X-Midas CodeCount™ Counting Standard

University of Southern California

Center for Systems and Software Engineering

October , 2009

Revision Sheet

Date	Version	Revision Description	Author
10/30/2009	1.0	Original Release	CSSE
1/2/2013	1.1	Updated document template	CSSE

Table of Contents

1. Definitions

NOTE: This document covers both the X-Midas macro language as well as the similar updated NeXtMidas macro language. Items denoted by (XM) indicate X-Midas 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, X-Midas), 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 X-Midas keywords that denote data declaration lines:

local (XM)
global (NM)

Table 1 Data Declaration Types

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 X-Midas keywords that denote compiler directive lines:

include

Table 2 Compiler Directives

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.

X-Midas 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, elseif, else and nested if statements

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
if <boolean expression> <statements>	if x neq 0 say “non-zero”	1 1

if <boolean expression>
  <statements>
else
  <statements>
endif
	if x gt 0
	  say “positive”
	else
	  say “negative”
	endif
		1
		1
		0
		1
		0

if <boolean expression>
  <statements>
elseif <boolean expression>
  <statements>
.
.
else
  <statements>
endif
	if x eq 0
	  say “zero”
	elseif x gt 0
	  say “positive”
	.
	.
	else
	  say “negative”
	endif
		1
		1
		1
		1
		
		0
		1
		0

if <boolean expression> then <statement> NOTE: complexity is not considered, i.e. multiple “and” or “or” as part of the expression.	if x neq 0 then say “positive”	2

ESS2 – trap

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

ITERATION Statement

EIS1 – loop

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

EIS2 – empty statements (could be used for time delays)

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

EIS3 – while

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT

while <boolean expression>
  <statements>
endwhile
	while i lt 10
	  say “^i”
	  calc i i 1 +
	endwhile
		1
		1
		1
		0

EIS4 – forall

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

EIS5 – do (NM)

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

EIS6 – foreach (NM)

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

JUMP Statement

EJS1 – return

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
return	return	1

EJS2 – goto, label

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

EJS3 – break

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

EJS4 – continue

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
continue	continue	if i gt 10 then continue

EXPRESSION Statement

EES1 – macro call

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

EES2 – subroutine call

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

EES3 – procedure call

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

EES4 – assignment statement

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

DECLARATION OR DATA LINES

DDL1 – variable declaration (XM)

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

DDL2 – variable declaration (NM)

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

COMPILER DIRECTIVES

CDL1 – directive types

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