X-Midas CodeCount™

Counting Standard

University of Southern California

Center for Systems and Software Engineering

Fall 2009

Revision Sheet

Date	Version	Revision Description	Author
10/30/09	1.0	Original Release	CSSE

1.0 CHECKLIST FOR SOURCE STATEMENT COUNTS

PHYSICAL AND LOGICAL SLOC COUNTING RULES

Measurement Unit	Order of Precedence	Physical SLOC	Logical SLOC	Comments
Executable lines	1	One per line	See table below	Defined in 2.9
Non-executable lines				
Declaration (Data) lines	2	One per line	See table below	Defined in 2.4
Compiler directives	3	One per line	See table below	Defined in 2.5
Comments				Defined in 2.8
On their own lines	4	Not included (NI)	NI	
Embedded	5	NI	NI	
Banners	6	NI	NI	
Empty comments	7	NI	NI	
Blank lines	8	NI	NI	Defined in 2.7

Table 1 Physical and Logical SLOC Counting Counts

LOGICAL SLOC COUNTING RULES

No.	Structure	Order of Precedence	Logical SLOC Rules	Comments
R01	"*loop*", "*while*" or "*if*" statement	1	Count once per structure	
R02	Data declaration and data assignment	2	Count once per declaration/assignment	
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	

Table 2 Logical SLOC Counting Rules

2.0 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.

2.1 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.

2.2 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.

2.3 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.

2.4 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 X-Midas keywords that denote data declaration lines:

local (XM)	global (NM)		

Table 3 Data Declaration Types

NOTE: See Section 3 of this document for examples of data declaration lines.

2.5 Compiler directive - A statement that tells the compiler how to compile a program, but not what to compile.

A list of common X-Midas directives is presented in the table below:

include			

Table 4 Compiler Directives

NOTE: See Section 3 of this document for examples of compiler directive lines.

2.6 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).

2.7 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.

2.8 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.)

  NOTE: See Section 3 of this document for examples of control statements.

- 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

3.0 EXAMPLES OF LOGICAL SLOC COUNTING

EXECUTABLE LINES				
SELECTION STATEMENTS				
ID	**STATEMENT DESCRIPTION**	**GENERAL FORM**	**SPECIFIC EXAMPLE**	**SLOC COUNT**
ESS1	If, elseif, else, and nested if statements	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	trap error <label name>	trap error FOUNDERR	1
		.	.	
		.	.	
		endmode (or stop)	endmode	1
			label FOUNDERR	0
			error “Found an error!”	1

ITERATIONS STATEMENTS				
ID	**STATEMENT DESCRIPTION**	**GENERAL FORM**	**SPECIFIC EXAMPLE**	**SLOC COUNT**
EIS1	loop	loop <iterations> <count>	loop 10 count	1

DECLARATION (DATA) LINES

ID	STATEMENT DESCRIPTION	GENERAL FORM	SPECIFIC EXAMPLE	SLOC COUNT
DDL1	variable declaration (XM)	local <type>:<name>	local A:param local amount, sum, total	1 1
DDL2	variable declaration (NM)	global <type>:<name>	global A:param global amount, sum, total	1 1

COMPILER DIRECTIVES

ID	STATEMENT DESCRIPTION	GENERAL FORM	SPECIFIC EXAMPLE	SLOC COUNT
CDL1	directive types	include <macro name>	include %MACRO	1