nb_line_number_referenced	Is line referenced?
nb_program_end	Return end of program
nb_program_size	Program size in bytes
nb_replace_line	Replace program line

CrossWorks CoreBASIC Library

NB_CORE_MODULE_INDEX

Synopsis

```
#define NB_CORE_MODULE_INDEX 0
```

Must be zero.

NB_CORE_TOKENS

Synopsis

```
#define NB_CORE_TOKENS \
   NB_TOKEN("?<eol>",        NB_TOKEN_EOL),          \
   NB_TOKEN("?<pad>",        NB_TOKEN_PAD),          \
   NB_TOKEN("'",             NB_TOKEN_APOSTROPHE),   \
   NB_TOKEN(":",             NB_TOKEN_COLON),        \
   NB_TOKEN("ELSE",          NB_TOKEN_LINE_ELSE),    \
   NB_TOKEN("LIST",          NB_TOKEN_LIST),         \
   NB_TOKEN("PRINT",         NB_TOKEN_PRINT),        \
   NB_TOKEN("WRITE",         NB_TOKEN_WRITE),        \
   NB_TOKEN("GOTO",          NB_TOKEN_GOTO),         \
   NB_TOKEN("GOSUB",         NB_TOKEN_GOSUB),        \
   NB_TOKEN("RETURN",        NB_TOKEN_RETURN),       \
   NB_TOKEN("IF",            NB_TOKEN_LINE_IF),      \
   NB_TOKEN("IF",            NB_TOKEN_BLOCK_IF),     \
   NB_TOKEN("FOR EACH",      NB_TOKEN_FOR_EACH),     \
   NB_TOKEN("FOR",           NB_TOKEN_FOR),          \
   NB_TOKEN("WHILE",         NB_TOKEN_WHILE),        \
   NB_TOKEN("REPEAT",        NB_TOKEN_REPEAT),       \
   NB_TOKEN("CASE",          NB_TOKEN_CASE),         \
   NB_TOKEN("WHEN",          NB_TOKEN_WHEN),         \
   NB_TOKEN("OTHERWISE",     NB_TOKEN_OTHERWISE),    \
   NB_TOKEN("ENDCASE",       NB_TOKEN_ENDCASE),      \
   NB_TOKEN("DEFPROC",       NB_TOKEN_DEFPROC),      \
   NB_TOKEN("DEFFN",         NB_TOKEN_DEFFN),        \
   NB_TOKEN("NEXT",          NB_TOKEN_NEXT),         \
   NB_TOKEN("WEND",          NB_TOKEN_WEND),         \
   NB_TOKEN("UNTIL",         NB_TOKEN_UNTIL),        \
   NB_TOKEN("ENDPROC",       NB_TOKEN_ENDPROC),      \
   NB_TOKEN("ENDFN",         NB_TOKEN_ENDFN),        \
   NB_TOKEN("EXIT FOR",      NB_TOKEN_EXIT_FOR),     \
   NB_TOKEN("EXIT WHILE",    NB_TOKEN_EXIT_WHILE),   \
   NB_TOKEN("EXIT REPEAT",   NB_TOKEN_EXIT_REPEAT),  \
   NB_TOKEN("EXIT",          NB_TOKEN_EXIT),         \
   NB_TOKEN("END",           NB_TOKEN_END),          \
   NB_TOKEN("ENDIF",         NB_TOKEN_ENDIF),        \
   NB_TOKEN("ELSE",          NB_TOKEN_BLOCK_ELSE),   \
   NB_TOKEN("TRY",           NB_TOKEN_TRY),          \
   NB_TOKEN("THROW",         NB_TOKEN_THROW),        \
   NB_TOKEN("STOP",          NB_TOKEN_STOP),         \
   NB_TOKEN("CALL",          NB_TOKEN_CALL),         \
   NB_TOKEN("PROC",          NB_TOKEN_PROC),         \
   NB_TOKEN("REM",           NB_TOKEN_REM),          \
   NB_TOKEN("NEW",           NB_TOKEN_NEW),          \
   NB_TOKEN("LET",           NB_TOKEN_LET),          \
   NB_TOKEN("RUN",           NB_TOKEN_RUN),          \
   NB_TOKEN("MAT",           NB_TOKEN_MAT),          \
   NB_TOKEN("CORE",          NB_TOKEN_CORE),         \
   NB_TOKEN("INSTALL",       NB_TOKEN_INSTALL),      \
   NB_TOKEN("REPORT",        NB_TOKEN_REPORT),       \
   NB_TOKEN("ERROR",         NB_TOKEN_ERROR),        \
   NB_TOKEN("RECYCLE",       NB_TOKEN_RECYCLE),      \
   NB_TOKEN("CLASSIC",       NB_TOKEN_CLASSIC),      \
   NB_TOKEN("READ",          NB_TOKEN_READ),         \
   NB_TOKEN("DATA",          NB_TOKEN_DATA),         \
   NB_TOKEN("RESTORE",       NB_TOKEN_RESTORE),      \
   NB_TOKEN("DUMP",          NB_TOKEN_DUMP),         \
```

```
    NB_TOKEN("RANDOMIZE",     NB_TOKEN_RANDOMIZE),    \
    NB_TOKEN("SORT",          NB_TOKEN_SORT),         \
    NB_TOKEN("REVERSE",       NB_TOKEN_REVERSE),      \
    NB_TOKEN("SHUFFLE",       NB_TOKEN_SHUFFLE),      \
    NB_TOKEN("CLS",           NB_TOKEN_CLS),          \
    NB_TOKEN("VDU",           NB_TOKEN_VDU),          \
    NB_TOKEN("VERSION",       NB_TOKEN_VERSION),      \
    NB_TOKEN("DIM",           NB_TOKEN_DIM),          \
    NB_TOKEN("PAUSE",         NB_TOKEN_PAUSE),        \
    NB_TOKEN("WAIT",          NB_TOKEN_WAIT),         \
    NB_TOKEN("SYSTEM",        NB_TOKEN_SYSTEM),       \
    NB_TOKEN("TIMER",         NB_TOKEN_TIMER),        \
    NB_TOKEN("USING",         NB_TOKEN_USING),        \
    NB_TOKEN("INPUT",         NB_TOKEN_INPUT),        \
    NB_TOKEN("OUTPUT",        NB_TOKEN_OUTPUT),       \
    NB_TOKEN("ANALOG",        NB_TOKEN_ANALOG),       \
    NB_TOKEN("DIGITAL",       NB_TOKEN_DIGITAL),      \
    NB_TOKEN("?<ext_0>",      NB_TOKEN_EXTENSION_0),  \
    NB_TOKEN("?<ext_1>",      NB_TOKEN_EXTENSION_1),  \
    NB_TOKEN("?<ext_2>",      NB_TOKEN_EXTENSION_2),  \
    NB_TOKEN("?<ext_3>",      NB_TOKEN_EXTENSION_3),  \
    NB_TOKEN("?<ext_4>",      NB_TOKEN_EXTENSION_4),  \
    NB_TOKEN("?<ext_5>",      NB_TOKEN_EXTENSION_5),  \
    NB_TOKEN("?<ext_6>",      NB_TOKEN_EXTENSION_6),  \
    NB_TOKEN("?<ext_7>",      NB_TOKEN_EXTENSION_7),  \
    NB_TOKEN("?<ext_8>",      NB_TOKEN_EXTENSION_8),  \
    NB_TOKEN("?<ext_9>",      NB_TOKEN_EXTENSION_9),  \
    NB_TOKEN("?<ext_10>",     NB_TOKEN_EXTENSION_10), \
    NB_TOKEN("?<ext_11>",     NB_TOKEN_EXTENSION_11), \
    NB_TOKEN("?<ext_12>",     NB_TOKEN_EXTENSION_12), \
    NB_TOKEN("?<ext_13>",     NB_TOKEN_EXTENSION_13), \
    NB_TOKEN("?<ext_14>",     NB_TOKEN_EXTENSION_14), \
    NB_TOKEN("?<ext_15>",     NB_TOKEN_EXTENSION_15), \
    NB_TOKEN("?<var>",        NB_TOKEN_VAR),          \
    NB_TOKEN("?<property>",   NB_TOKEN_PROPERTY),     \
    NB_TOKEN("=",             NB_TOKEN_EQ),           \
    NB_TOKEN("<>",            NB_TOKEN_NE),           \
    NB_TOKEN("<=",            NB_TOKEN_LE),           \
    NB_TOKEN(">=",            NB_TOKEN_GE),           \
    NB_TOKEN("<",             NB_TOKEN_LT),           \
    NB_TOKEN(">",             NB_TOKEN_GT),           \
    NB_TOKEN("^",             NB_TOKEN_POW),          \
    NB_TOKEN("+",             NB_TOKEN_PLUS),         \
    NB_TOKEN("-",             NB_TOKEN_MINUS),        \
    NB_TOKEN("*",             NB_TOKEN_STAR),         \
    NB_TOKEN("/",             NB_TOKEN_SLASH),        \
    NB_TOKEN("\\",            NB_TOKEN_BACKSLASH),    \
    NB_TOKEN("MOD",           NB_TOKEN_MOD),          \
    NB_TOKEN("&",             NB_TOKEN_AMP),          \
    NB_TOKEN("AND",           NB_TOKEN_AND),          \
    NB_TOKEN("OR",            NB_TOKEN_OR),           \
    NB_TOKEN("XOR",           NB_TOKEN_XOR),          \
    NB_TOKEN("EQV",           NB_TOKEN_EQV),          \
    NB_TOKEN("IMP",           NB_TOKEN_IMP),          \
    NB_TOKEN("MAX",           NB_TOKEN_MAX),          \
    NB_TOKEN("MIN",           NB_TOKEN_MIN),          \
    NB_TOKEN("AND THEN",      NB_TOKEN_AND_THEN),     \
    NB_TOKEN("OR ELSE",       NB_TOKEN_OR_ELSE),      \
    NB_TOKEN("-",             NB_TOKEN_NEGATE),       \
    NB_TOKEN("CNJ",           NB_TOKEN_CNJ),          \
    NB_TOKEN("ARG",           NB_TOKEN_ARG),          \
    NB_TOKEN("RE",            NB_TOKEN_RE),           \
```

CrossWorks CoreBASIC Library

```
    NB_TOKEN("IM",           NB_TOKEN_IM),          \
    NB_TOKEN("NOT",          NB_TOKEN_NOT),         \
    NB_TOKEN("SGN",          NB_TOKEN_SGN),         \
    NB_TOKEN("ABS",          NB_TOKEN_ABS),         \
    NB_TOKEN("INV",          NB_TOKEN_INV),         \
    NB_TOKEN("RND",          NB_TOKEN_RND),         \
    NB_TOKEN("SQR",          NB_TOKEN_SQR),         \
    NB_TOKEN("EXP",          NB_TOKEN_EXP),         \
    NB_TOKEN("LOG",          NB_TOKEN_LOG),         \
    NB_TOKEN("LOG2",         NB_TOKEN_LOG2),        \
    NB_TOKEN("LOG10",        NB_TOKEN_LOG10),       \
    NB_TOKEN("FIX",          NB_TOKEN_FIX),         \
    NB_TOKEN("FLT",          NB_TOKEN_FLT),         \
    NB_TOKEN("INT",          NB_TOKEN_INT),         \
    NB_TOKEN("CINT",         NB_TOKEN_CINT),        \
    NB_TOKEN("STR",          NB_TOKEN_STR),         \
    NB_TOKEN("VAL",          NB_TOKEN_VAL),         \
    NB_TOKEN("INF",          NB_TOKEN_INF),         \
    NB_TOKEN("NAN",          NB_TOKEN_NAN),         \
    NB_TOKEN("LEN",          NB_TOKEN_LEN),         \
    NB_TOKEN("HIGH",         NB_TOKEN_HIGH),        \
    NB_TOKEN("TYP",          NB_TOKEN_TYP),         \
    NB_TOKEN("ZER",          NB_TOKEN_ZER),         \
    NB_TOKEN("IDN",          NB_TOKEN_IDN),         \
    NB_TOKEN("CON",          NB_TOKEN_CON),         \
    NB_TOKEN("COL",          NB_TOKEN_COL),         \
    NB_TOKEN("ROW",          NB_TOKEN_ROW),         \
    NB_TOKEN("TRN",          NB_TOKEN_TRN),         \
    NB_TOKEN("DET",          NB_TOKEN_DET),         \
    NB_TOKEN("GEN",          NB_TOKEN_GEN),         \
    NB_TOKEN("SAMPLE",       NB_TOKEN_SAMPLE),      \
    NB_TOKEN("DEFER",        NB_TOKEN_DEFER),       \
    NB_TOKEN("EVAL",         NB_TOKEN_EVAL),        \
    NB_TOKEN("SUM",          NB_TOKEN_SUM),         \
    NB_TOKEN("PI",           NB_TOKEN_PI),          \
    NB_TOKEN("CMPLX",        NB_TOKEN_CMPLX),       \
    NB_TOKEN("QUAT",         NB_TOKEN_QUAT),        \
    NB_TOKEN("MERGE",        NB_TOKEN_MERGE),       \
    NB_TOKEN("CLONE",        NB_TOKEN_CLONE),       \
    NB_TOKEN("?<zero>",      NB_TOKEN_ZERO),        \
    NB_TOKEN("?<int_lit_2>", NB_TOKEN_INT_LIT_2),   \
    NB_TOKEN("?<int_lit_x>", NB_TOKEN_INT_LIT_X),   \
    NB_TOKEN("?<hex_lit_2>", NB_TOKEN_HEX_LIT_2),   \
    NB_TOKEN("?<hex_lit_x>", NB_TOKEN_HEX_LIT_X),   \
    NB_TOKEN("?<flt_lit_2>", NB_TOKEN_FLT_LIT_2),   \
    NB_TOKEN("?<flt_lit_x>", NB_TOKEN_FLT_LIT_X),   \
    NB_TOKEN("?<str_lit>",   NB_TOKEN_STR_LIT),     \
    NB_TOKEN("LEFT",         NB_TOKEN_LEFT),        \
    NB_TOKEN("MID",          NB_TOKEN_MID),         \
    NB_TOKEN("RIGHT",        NB_TOKEN_RIGHT),       \
    NB_TOKEN("INNER",        NB_TOKEN_INNER),       \
    NB_TOKEN("REDUCE",       NB_TOKEN_REDUCE),      \
    NB_TOKEN("SELECT",       NB_TOKEN_SELECT),      \
    NB_TOKEN("RAVEL",        NB_TOKEN_RAVEL),       \
    NB_TOKEN("PICK",         NB_TOKEN_PICK),        \
    NB_TOKEN("IFF",          NB_TOKEN_IFF),         \
    NB_TOKEN(".",            NB_TOKEN_PERIOD),      \
    NB_TOKEN(";",            NB_TOKEN_SEMI),        \
    NB_TOKEN(",",            NB_TOKEN_COMMA),       \
    NB_TOKEN("|",            NB_TOKEN_BAR),         \
    NB_TOKEN("#",            NB_TOKEN_SHARP),       \
    NB_TOKEN("@",            NB_TOKEN_AT),          \
```

CrossWorks CoreBASIC Library

```
    NB_TOKEN("(",            NB_TOKEN_LPAR),         \
    NB_TOKEN(")",            NB_TOKEN_RPAR),         \
    NB_TOKEN("[",            NB_TOKEN_LBRK),         \
    NB_TOKEN("]",            NB_TOKEN_RBRK),         \
    NB_TOKEN("TO",           NB_TOKEN_TO),           \
    NB_TOKEN("IN",           NB_TOKEN_IN),           \
    NB_TOKEN("ON",           NB_TOKEN_ON),           \
    NB_TOKEN("OFF",          NB_TOKEN_OFF),          \
    NB_TOKEN("THEN",         NB_TOKEN_THEN),         \
    NB_TOKEN("STEP",         NB_TOKEN_STEP),         \
    NB_TOKEN("AS",           NB_TOKEN_AS),           \
    NB_TOKEN("MODULES",      NB_TOKEN_MODULES),      \
    NB_TOKEN("?<lnref>",     NB_TOKEN_LNREF),        \
    NB_TOKEN("?<escape>",    NB_TOKEN_ESCAPE)
```

CrossWorks CoreBASIC Library

NB_ERROR_t

Synopsis

```
typedef enum {
  NB_NO_ERROR,
  NB_PROGRAM_ENDED,
  NB_OUT_OF_MEMORY,
  NB_NO_SUCH_LINE,
  NB_SYNTAX_ERROR,
  NB_BREAKPOINT,
  NB_WATCHDOG,
  NB_IMMEDIATE_MODE_ONLY,
  NB_PROGRAM_MODE_ONLY,
  NB_ARGUMENT_ERROR,
  NB_TYPE_MISMATCH,
  NB_UNIMPLEMENTED,
  NB_DIMENSION_ERROR,
  NB_SUBSCRIPT_ERROR,
  NB_BAD_LINE_NUMBER,
  NB_STRING_TOO_LONG,
  NB_EXPRESSION_TOO_COMPLEX,
  NB_NEXT_WITHOUT_FOR,
  NB_WEND_WITHOUT_WHILE,
  NB_UNTIL_WITHOUT_REPEAT,
  NB_ENDPROC_WITHOUT_DEFPROC,
  NB_ENDFN_WITHOUT_DEFFN,
  NB_FOR_EACH_WITHOUT_NEXT,
  NB_FOR_WITHOUT_NEXT,
  NB_WHILE_WITHOUT_WEND,
  NB_REPEAT_WITHOUT_UNTIL,
  NB_DEFPROC_WITHOUT_ENDPROC,
  NB_DEFFN_WITHOUT_ENDFN,
  NB_RETURN_WITHOUT_GOSUB,
  NB_CANT_CONTINUE,
  NB_GRAPHICS_ERROR,
  NB_DRIVER_NOT_FOUND,
  NB_DEFPROC_CANNOT_NEST,
  NB_FELL_INTO_DEFPROC,
  NB_FONT_NOT_FOUND,
  NB_PROC_NOT_DEFINED,
  NB_DUPLICATE_PROC,
  NB_ENDPROC_WITHOUT_CALL,
  NB_TOO_MANY_PARAMETERS,
  NB_NOT_ENOUGH_PARAMETERS,
  NB_TOO_MANY_DRIVERS,
  NB_TOO_MANY_STREAMS,
  NB_BAD_FILENAME,
  NB_BAD_PIN_CONFIGURATION,
  NB_PROGRAM_NOT_FOUND,
  NB_OUT_OF_DATA,
  NB_UNSUPPORTED_PROPERTY,
  NB_USING_ERROR,
  NB_EXPECTED_COMMA,
  NB_EXPECTED_SEMI,
  NB_EXPECTED_EQ,
  NB_EXPECTED_LPAR,
  NB_EXPECTED_RPAR,
  NB_EXPECTED_RBRK,
  NB_EXPECTED_TO,
```

```
  NB_EXPECTED_THEN,
  NB_EXPECTED_VAR,
  NB_WHEN_WITHOUT_CASE,
  NB_OTHERWISE_WITHOUT_CASE,
  NB_ENDCASE_WITHOUT_CASE,
  NB_ELSE_WITHOUT_IF,
  NB_IF_WITHOUT_ENDIF,
  NB_CASE_WITHOUT_ENDCASE,
  NB_PAGE_DOES_NOT_EXIST,
  NB_BAD_INPUT,
  NB_BAD_STATEMENT_AFTER_TRY,
  NB_FELL_INTO_ENDFN,
  NB_TOO_MANY_VARIABLES,
  NB_PIN_CONFLICT,
  NB_EXIT_FOR_WITHOUT_FOR,
  NB_EXIT_WHILE_WITHOUT_WHILE,
  NB_EXIT_REPEAT_WITHOUT_REPEAT,
  NB_INTERNAL_ERROR
} NB_ERROR_t;
```

Description

Although the ordering of these error messages is essentially arbitrary, it is important to keep the WHILE, REPEAT, and FOR errors together in a group along with the tokens that these relate to because **nb_check_structure** expects the token numbering and error messages to have this structure.

NB_NO_ERROR

Same as CTL_NO_ERROR and has the value zero.

NB_PROGRAM_ENDED

Internal code indicating the CoreBASIC program terminated at END or for some other reason. This is silently turned into NB_NO_ERROR when caught.

NB_OUT_OF_MEMORY

Interpreter ran out of memory.

NB_NO_SUCH_LINE

Line doesn't exist in GOTO, GOSUB, or RESTORE.

NB_SYNTAX_ERROR

Generic syntax error.

NB_BREAKPOINT

User interrupted execution or program executes STOP.

NB_WATCHDOG

Watchdog fired when debugging.

NB_IMMEDIATE_MODE_ONLY

Command is available in immediate mode only.

NB_PROGRAM_MODE_ONLY

Command is available in program mode only.

NB_ARGUMENT_ERROR

Argument to standard function is in error.

NB_TYPE_MISMATCH

Wrong type in general.

NB_UNIMPLEMENTED

I haven't got there yet.

NB_DIMENSION_ERROR

Wrong shape of array.

NB_SUBSCRIPT_ERROR

Subscript out of range.

NB_BAD_LINE_NUMBER

Line 0 is not a valid line number.

NB_STRING_TOO_LONG

String is longer than maximum length allowed.

NB_EXPRESSION_TOO_COMPLEX

Too many items on the expression stack.

NB_NEXT_WITHOUT_FOR

NEXT has no matching FOR (preflight check).

NB_WEND_WITHOUT_WHILE

WEND has no matching WHILE (preflight check).

NB_UNTIL_WITHOUT_REPEAT

UNTIL has no matching REPEAT (preflight check).

NB_ENDPROC_WITHOUT_DEFPROC

ENDPROC has no matching DEFPROC (preflight check).

NB_ENDFN_WITHOUT_DEFFN

ENDFN has no matching DEFFN (preflight check).

NB_FOR_EACH_WITHOUT_NEXT

FOR EACH has no matching NEXT (preflight check).

NB_FOR_WITHOUT_NEXT

FOR has no matching NEXT (preflight check).

NB_WHILE_WITHOUT_WEND

WHILE has no matching WEND (preflight check).

NB_REPEAT_WITHOUT_UNTIL

REPEAT has no matching UNTIL (preflight check).

CrossWorks CoreBASIC Library

NB_DEFPROC_WITHOUT_ENDPROC

DEFPROC has no matching ENDPROC (preflight check).

NB_DEFFN_WITHOUT_ENDFN

DEFFN has no matching ENDFN (preflight check).

NB_RETURN_WITHOUT_GOSUB

RETURN has no matching GOSUB (runtime check).

CrossWorks CoreBASIC Library

NB_INPUT_BUFFER_SIZE

Synopsis

```
#define NB_INPUT_BUFFER_SIZE 256
```

See Also

nb_input_buffer.

CrossWorks CoreBASIC Library

NB_LINE_HEADER_t

Synopsis

```
typedef struct {
  unsigned short next_offset;
  unsigned short line_number;
} NB_LINE_HEADER_t;
```

Description

NB_LINE_HEADER_t is the structure of a line header which comes immediately before the tokens comprising the line.

`next_offset`

Self-relative byte offset to next line. NOTE: could shorten this to an unsigned char, limit lines to 255 bytes in total, and then use the extra byte for a set of flags (e.g. breakpoint flags...)

`line_number`

The CoreBASIC line number.

CrossWorks CoreBASIC Library

NB_PROPERTIES

Synopsis

```
#define NB_PROPERTIES \
   NB_TOKEN("NULL",          NB_PROPERTY_NULL),          \
   NB_TOKEN("D0",            NB_PROPERTY_D0),            \
   NB_TOKEN("D1",            NB_PROPERTY_D1),            \
   NB_TOKEN("D2",            NB_PROPERTY_D2),            \
   NB_TOKEN("D3",            NB_PROPERTY_D3),            \
   NB_TOKEN("D4",            NB_PROPERTY_D4),            \
   NB_TOKEN("D5",            NB_PROPERTY_D5),            \
   NB_TOKEN("D6",            NB_PROPERTY_D6),            \
   NB_TOKEN("D7",            NB_PROPERTY_D7),            \
   NB_TOKEN("D8",            NB_PROPERTY_D8),            \
   NB_TOKEN("D9",            NB_PROPERTY_D9),            \
   NB_TOKEN("D10",           NB_PROPERTY_D10),           \
   NB_TOKEN("D11",           NB_PROPERTY_D11),           \
   NB_TOKEN("D12",           NB_PROPERTY_D12),           \
   NB_TOKEN("D13",           NB_PROPERTY_D13),           \
   NB_TOKEN("D14",           NB_PROPERTY_D14),           \
   NB_TOKEN("D15",           NB_PROPERTY_D15),           \
   NB_TOKEN("D16",           NB_PROPERTY_D16),           \
   NB_TOKEN("D17",           NB_PROPERTY_D17),           \
   NB_TOKEN("D18",           NB_PROPERTY_D18),           \
   NB_TOKEN("D19",           NB_PROPERTY_D19),           \
   NB_TOKEN("A0",            NB_PROPERTY_A0),            \
   NB_TOKEN("A1",            NB_PROPERTY_A1),            \
   NB_TOKEN("A2",            NB_PROPERTY_A2),            \
   NB_TOKEN("A3",            NB_PROPERTY_A3),            \
   NB_TOKEN("A4",            NB_PROPERTY_A4),            \
   NB_TOKEN("A5",            NB_PROPERTY_A5),            \
   NB_TOKEN("A6",            NB_PROPERTY_A6),            \
   NB_TOKEN("A7",            NB_PROPERTY_A7),            \
   NB_TOKEN("A8",            NB_PROPERTY_A8),            \
   NB_TOKEN("A9",            NB_PROPERTY_A9),            \
   NB_TOKEN("A10",           NB_PROPERTY_A10),           \
   NB_TOKEN("A11",           NB_PROPERTY_A11),           \
   NB_TOKEN("A12",           NB_PROPERTY_A12),           \
   NB_TOKEN("A13",           NB_PROPERTY_A13),           \
   NB_TOKEN("A14",           NB_PROPERTY_A14),           \
   NB_TOKEN("A15",           NB_PROPERTY_A15),           \
   NB_TOKEN("A16",           NB_PROPERTY_A16),           \
   NB_TOKEN("A17",           NB_PROPERTY_A17),           \
   NB_TOKEN("A18",           NB_PROPERTY_A18),           \
   NB_TOKEN("A19",           NB_PROPERTY_A19),           \
   NB_TOKEN("A",             NB_PROPERTY_A),             \
   NB_TOKEN("B",             NB_PROPERTY_B),             \
   NB_TOKEN("C",             NB_PROPERTY_C),             \
   NB_TOKEN("D",             NB_PROPERTY_D),             \
   NB_TOKEN("E",             NB_PROPERTY_E),             \
   NB_TOKEN("F",             NB_PROPERTY_F),             \
   NB_TOKEN("G",             NB_PROPERTY_G),             \
   NB_TOKEN("H",             NB_PROPERTY_H),             \
   NB_TOKEN("I",             NB_PROPERTY_I),             \
   NB_TOKEN("J",             NB_PROPERTY_J),             \
   NB_TOKEN("K",             NB_PROPERTY_K),             \
   NB_TOKEN("L",             NB_PROPERTY_L),             \
   NB_TOKEN("M",             NB_PROPERTY_M),             \
   NB_TOKEN("Q",             NB_PROPERTY_Q),             \
```

CrossWorks CoreBASIC Library

```
    NB_TOKEN("R",             NB_PROPERTY_R),             \
    NB_TOKEN("V",             NB_PROPERTY_V),             \
    NB_TOKEN("V",             NB_PROPERTY_W),             \
    NB_TOKEN("X",             NB_PROPERTY_X),             \
    NB_TOKEN("Y",             NB_PROPERTY_Y),             \
    NB_TOKEN("Z",             NB_PROPERTY_Z),             \
    NB_TOKEN("INTERFACES",    NB_PROPERTY_INTERFACES),    \
    NB_TOKEN("DEFAULT",       NB_PROPERTY_DEFAULT),       \
    NB_TOKEN("LEFT",          NB_PROPERTY_LEFT),          \
    NB_TOKEN("RIGHT",         NB_PROPERTY_RIGHT),         \
    NB_TOKEN("MOTOR",         NB_PROPERTY_MOTOR),         \
    NB_TOKEN("UP",            NB_PROPERTY_UP),            \
    NB_TOKEN("DOWN",          NB_PROPERTY_DOWN),          \
    NB_TOKEN("ON",            NB_PROPERTY_ON),            \
    NB_TOKEN("OFF",           NB_PROPERTY_OFF),           \
    NB_TOKEN("POLARITY",      NB_PROPERTY_POLARITY),      \
    NB_TOKEN("HOME",          NB_PROPERTY_HOME),          \
    NB_TOKEN("START",         NB_PROPERTY_START),         \
    NB_TOKEN("SINGLE",        NB_PROPERTY_SINGLE),        \
    NB_TOKEN("SELECT",        NB_PROPERTY_SELECT),        \
    NB_TOKEN("ORIGIN",        NB_PROPERTY_ORIGIN),        \
    NB_TOKEN("PRESS",         NB_PROPERTY_PRESS),         \
    NB_TOKEN("KEYS",          NB_PROPERTY_KEYS),          \
    NB_TOKEN("BUTTON",        NB_PROPERTY_BUTTON),        \
    NB_TOKEN("SOURCE",        NB_PROPERTY_SOURCE),        \
    NB_TOKEN("IMAGE",         NB_PROPERTY_IMAGE),         \
    NB_TOKEN("SCREEN",        NB_PROPERTY_SCREEN),        \
    NB_TOKEN("CHARACTER",     NB_PROPERTY_CHARACTER),     \
    NB_TOKEN("CONTROL",       NB_PROPERTY_CONTROL),       \
    NB_TOKEN("ROTATION",      NB_PROPERTY_ROTATION),      \
    NB_TOKEN("PALETTE",       NB_PROPERTY_PALETTE),       \
    NB_TOKEN("BIAS",          NB_PROPERTY_BIAS),          \
    NB_TOKEN("GAIN",          NB_PROPERTY_GAIN),          \
    NB_TOKEN("AMPLIFIER",     NB_PROPERTY_AMPLIFIER),     \
    NB_TOKEN("CONTRAST",      NB_PROPERTY_CONTRAST),      \
    NB_TOKEN("SEEK",          NB_PROPERTY_SEEK),          \
    NB_TOKEN("COMMAND",       NB_PROPERTY_COMMAND),       \
    NB_TOKEN("RESPONSE",      NB_PROPERTY_RESPONSE),      \
    NB_TOKEN("MAP",           NB_PROPERTY_MAP),           \
    NB_TOKEN("ZL",            NB_PROPERTY_ZL),            \
    NB_TOKEN("ZR",            NB_PROPERTY_ZR),            \
    NB_TOKEN("OX",            NB_PROPERTY_OX),            \
    NB_TOKEN("OY",            NB_PROPERTY_OY),            \
    NB_TOKEN("ALL",           NB_PROPERTY_ALL),           \
    NB_TOKEN("TX",            NB_PROPERTY_TX),            \
    NB_TOKEN("RX",            NB_PROPERTY_RX),            \
    NB_TOKEN("READY",         NB_PROPERTY_READY),         \
    NB_TOKEN("TEST",          NB_PROPERTY_TEST),          \
    NB_TOKEN("ROW",           NB_PROPERTY_ROW),           \
    NB_TOKEN("COL",           NB_PROPERTY_COL),           \
    NB_TOKEN("LINE",          NB_PROPERTY_LINE),          \
    NB_TOKEN("CENTER",        NB_PROPERTY_CENTER),        \
    NB_TOKEN("CURSOR",        NB_PROPERTY_CURSOR),        \
    NB_TOKEN("WRITE",         NB_PROPERTY_WRITE),         \
    NB_TOKEN("PRINT",         NB_PROPERTY_PRINT),         \
    NB_TOKEN("OPTION",        NB_PROPERTY_OPTION),        \
    NB_TOKEN("ERROR",         NB_PROPERTY_ERROR),         \
    NB_TOKEN("AX",            NB_PROPERTY_AX),            \
    NB_TOKEN("AY",            NB_PROPERTY_AY),            \
    NB_TOKEN("AZ",            NB_PROPERTY_AZ),            \
    NB_TOKEN("GX",            NB_PROPERTY_GX),            \
    NB_TOKEN("GY",            NB_PROPERTY_GY),            \
```

```
    NB_TOKEN("GZ",            NB_PROPERTY_GZ),            \
    NB_TOKEN("MX",            NB_PROPERTY_MX),            \
    NB_TOKEN("MY",            NB_PROPERTY_MY),            \
    NB_TOKEN("MZ",            NB_PROPERTY_MZ),            \
    NB_TOKEN("LED",           NB_PROPERTY_LED),           \
    NB_TOKEN("RED",           NB_PROPERTY_RED),           /*could  get rid of this*/ \
    NB_TOKEN("GREEN",         NB_PROPERTY_GREEN),         /*could  get rid of this*/ \
    NB_TOKEN("SEGMENT",       NB_PROPERTY_SEGMENT),       \
    NB_TOKEN("DIGIT",         NB_PROPERTY_DIGIT),         \
    NB_TOKEN("WAVEFORM",      NB_PROPERTY_WAVEFORM),      \
    NB_TOKEN("FREQUENCY",     NB_PROPERTY_FREQUENCY),     \
    NB_TOKEN("PROPERTY",      NB_PROPERTY_PROPERTY),      \
    NB_TOKEN("VOLUME",        NB_PROPERTY_VOLUME),        \
    NB_TOKEN("MUTE",          NB_PROPERTY_MUTE),          \
    NB_TOKEN("VERSION",       NB_PROPERTY_VERSION),       \
    NB_TOKEN("ID",            NB_PROPERTY_ID),            \
    NB_TOKEN("FRAME",         NB_PROPERTY_FRAME),         \
    NB_TOKEN("BLANKING",      NB_PROPERTY_BLANKING),      \
    NB_TOKEN("PAGE",          NB_PROPERTY_PAGE),          \
    NB_TOKEN("POKE",          NB_PROPERTY_POKE),          \
    NB_TOKEN("PEEK",          NB_PROPERTY_PEEK),          \
    NB_TOKEN("BYTE",          NB_PROPERTY_BYTE),          \
    NB_TOKEN("HALF",          NB_PROPERTY_HALF),          \
    NB_TOKEN("WORD",          NB_PROPERTY_WORD),          \
    NB_TOKEN("EEPROM",        NB_PROPERTY_EEPROM),        \
    NB_TOKEN("REGISTER",      NB_PROPERTY_REGISTER),      \
    NB_TOKEN("STATUS",        NB_PROPERTY_STATUS),        \
    NB_TOKEN("DEVICE",        NB_PROPERTY_DEVICE),        \
    NB_TOKEN("STRING",        NB_PROPERTY_STRING),        \
    NB_TOKEN("CONFIGURATION", NB_PROPERTY_CONFIGURATION), \
    NB_TOKEN("LANGUAGE",      NB_PROPERTY_LANGUAGE),      \
    NB_TOKEN("ENDPOINT",      NB_PROPERTY_ENDPOINT),      \
    NB_TOKEN("LENGTH",        NB_PROPERTY_LENGTH),        \
    NB_TOKEN("UNLOCK",        NB_PROPERTY_UNLOCK),        \
    NB_TOKEN("CLASS",         NB_PROPERTY_CLASS),         \
    NB_TOKEN("RANGE",         NB_PROPERTY_RANGE),         \
    NB_TOKEN("BANDWIDTH",     NB_PROPERTY_BANDWIDTH),     \
    NB_TOKEN("RESOLUTION",    NB_PROPERTY_RESOLUTION),    \
    NB_TOKEN("HOUR",          NB_PROPERTY_HOUR),          \
    NB_TOKEN("MINUTE",        NB_PROPERTY_MINUTE),        \
    NB_TOKEN("SECOND",        NB_PROPERTY_SECOND),        \
    NB_TOKEN("DAY",           NB_PROPERTY_DAY),           \
    NB_TOKEN("DATE",          NB_PROPERTY_DATE),          \
    NB_TOKEN("MONTH",         NB_PROPERTY_MONTH),         \
    NB_TOKEN("TIME",          NB_PROPERTY_TIME),          \
    NB_TOKEN("YEAR",          NB_PROPERTY_YEAR),          \
    NB_TOKEN("IPADDR",        NB_PROPERTY_IPADDR),        \
    NB_TOKEN("NETMASK",       NB_PROPERTY_NETMASK),       \
    NB_TOKEN("GATEWAY",       NB_PROPERTY_GATEWAY),       \
    NB_TOKEN("DHCP",          NB_PROPERTY_DHCP),          \
    NB_TOKEN("SMTPSERVER",    NB_PROPERTY_SMTPSERVER),    \
    NB_TOKEN("TIMESERVER",    NB_PROPERTY_TIMESERVER),    \
    NB_TOKEN("CORESERVER",    NB_PROPERTY_CORESERVER),    \
    NB_TOKEN("PROXYSERVER",   NB_PROPERTY_PROXYSERVER),   \
    NB_TOKEN("PROXYPORT",     NB_PROPERTY_PROXYPORT),     \
    NB_TOKEN("NODE",          NB_PROPERTY_NODE),          \
    NB_TOKEN("PAN",           NB_PROPERTY_PAN),           \
    NB_TOKEN("PROMISCUOUS",   NB_PROPERTY_PROMISCUOUS),   \
    NB_TOKEN("MODE",          NB_PROPERTY_MODE),          \
    NB_TOKEN("DRIVE",         NB_PROPERTY_DRIVE),         \
    NB_TOKEN("NAME",          NB_PROPERTY_NAME),          \
    NB_TOKEN("MODEL",         NB_PROPERTY_MODEL),         \
```

```
    NB_TOKEN("SERIAL",         NB_PROPERTY_SERIAL),         \
    NB_TOKEN("DATA",           NB_PROPERTY_DATA),           \
    NB_TOKEN("STOP",           NB_PROPERTY_STOP),           \
    NB_TOKEN("PARITY",         NB_PROPERTY_PARITY),         \
    NB_TOKEN("SPEED",          NB_PROPERTY_SPEED),          \
    NB_TOKEN("WIDTH",          NB_PROPERTY_WIDTH),          \
    NB_TOKEN("IDLE",           NB_PROPERTY_IDLE),           \
    NB_TOKEN("BUS",            NB_PROPERTY_BUS),            \
    NB_TOKEN("HEIGHT",         NB_PROPERTY_HEIGHT),         \
    NB_TOKEN("DEPTH",          NB_PROPERTY_DEPTH),          \
    NB_TOKEN("FONT",           NB_PROPERTY_FONT),           \
    NB_TOKEN("INPUT",          NB_PROPERTY_INPUT),          \
    NB_TOKEN("OUTPUT",         NB_PROPERTY_OUTPUT),         \
    NB_TOKEN("CHANNEL",        NB_PROPERTY_CHANNEL),        \
    NB_TOKEN("CHAR",           NB_PROPERTY_CHAR),           \
    NB_TOKEN("MACADDR",        NB_PROPERTY_MACADDR),        \
    NB_TOKEN("DNS",            NB_PROPERTY_DNS),            \
    NB_TOKEN("PROMPT",         NB_PROPERTY_PROMPT),         \
    NB_TOKEN("POINT",          NB_PROPERTY_POINT),          \
    NB_TOKEN("MASK",           NB_PROPERTY_MASK),           \
    NB_TOKEN("DIRECTION",      NB_PROPERTY_DIRECTION),      \
    NB_TOKEN("RUN",            NB_PROPERTY_RUN),            \
    NB_TOKEN("POS",            NB_PROPERTY_POS),            \
    NB_TOKEN("READ",           NB_PROPERTY_READ),           \
    NB_TOKEN("BACKGROUND",     NB_PROPERTY_BACKGROUND),     \
    NB_TOKEN("COLOR",          NB_PROPERTY_COLOR),          \
    NB_TOKEN("LIGHT",          NB_PROPERTY_LIGHT),          \
    NB_TOKEN("TEMP",           NB_PROPERTY_TEMP),           \
    NB_TOKEN("HUMIDITY",       NB_PROPERTY_HUMIDITY),       \
    NB_TOKEN("DEWPOINT",       NB_PROPERTY_DEWPOINT),       \
    NB_TOKEN("HEADING",        NB_PROPERTY_HEADING),        \
    NB_TOKEN("TOUCH",          NB_PROPERTY_TOUCH),          \
    NB_TOKEN("PITCH",          NB_PROPERTY_PITCH),          \
    NB_TOKEN("ROLL",           NB_PROPERTY_ROLL),           \
    NB_TOKEN("YAW",            NB_PROPERTY_YAW),            \
    NB_TOKEN("USER",           NB_PROPERTY_USER),           \
    NB_TOKEN("PRESSURE",       NB_PROPERTY_PRESSURE),       \
    NB_TOKEN("SENTENCE",       NB_PROPERTY_SENTENCE),       \
    NB_TOKEN("LATITUDE",       NB_PROPERTY_LATITUDE),       \
    NB_TOKEN("LONGITUDE",      NB_PROPERTY_LONGITUDE),      \
    NB_TOKEN("SATELLITES",     NB_PROPERTY_SATELLITES),     \
    NB_TOKEN("TICK",           NB_PROPERTY_TICK),           \
    NB_TOKEN("I2C",            NB_PROPERTY_I2C),            \
    NB_TOKEN("SPI",            NB_PROPERTY_SPI)
```

CrossWorks CoreBASIC Library

NB_SCRATCH_CELLS

Synopsis

```
#define NB_SCRATCH_CELLS 256
```

Variables grow from the start of memory and the expression stack grows down from the end of memory. Increasing this value means that more complex expressions can be evaluated or more variables can be used. The expression stack is usually very small, but can grow for complex vector operations (but not that much).

CrossWorks CoreBASIC Library

NB_TOKEN_t

Synopsis

```
typedef enum {
  NB_CORE_TOKENS
} NB_TOKEN_t;
```

Description

NB_TOKEN_t enumeration is expanded from the NB_CORE_TOKENS macro and instantiates each core token.

CrossWorks CoreBASIC Library

NB_TRY_CONTEXT_t

Synopsis

```
typedef struct {
  jmp_buf buf;
  NB_TRY_CONTEXT_s *parent;
} NB_TRY_CONTEXT_t;
```

CrossWorks CoreBASIC Library

NB_USE_OWN_ACOSH

Synopsis

```
#define NB_USE_OWN_ACOSH  1
```

Description

Define **NB_USE_OWN_ACOSH** to have the interpreter use its own implementation of acosh.

The system version of **acosh** is preferred, but some compilers do not provide an implementation of **acosh**. In this case, a substitute is provided so that the CoreBASIC intrinsic is still available, but it is not guaranteed to be numerically stable for all arguments.

CrossWorks CoreBASIC Library

NB_USE_OWN_ASINH

Synopsis

```
#define NB_USE_OWN_ASINH  1
```

Description

Define **NB_USE_OWN_ASINH** to have the interpreter use its own implementation of `asinh`.

The system version of **asinh** is preferred, but some compilers do not provide an implementation of **asinh**. In this case, a substitute is provided so that the CoreBASIC intrinsic is still available, but it is not guaranteed to be numerically stable for all arguments.

CrossWorks CoreBASIC Library

NB_USE_OWN_ATANH

Synopsis

```
#define NB_USE_OWN_ATANH  1
```

Description

Define **NB_USE_OWN_ATANH** to have the interpreter use its own implementation of `atanh`.

The system version of **atanh** is preferred, but some compilers do not provide an implementation of **atanh**. In this case, a substitute is provided so that the CoreBASIC intrinsic is still available, but it is not guaranteed to be numerically stable for all arguments.

CrossWorks CoreBASIC Library

nb_array_data

X of type TYPE_ARRAY.

CrossWorks CoreBASIC Library

nb_assign_variable

Synopsis

```
void nb_assign_variable(unsigned index,
                        nb_cell_t *value);
```

Description

nb_assign_variable Assigns the value **value** to the object **obj**.

You can use this function to assign to any object as it copies the type and value fields only, leaving the name field of the destination object **obj** intact.

CrossWorks CoreBASIC Library

nb_boolean_t

Synopsis

```
typedef int nb_boolean_t;
```

Description

Define **nb_boolean_t** to be whatever makes Booleans efficient on your architecture and compiler. For many 8-bit systems, `unsigned char` is an obvious choice, and for 16-bit and 32-bit systems `int` is an obvious choice.

CrossWorks CoreBASIC Library

nb_broadcast_event

Synopsis

```
void nb_broadcast_event(nb_event_t event,
                        unsigned parameter);
```

Description

nb_broadcast_event broadcasts the event **event** and parameter **parameter** to all modules in the module list.

CrossWorks CoreBASIC Library

nb_cell_index

Synopsis

```
#define nb_cell_index(X) ((nb_cell_index_t)((size_t)((unsigned char *)
(X) - (unsigned char *)nb_memory.objects) / sizeof(nb_cell_t)))
```

Description

Although we could simply use X - memory.objects to compute the object index, but that would generate a signed division because of the way that C specifies the result of the difference of two pointers. So, we manually calculate the result which is an unsigned shift.

CrossWorks CoreBASIC Library

nb_cell_index_t

Synopsis

```
typedef unsigned short nb_cell_index_t;
```

Description

nb_cell_index_t defines a cell index into the memory array `nb_memory`.

nb_cell_type_t

Synopsis

```
typedef enum {
  NB_TYPE_INTEGER,
  NB_TYPE_FLOAT,
  NB_TYPE_COMPLEX,
  NB_TYPE_QUATERNION,
  NB_TYPE_STRING,
  NB_TYPE_ARRAY,
  NB_TYPE_PROC,
  NB_TYPE_OBJECT,
  NB_TYPE_REF,
  NB_TYPE_DEFER,
  NB_TYPE_MATRIX,
  NB_TYPE_PROGRAM,
  NB_TYPE_FREE,
  NB_TYPE_SENTINEL
} nb_cell_type_t;
```

Integer, float, and reference objects take one cell and can be held on the evaluation stack. Other objects (array, string, and program) are only held in memory and can span multiple object cells. Free cells are set to NB_TYPE_FREE so that the garbage collector can skip over them.

The ordering of these is specific in that the four language-level types that the user sees come first so that dispatching of operators based on type can be done by indexing a table of function pointers.

NB_TYPE_INTEGER
: A 32-bit signed integer.

NB_TYPE_FLOAT
: A 32-bit floating point value.

NB_TYPE_COMPLEX
: A complex number spanning two cells.

NB_TYPE_QUATERNION
: A quaternion spanning four cells.

NB_TYPE_STRING
: A multi-cell string value.

NB_TYPE_ARRAY
: A multi-cell array value.

NB_TYPE_PROC
: A procedure. This isn't directly accessible.

NB_TYPE_OBJECT
: An object. This isn't directly accessible.

NB_TYPE_REF

A reference to string, array, complex, or quaternion.

NB_TYPE_DEFER

A deferred expression.

NB_TYPE_MATRIX

A two-dimensional array of real values. This type is internal, has a short life, and and is not exposed to the user.

NB_TYPE_PROGRAM

The application program stored in object array. There is exactly one of these over the whole object array.

NB_TYPE_FREE

An unused run of cells in the heap.

NB_TYPE_SENTINEL

The sentinel marker. There is exactly one sentinel cell, the last cell of the memory array, with index `NB_SENTINEL_INDEX`. This cell marks the end of the cell array.

CrossWorks CoreBASIC Library

nb_check_immediate_mode

Synopsis

```
void nb_check_immediate_mode(void);
```

Description

Throw an exception if CoreBASIC is not in immediate mode.

CrossWorks CoreBASIC Library

nb_check_optional_token

Synopsis

```
NB_INLINE nb_boolean_t  NB_INLINE nb_boolean_t nb_check_optional_token(int token);
```

If the token pointed to by the token pointer matches **token**, the token pointer is advanced and this function returns TRUE.

If the token isn't matched, the token pointer is unchanged and this function returns FALSE.

CrossWorks CoreBASIC Library

nb_check_optional_token_2

Synopsis

```
#define nb_check_optional_token_2(X, Y) \
   (nb_check_optional_token(X) || nb_check_optional_token(Y))
```

CrossWorks CoreBASIC Library

nb_check_program_mode

Synopsis

```
void nb_check_program_mode(void);
```

Description

Throw an exception if CoreBASIC is not in program mode.

CrossWorks CoreBASIC Library

nb_check_stack

Synopsis

```
NB_INLINE void nb_check_stack(unsigned entries);
```

Description

Ensure that there are at least **entries** available spaces on the stack.

CrossWorks CoreBASIC Library

nb_clear_flags

Synopsis

```
void nb_clear_flags(unsigned flags);
```

Description

Clear the CoreBASIC flags in **flags**.

CrossWorks CoreBASIC Library

nb_control_stack_item_t

Synopsis

```
typedef struct {
  unsigned token;
  unsigned top;
  unsigned next;
} nb_control_stack_item_t;
```

Description

The stack used by the interpreter to record its statement execution context.

CrossWorks CoreBASIC Library

nb_core_module

Synopsis

```
nb_module_t nb_core_module;
```

Description

nb_core_module is the CORE module.

CrossWorks CoreBASIC Library

nb_current_ctx

Synopsis

```
NB_TRY_CONTEXT_t *nb_current_ctx;
```

Description

nb_current_ctx contains the topmost 'try' context created by **nb_try**. When CoreBASIC throws an error using **nb_throw_exception**, the error is thrown to the topmost try in **nb_current_ctx**.

See Also

nb_try, nb_throw_exception, NB_TRY_CONTEXT_t

CrossWorks CoreBASIC Library

nb_delete_line

Synopsis

```
void nb_delete_line(nb_line_t *lp);
```

Description

nb_delete_line deletes the program line pointed to by **lp**. If **lp** is zero, no modification is made to the program.

Only the program text is modified, nothing else. After deleting the line, you must update the program object by calling nb_fix_program_object at some point.

CrossWorks CoreBASIC Library

nb_delete_line_number

Synopsis

```
void nb_delete_line_number(unsigned line_number);
```

Description

nb_delete_line_number deletes the program line number **line_number**. If the line **line_number** doesn't exist, no modification is made to the program.

CrossWorks CoreBASIC Library

nb_delete_token

Synopsis

```
void nb_delete_token(nb_line_t *line,
                     unsigned char *tp);
```

Description

nb_delete_token deletes a single token at **tp** in the line **line**.

CrossWorks CoreBASIC Library

nb_dyadic_index

Synopsis

```
#define nb_dyadic_index(X) ((X)-NB_TOKEN_EQ)
```

Converts a token into its dyadic table index, so the first token NB_TOKEN_EQ has dyadic index 0, NB_TOKEN_NE has index 1, and so on.

CrossWorks CoreBASIC Library

nb_error_line_number

We use this when typing "EDIT" without a line number so that the line in error is automatically edited, or it's the last line that was edited...

nb_event_t

Synopsis

```
typedef enum {
  NB_REGISTER_MODULE_EVENT,
  NB_COLD_START_EVENT,
  NB_SIGNON_EVENT,
  NB_CLEAR_EVENT,
  NB_RESET_NAME_EVENT,
  NB_SAVE_WORK_EVENT,
  NB_AUTOEXEC_EVENT,
  NB_RUNNING_EVENT,
  NB_READY_EVENT,
  NB_BYE_EVENT
} nb_event_t;
```

Description

nb_event_t describes an event that is broadcast to all installed modules. The events are:

NB_REGISTER_MODULE_EVENT
: CoreBASIC is registering the module and providing the assigned module index.

NB_COLD_START_EVENT
: CoreBASIC is starting from cold: the module should make any required (non-CoreBASIC) initializations.

NB_SIGNON_EVENT
: CoreBASIC is signing on—the module has an opportunity to display additional sign-on details.

NB_CLEAR_EVENT
: CoreBASIC is clearing data when the program is changing or being prepared for execution.

NB_RESET_NAME_EVENT
: CoreBASIC requests that the user program name be reset.

NB_SAVE_WORK_EVENT
: CoreBASIC requests that the user program is saved to disk.

NB_RUNNING_EVENT
: Indicate whether CoreBASIC is running user code or not. The parameter passed with the event is non-zero when running user code.

NB_READY_EVENT
: Indicates that CoreBASIC is at the *ready* prompt and waiting for input.

NB_AUTOEXEC_EVENT
: Request that a file be executed. The file name is passed in `nb_program_name`.

NB_BYE_EVENT

Broadcast when closing a CoreBASIC session with BYE. Modules can close down whatever they need and, optionally, terminate CoreBASIC.

CrossWorks CoreBASIC Library

nb_execute_quick_recycle

Synopsis

```
void nb_execute_quick_recycle(void);
```

Description

this runs the garbage collector to throw away all orphaned objects. As a CoreBASIC application programmer, you don't need to worry about executing RECYCLE as the interpreter automatically recycles memory when it needs to. However, if you modify the CoreBASIC interpreter, you need to be acutely aware that creating object may well cause a flash mark-sweep collection or, worse, a compacting collection.

CrossWorks CoreBASIC Library

nb_execute_recycle

Synopsis

```
void nb_execute_recycle(void);
```

Description

this runs the garbage collector to throw away all orphaned objects. As a CoreBASIC application programmer, you don't need to worry about executing RECYCLE as the interpreter automatically recycles memory when it needs to. However, if you modify the CoreBASIC interpreter, you need to be acutely aware that creating object may well cause a flash mark-sweep collection or, worse, a compacting collection.

CrossWorks CoreBASIC Library

nb_find_line

Synopsis

```
nb_line_t *nb_find_line(unsigned line_number);
```

Description

nb_find_line finds the program line with line number **line_number** and peturns a pointer to the line structure if the line exists, or a zero pointer if it doesn't.

CrossWorks CoreBASIC Library

nb_first_line

Synopsis

```
nb_line_t *nb_first_line(void);
```

Description

nb_first_line returns a pointer to the first line in the program.

Note

Although this looks like it should be a macro, eventually it could support programs frozen into FLASH rather than in RAM, and at this point **nb_first_line** will return an address in RAM or FLASH depending upon whether the program is frozen or thawed.

CrossWorks CoreBASIC Library

nb_fix_program_object

Synopsis

```
void nb_fix_program_object(void);
```

Description

nb_fix_program_object Fixes up the program object so that it reflects the program's true size. After any changes to the program, you *must* call **nb_fix_program_object** otherwise the garbage collector will potentially sweep off the end of the program causing havoc. The low-level routines don't call this by design as it's a fairly expensive operation—it's up to high-level code to call this when the program has been changed.

nb_flag_t

Synopsis

```
typedef enum {
  NB_FLAG_BREAK,
  NB_FLAG_WATCHDOG,
  NB_FLAG_STEP,
  NB_FLAG_TRACE
} nb_flag_t;
```

The state variable `nb_flags` is a bitmask with each bit corresponding to one of these flags. The interpreter examines `nb_flags` before it starts to execute a new statement to see if any special processing is required.

NB_FLAG_BREAK

Break execution back to top level. Setting this flag causes the interpreter to exit to the top-level loop using the `NB_BREAKPOINT` exception. This flag can be set asynchronously to indicate to the interpreter that the user has pressed the break key.

NB_FLAG_WATCHDOG

Break execution back to top level and indicate that this is a watchdog timeout. Setting this flag causes the interpreter to exit to the top-level loop using the `NB_WATCHDOG` exception. This flag can be set asynchronously to indicate to the watchdog has fired when debugging the application.

NB_FLAG_STEP

Step one statement and then set the `NB_FLAG_BREAK` flag.

NB_FLAG_TRACE

Trace-enabled flag. Setting this flag causes the interpreter to print each statement with a before it executes it.

See Also

nb_flags

CrossWorks CoreBASIC Library

nb_flags

Synopsis

```
unsigned char nb_flags;
```

Description

nb_flags variable holds the interpreter mode flags. **nb_flags** is checked to see if any flag is set before each statement is run and, if a flag is set, the flags are serviced.

Note

Do not extend the flags in **nb_flags** without understanding what the implications are. If you add another flag, for instance, to indicate some interpreter mode, setting that flag will cause the interpreter's performance to degrade because on each statement the flag will be set and the interpreter will try to service the flag. Better would be to move that flag into another variable and keep **nb_flags** uncluttered for best performance.

CrossWorks CoreBASIC Library

nb_immediate_mode

Synopsis

```
nb_boolean_t nb_immediate_mode;
```

Description

nb_immediate_mode Boolean indicates whether the interpreter is in immediate mode or program mode.

See Also

nb_check_immediate_mode, nb_check_program_mode

CrossWorks CoreBASIC Library

nb_input_buffer

Synopsis

```
char nb_input_buffer[];
```

Description

nb_input_buffer is the input buffer used inside CoreBASIC to gather input from the user. It is also used as a scratch buffer by some functions as, after tokenization, this buffer is of no further use.

CrossWorks CoreBASIC Library

nb_is_end_of_line

Synopsis

```
#define nb_is_end_of_line(X)  ((X) <= NB_TOKEN_PAD)
```

nb_is_end_of_statement

Synopsis

```
#define nb_is_end_of_statement(X)  ((X) <= NB_TOKEN_LINE_ELSE)
```

Description

nb_is_end_of_statement returns whether the token X is an end of statement token.

Logically, this is X == NB_TOKEN_EOL || X == NB_TOKEN_PAD || X == NB_TOKEN_COLON..., but by using the fact that all these tokens are grouped together about zero, a much simpler test is used. A statement is ended by an in-line remark, the introduction of an ELSE, a statement separator, or a real end of line.

CrossWorks CoreBASIC Library

nb_is_zero

Synopsis

```
nb_boolean_t nb_is_zero(nb_cell_t *x);
```

Returns 1 if **x** is zero and throws a type mismatch exception if the top of stack is not numeric.

CrossWorks CoreBASIC Library

nb_join_lines

Synopsis

```
void nb_join_lines(nb_line_t *line);
```

Description

nb_join_lines joins the line **line** with the following line.

CrossWorks CoreBASIC Library

nb_line_number_referenced

Synopsis

```
int nb_line_number_referenced(int line_number);
```

Description

nb_line_number_referenced returns non-zero if the line with line number **line_number** is referenced by a statement in the program.

nb_line_t

Synopsis

```
typedef struct {
  NB_LINE_HEADER_t header;
  unsigned char tokens[];
} nb_line_t;
```

The CoreBASIC interpreter only uses pointers to a nb_line_t so it doesn't matter about the size of the tokens member, but defining it to have 32 token bytes means that it's easy to take a look at the line and its tokens in the debugger.

`header`

The line header.

`tokens`

Tokens comprising the line, of dynamic length, but set to 32 for the debugger.

CrossWorks CoreBASIC Library

nb_matrix_data

X of type TYPE_MATRIX.

nb_module_t

Synopsis

```
typedef struct {
  const char *const name;
  void (*execute_statement)(void);
  void (*check_statement)(void);
  void (*evaluate_intrinsic)(void);
  void (*check_intrinsic)(void);
  void (*event)(nb_event_t , unsigned);
  const char *const *keywords;
  const  nb_monadic_dispatch_tag *apply_monadic;
} nb_module_t;
```

Statement extensions must be *simple*. That is, they can't refer to line numbers (i.e. provide a different naming of GOTO, for instance) and can't introduce control structures.

`name`

The module name.

`execute_statement`

Method to execute the statement at **nb_tp**.

`check_statement`

Method to syntax check the statement at **nb_tp**

`evaluate_intrinsic`

Method to evaluate the intrinsic at **nb_tp**.

`check_intrinsic`

Method to syntax check the intrinsic at **nb_tp**.

`event`

Method to progress a CoreBASIC event.

`keywords`

The array of keywords recognized by this module.

`apply_monadic`

Monadic dispatch table.

CrossWorks CoreBASIC Library

nb_modules

Synopsis

```
nb_module_t *const nb_modules[];
```

The first module *must* be the core set module `nb_core_module`, and other modules follow.

CrossWorks CoreBASIC Library

nb_monadic_index

Synopsis

```
#define nb_monadic_index(X) ((X)-NB_TOKEN_NEGATE)
```

Converts a token into its monadic table index so the first token, NB_TOKEN_NEGATE has monadic index 0, NB_TOKEN_SGN has index 1, and so on.

CrossWorks CoreBASIC Library

nb_next_line

Synopsis

```
#define nb_next_line(X) ((nb_line_t *)((unsigned char *)(X) + (X)->header.next_offset))
```

Description

nb_next_line returns a pointer to the start of the line following **X**.

CrossWorks CoreBASIC Library

nb_print_line

Synopsis

```
void nb_print_line(nb_line_t *lp,
                   const unsigned char *tp,
                   int indent);
```

Description

nb_print_line prints a tokenized line to standard output. The line number is printed in a width of five, followed by the tokens, followed by a newline.

CrossWorks CoreBASIC Library

nb_print_tokens

Synopsis

```
void nb_print_tokens(const unsigned char *tp,
                     const unsigned char *mark);
```

Description

This prints a sequence of tokens to standard output.

Tokens are printed starting at **tp** and continue printing until the end-of-line marker is seen. No newline is printed at the end of line.

During printing, if any token address covers **mark**, then a special mark is shown in the printed output to mark the token. This allows more precise information when printing errors and is also used when tracing a program to indicate which statement will be executed next.

CrossWorks CoreBASIC Library

nb_program_address

Synopsis

```
#define nb_program_address(X) (nb_memory.bytes + (X))
```

Description

nb_program_address returns a token pointer corresponding to the 16-bit offset **X**.

See Also

nb_program_offset.

CrossWorks CoreBASIC Library

nb_program_end

Synopsis

```
unsigned char *nb_program_end(void);
```

Description

nb_program_end returns the address of the first byte following the end of the program.

Consider a program containing a single line `10 PRINT`. In fact, there is always an additional phantom line at the end which has line number 0 with the `END` token on it. So, the program looks like this in memory:

```
06 00 0a 00 NB_TOKEN_PRINT NB_TOKEN_EOL \n
00 00 00 00 NB_TOKEN_END NB_TOKEN_EOL
```

So, the program size is 12 bytes and the returned address points to the byte following the second `NB_TOKEN_PAD`

See Also

nb_program_size

CrossWorks CoreBASIC Library

nb_program_offset

Synopsis

```
#define nb_program_offset(TP) ((unsigned short)((unsigned char *)(TP) - nb_memory.bytes))
```

Description

nb_program_offset returns a 16-bit offset from the start of the CoreBASIC program to the token pointer **TP**

See Also

nb_program_address.

CrossWorks CoreBASIC Library

nb_program_size

Synopsis

```
unsigned nb_program_size(void);
```

Description

nb_program_size returns the number of bytes occupied by the program excluding the program header cell.

See Also

nb_program_end

CrossWorks CoreBASIC Library

nb_property_t

Synopsis

```
typedef enum {
  NB_PROPERTIES
} nb_property_t;
```

Description

nb_property_t enumeration is expanded from the NB_PROPERTIES macro and instantiates each property.

CrossWorks CoreBASIC Library

nb_push_integer

Synopsis

```
void nb_push_integer(nb_int_t i);
```

Description

Push the integer value **i** onto the expression stack.

Note

There is no check for stack overflow: use **nb_check_stack** beforehand.

CrossWorks CoreBASIC Library

nb_replace_float

Synopsis

```
void nb_replace_float(nb_float_t f);
```

Note

There is no requirement for a stack check when you use this as it is assumed that the stack contains at least one item.

CrossWorks CoreBASIC Library

nb_replace_integer

Synopsis

```
void nb_replace_integer(nb_int_t i);
```

There is no need for a stack check when you use this as it is assumed that the stack has at least one item on it.

CrossWorks CoreBASIC Library

nb_replace_line

Synopsis

```
void nb_replace_line(unsigned line_number,
                     unsigned char *tokens);
```

Description

nb_replace_line replaces the program line with line number **line_number** with the tokenized text **tokens**. If the line **line_number** does not already exist, is inserted into the program.

CrossWorks CoreBASIC Library

nb_sentinel_index

Synopsis

```
unsigned nb_sentinel_index;
```

CrossWorks CoreBASIC Library

nb_signon

Synopsis

```
void nb_signon(void);
```

Description

Traverse all modules and displays their sign-on messages.

CrossWorks CoreBASIC Library

nb_sp

Synopsis

```
nb_cell_t *nb_sp;
```

Description

nb_sp is a pointer to the last item pushed onto the evaluation stack

CrossWorks CoreBASIC Library

nb_string_data

X of type TYPE_STRING.

CrossWorks CoreBASIC Library

nb_stringize

Synopsis

```
void nb_stringize(void);
```

Description

nb_stringize converts the top of stack to a string by applying STR.

CrossWorks CoreBASIC Library

nb_throw_exception

Synopsis

```
__NB_NO_RETURN void nb_throw_exception(CTL_STATUS_t code);
```

Description

Throw a CoreBASIC exception with code **code**. Throwing an exception will return to the top-level loop.

CrossWorks CoreBASIC Library

nb_token_auto

Synopsis

```
unsigned short nb_token_auto;
```

Description

nb_token_auto contains the token index of the AUTO token.

CrossWorks CoreBASIC Library

nb_token_catalog

Synopsis

```
unsigned short nb_token_catalog;
```

Description

nb_token_catalog contains the token index of the CATALOG token.

CrossWorks CoreBASIC Library

nb_token_cd

Synopsis

```
unsigned short nb_token_cd;
```

Description

nb_token_cd contains the token index of the CD token.

CrossWorks CoreBASIC Library

nb_token_chain

Synopsis

```
unsigned short nb_token_chain;
```

Description

nb_token_chain contains the token index of the `CHAIN` token.

CrossWorks CoreBASIC Library

nb_token_chdir

Synopsis

```
unsigned short nb_token_chdir;
```

Description

nb_token_chdir contains the token index of the `CHDIR` token.

CrossWorks CoreBASIC Library

nb_token_check

Synopsis

```
unsigned short nb_token_check;
```

Description

nb_token_check contains the token index of the CHECK token.

CrossWorks CoreBASIC Library

nb_token_dir

Synopsis

```
unsigned short nb_token_dir;
```

Description

nb_token_dir contains the token index of the `DIR` token.

CrossWorks CoreBASIC Library

nb_token_example

Synopsis

```
unsigned short nb_token_example;
```

Description

nb_token_example contains the token index of the EXAMPLE token.

CrossWorks CoreBASIC Library

nb_token_flush

Synopsis

```
unsigned short nb_token_flush;
```

Description

nb_token_flush contains the token index of the FLUSH token.

CrossWorks CoreBASIC Library

nb_token_get

Synopsis

```
unsigned short nb_token_get;
```

Description

nb_token_get contains the token index of the GET token.

CrossWorks CoreBASIC Library

nb_token_history

Synopsis

```
unsigned short nb_token_history;
```

Description

nb_token_history contains the token index of the `HISTORY` token.

CrossWorks CoreBASIC Library

nb_token_kill

Synopsis

```
unsigned short nb_token_kill;
```

Description

nb_token_kill contains the token index of the `KILL` token.

CrossWorks CoreBASIC Library

nb_token_length_inline

tp.

CrossWorks CoreBASIC Library

nb_token_load

Synopsis

```
unsigned short nb_token_load;
```

Description

nb_token_load contains the token index of the LOAD token.

CrossWorks CoreBASIC Library

nb_token_reboot

Synopsis

```
unsigned short nb_token_reboot;
```

Description

nb_token_reboot contains the token index of the REB00T token.

CrossWorks CoreBASIC Library

nb_token_save

Synopsis

```
unsigned short nb_token_save;
```

Description

nb_token_save contains the token index of the SAVE token.

CrossWorks CoreBASIC Library

nb_tokenize

Synopsis

```
void nb_tokenize(char *text,
                 unsigned char *tp);
```

Description

nb_tokenize tokenizes the input **text** into the token stream **tokens**.

CrossWorks CoreBASIC Library

nb_truth_value

Synopsis

```
#define nb_truth_value(X) ((X) ? 1 : 0)
```

CrossWorks CoreBASIC Library

nb_try

Synopsis

```
#define nb_try(X)  ((X)->parent = nb_current_ctx, nb_current_ctx = X, setjmp((X)->buf))
```

CrossWorks CoreBASIC Library

nb_unwind_try

Synopsis

```
void nb_unwind_try(void);
```

Description

nb_unwind_try unwinds the topmost try. After catching an exception, you must unwind the try stack.

CrossWorks CoreBASIC Library

nb_var_front_index

Synopsis

```
unsigned nb_var_front_index;
```

Description

nb_var_front_index is the highest unused variable index in scratch memory.