Nextmidas CodeCount™

Counting Standard

University of Southern California

Center for Systems and Software Engineering

December , 2016

Revision Sheet

Date	Version	Revision Description	Author
8/25/2016	1.0	Original Release	Matthew Swartz

Table of Contents

1. Definitions

NOTE: This document covers both the NeXtMidas macro language as well as the similar updated NeXtMidas macro language. Items denoted by (XM) indicate NeXtMidas exclusive keywords, and items denoted by (NM) indicate NeXtMidas exclusive keywords.

1.1. **SLOC –** Source Lines of Code is a unit used to measure the size of software program. SLOC counts the program source code based on a certain set of rules. SLOC is a key input for estimating project effort and is also used to calculate productivity and other measurements.

1.2. **Physical SLOC –** One physical SLOC is corresponding to one line starting with the first character and ending by a carriage return or an end-of-file marker of the same line, and which excludes the blank and comment line.

1.3. **Logical SLOC –** Lines of code intended to measure "statements", which normally terminate by a semicolon (C/C++, Java, C#) or a carriage return (VB, Assembly, NeXtMidas), etc. Logical SLOC are not sensitive to format and style conventions, but they are language-dependent.

1.4. **Data declaration line or data line –** A line that contains declaration of data and used by an assembler or compiler to interpret other elements of the program.

The following table lists the NeXtMidas keywords that denote data declaration lines:

global

Table 1 NeXtMidas Data Keywords

1.5. **Compiler Directives –** A statement that tells the compiler how to compile a program, but not what to compile.

The following table lists the NeXtMidas keywords that denote compiler directive lines:

include

Table 2 Fortran Compiler Directive

1.6. **Blank Line –** A physical line of code, which contains any number of white space characters (spaces, tabs, form feed, carriage return, line feed, or their derivatives).

1.7. **Comment Line –** A comment is defined as a string of zero or more characters that follow language-specific comment delimiter. NeXtMidas comment delimiter is "!". A whole comment line may span one line and does not contain any compilable source code. An embedded comment can co-exist with compilable source code on the same physical line. Banners and empty comments are treated as types of comments.

1.8. **Executable Line of code –** A line that contains software instruction executed during runtime and on which a breakpoint can be set in a debugging tool. An instruction can be stated in a simple or compound form.

An executable line of code may contain the following program control statements:

- Selection statements (if)
- Iteration statements (loop, while, forall)
- Jump statements (return, goto, break, continue)
- Expression statements (macro/subroutine/procedure calls, assignment statements, operations, etc.)

An executable line of code may not contain the following statements:

- Compiler directives
- Data declaration (data) lines
- Whole line comments, including empty comments and banners
- Blank lines

2. Checklist for source statement counts

PHYSICAL SLOC COUNTING RULES			
MEASUREMENT UNIT	ORDER OF PRECEDENCE	PHYSICAL SLOC	COMMENTS
Executable lines	1	One per line	Defined in 1.8
Non-executable lines			
Declaration (Data) lines	2	One per line	Defined in 1.4
Compiler directives	3	One per line	Defined in 1.5
Comments			Defined in 1.7
On their own lines	4	Not included (NI)	
Embedded	5	NI	
Banners	6	NI	
Empty comments	7	NI	
Blank lines	8	NI	Defined in 1.6

LOGICAL SLOC COUNTING RULES				
NO.	STRUCTURE	ORDER OF PRECEDENCE	LOGICAL SLOC RULES	COMMENTS
R01	"loop", "while" or "if" statement	1	Count once	
R02	Data declaration and data assignment	2	Count once	
R03	Jump statement	3	Count once per keyword	
R04	Macro/subroutine/procedure call	4	Count once per call	
R05	Keyword statement	5	Count once per statement	

3. Examples

EXECUTABLE LINES

SELECTION Statement

ESS1 - if, else if, else and nested if statement

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
if <boolean expression>	if x neq 0	1
<statements>	say "non-zero"	1
if <boolean expression>	if x gt 0	1
<statements>	say "positive"	1
else	else	0
<statements>	say "negative"	1
endif	endif	0
if <boolean expression>	if x eq 0	1
<statements>	say "zero"	1
elseif <boolean expression>	elseif x gt 0	1
<statements>	say "positive"	1
.	.	
.	.	
else	else	0
<statements>	say "negative"	1
endif	endif	0
if <boolean expression> then <statement>	if x neq 0 then say "positive"	2
NOTE: complexity is not considered, i.e. multiple "and" or "or" as part of the expression.		

ESS2 - trap statement

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
	trap error FOUNDERR	1
trap error <label name>	.	
.	.	
.	endmode	1
endmode (or stop)	label FOUNDERR	0
	error "Found an error!"	1

ITERATION Statement

EIS1 – loop statement

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
loop <iterations> <count>	loop 10 count	1
<statements>	say count	1
endloop	endloop	0

EIS2 – empty statement

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
loop <iterations> <count>	loop 10 count	1
endloop	endloop	0

EJS1 – while statement

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
while <boolean expression>	while i lt 10	1
<statements>	say "^i"	1
endwhile	calc i i 1 +	1
	endwhile	0

EJS2 – forall statement

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
forall #=<start>:<end>;<inc> <command>	forall #=1:21;2 calc n n # +	2

EJS3 - do statements (NM)

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
do <count> <start> <end> <inc>	do count 1 7 1	1
<statements>	say "The count is at ^count"	1
enddo	enddo	0

EJS4 – foreach statement(NM)

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
foreach <item> <func> <in>	foreach key INTABLE mytable	1
<statements	say "Key ^key = ^mytable.^key"	1
endfor	endfor	0

JUMP Statement

EES1 – return statement

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
return	return	1

EES2 – goto, label statement

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
goto <label name> . . label <label name>	label loop1 calc x x 1 + if x lt y then goto loop1	0 1 2

EES3 – break statement

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
break	if i gt 10 then break	2

ESS2 - continue statement

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT

continue	if [["$name" != *[[:upper:]]*]];
	then
	continue
	fi
		1
		0
		1
		0

Expression Statement

EIS1 – macro call

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
<macro name> <parameters>	read_file name	1

EIS2 – subroutine call

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
call <subroutine name>	call read_file name	1

EJS1 – procedure call

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
jump <procedure name>	jump read_file name	1

EJS2 – assignment statement

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
results <name> <value>	results x 1	1

DECLARATION OR DATA LINES

DDL1 - function declaration subroutine declaration variable declaration type declaration

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
global <type>:<name>	global A:param global amount, sum, total	1 1

COMPILER DIRECTIVES

CDL1 – directive type

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
include <macro name>	include %MACRO	1

4. Complexity

Complexity measures the occurrences of different keywords in code baseline. Below table identifies the categories and their respective keywords that are counted as part of the complexity metrics.

Table 3 - Complexity Keywords List

Math Functions	Trig	Log	Calculations	Conditionals	Logic	Pre-processor	Assignment	Pointer
histogram	sincosine	logarithm	**	foreach	neqss	include	results	
magnitude	waveform		+	elseif	eqss			
normalize			-	forall	neqs			
polyphase			*	while	and			
transform			/	else	eqs			
transpose				loop	ngt			
peakpick				trap	nlt			
firhlbrt				do	nge			
firparks				if	nle			
multiply					neq			
passfilt					or			
polyeval					gt			
subtract					lt			
transfft					ge			
firwind					le			
hilbert					eq			
polyfit								
spectra								
maxmin								
invfft								
marray								
modulo								
parray								
random								
sarray								
smooth								
fcalc								
imfft								
morph								
mpoly								
phase								
polar								
pulse								
calc								
ramp								
mfft								
fft								