**Simulinkブロックの性能計測**

**使用説明書**

株式会社NEC情報システムズ

2015年12月21日

**目次**

[1. はじめに 2](#_Toc438038318)

[2. ソフトウェア構成 3](#_Toc438038319)

[2.1. 実行環境 4](#_Toc438038320)

[2.2. ファイル一覧 5](#_Toc438038321)

[3. ビルド手順 7](#_Toc438038322)

[3.1. ビルド準備 7](#_Toc438038323)

[3.1.1. JDK8のインストール 7](#_Toc438038324)

[3.1.2. Apache Antのインストール 7](#_Toc438038325)

[3.1.3. SimulinkModel.xsdの配置 7](#_Toc438038326)

[3.2. ビルド 7](#_Toc438038327)

[3.3. javadocの生成(オプショナル) 7](#_Toc438038328)

[4. 実行 8](#_Toc438038329)

[4.1. AddMeasurementコマンド仕様(javaコマンド使用時) 8](#_Toc438038330)

[4.2. AddMeasurement.shコマンド仕様(shellスクリプト使用時) 9](#_Toc438038331)

[4.3. AddMeasurementの制限事項 9](#_Toc438038332)

[4.4. AddMeasurementによる計測Cコードのマクロ 10](#_Toc438038333)

[4.4.1. ブロック番号 10](#_Toc438038334)

[4.4.2. 最大ブロック番号 10](#_Toc438038335)

[4.4.3. ブロック固有パラメタ(ターゲット依存) 10](#_Toc438038336)

[4.4.4. ブロック名の配列 10](#_Toc438038337)

[4.4.5. step関数およびinitialize関数名のマクロ 11](#_Toc438038338)

[4.4.6. if文の条件式のラップマクロ(ターゲット依存) 11](#_Toc438038339)

[4.4.7. TIDの条件式のラップマクロ(ターゲット依存) 11](#_Toc438038340)

[4.4.8. step関数の開始マクロ(ターゲット依存) 12](#_Toc438038341)

[4.4.9. step関数の終端マクロ(ターゲット依存) 12](#_Toc438038342)

[4.4.10. ブロックの計測開始マクロ(ターゲット依存) 12](#_Toc438038343)

[4.4.11. ブロックの計測停止マクロ(ターゲット依存) 13](#_Toc438038344)

[4.5. AddMeasurementが出力するブロック情報のファイル形式 13](#_Toc438038345)

[4.6. SetMeasuringResultコマンド仕様(javaコマンド使用時) 14](#_Toc438038346)

[4.7. SetMeasuringResultコマンド仕様(shellスクリプト使用時) 14](#_Toc438038347)

[4.8. SetMeasuringResult計測結果CSVファイル 15](#_Toc438038348)

[5. モジュール仕様 16](#_Toc438038349)

[5.1. AddMeasurementのクラス 16](#_Toc438038350)

[5.1.1. AddMeasureクラス 24](#_Toc438038351)

[5.1.1.1. 列挙子SubsystemType 24](#_Toc438038352)

[5.1.1.2. 列挙子PointType 24](#_Toc438038353)

[5.1.1.3. 静的メンバverbose 24](#_Toc438038354)

[5.1.1.4. 静的メンバoutputMacroName 24](#_Toc438038355)

[5.1.1.5. 静的メンバmacroPrefix 24](#_Toc438038356)

[5.1.1.6. 静的メンバincludeFiles 25](#_Toc438038357)

[5.1.1.7. 静的メンバbeginHierarchy 25](#_Toc438038358)

[5.1.1.8. 静的メンバblockHierarchy 25](#_Toc438038359)

[5.1.1.9. 静的メンバstepFunction 26](#_Toc438038360)

[5.1.1.10. 静的メンバtidPattern 26](#_Toc438038361)

[5.1.1.11. 静的メンバifPattern 26](#_Toc438038362)

[5.1.1.12. 静的メンバblockMark 27](#_Toc438038363)

[5.1.1.13. 静的メンバindexEndOf 27](#_Toc438038364)

[5.1.1.14. 静的メンバindexUpdateOf 28](#_Toc438038365)

[5.1.1.15. 静的メンバindexType 28](#_Toc438038366)

[5.1.1.16. 静的メンバindexSfUpdate 28](#_Toc438038367)

[5.1.1.17. 静的メンバindexSubSys 28](#_Toc438038368)

[5.1.1.18. 静的メンバindexBlock 28](#_Toc438038369)

[5.1.1.19. 静的メンバindexSystem 28](#_Toc438038370)

[5.1.1.20. 静的メンバindexCCB 29](#_Toc438038371)

[5.1.1.21. 静的メンバindexSC 29](#_Toc438038372)

[5.1.1.22. 静的メンバindexName 29](#_Toc438038373)

[5.1.1.23. 静的メンバblockEndMark 29](#_Toc438038374)

[5.1.1.24. 静的メンバhierarchyMap 29](#_Toc438038375)

[5.1.1.25. 静的メンバblockNameMap 30](#_Toc438038376)

[5.1.1.26. 静的メンバblockMap 30](#_Toc438038377)

[5.1.1.27. 静的メンバmeasurePointList 30](#_Toc438038378)

[5.1.1.28. 静的メンバstartComment 30](#_Toc438038379)

[5.1.1.29. 静的メンバendComment 30](#_Toc438038380)

[5.1.1.30. 静的メンバstartStatement 30](#_Toc438038381)

[5.1.1.31. 静的メンバendStatement 30](#_Toc438038382)

[5.1.1.32. 静的メンバstartBlock 31](#_Toc438038383)

[5.1.1.33. 静的メンバendBlock 31](#_Toc438038384)

[5.1.1.34. 静的メンバtidInfo 31](#_Toc438038385)

[5.1.1.35. 静的メンバparsePoint 31](#_Toc438038386)

[5.1.1.36. 静的メンバcodeEnd 31](#_Toc438038387)

[5.1.1.37. 静的メンバstartBlockComment 31](#_Toc438038388)

[5.1.1.38. 静的メンバcurrentBlock 31](#_Toc438038389)

[5.1.1.39. 静的メンバisUpdate 31](#_Toc438038390)

[5.1.1.40. 静的メンバhasIf 32](#_Toc438038391)

[5.1.1.41. 静的メンバpreamble 32](#_Toc438038392)

[5.1.1.42. 静的メンバpostamble 32](#_Toc438038393)

[5.1.1.43. 静的メンバrestart 32](#_Toc438038394)

[5.1.1.44. 静的メンバstateStack 32](#_Toc438038395)

[5.1.1.45. 静的メンバstepFunctionName 32](#_Toc438038396)

[5.1.1.46. 静的メンバinitializeFunctionName 32](#_Toc438038397)

[5.1.1.47. 静的メンバstepFunctionMacroName 33](#_Toc438038398)

[5.1.1.48. 静的メンバinitializeFunctionMacroName 33](#_Toc438038399)

[5.1.1.49. 静的メンバifCondtionMacroName 33](#_Toc438038400)

[5.1.1.50. 静的メンバtidCondtionMacroName 33](#_Toc438038401)

[5.1.1.51. 静的メンバstartMacroName 33](#_Toc438038402)

[5.1.1.52. 静的メンバstopMacroName 33](#_Toc438038403)

[5.1.1.53. 静的メンバstepBeginMacroName 33](#_Toc438038404)

[5.1.1.54. 静的メンバstepEndMacroName 34](#_Toc438038405)

[5.1.1.55. メソッドaddSkipPoint () 34](#_Toc438038406)

[5.1.1.56. メソッドaddReplacePoint () 34](#_Toc438038407)

[5.1.1.57. メソッドaddInsertPoint () 34](#_Toc438038408)

[5.1.1.58. メソッドaddDeletePoint () 34](#_Toc438038409)

[5.1.1.59. メソッドaddMeasurePoint () 34](#_Toc438038410)

[5.1.1.60. メソッドcanonicalize () 34](#_Toc438038411)

[5.1.1.61. メソッドmapFile () 35](#_Toc438038412)

[5.1.1.62. メソッドparseHeader () 35](#_Toc438038413)

[5.1.1.63. メソッドgoNextLine () 35](#_Toc438038414)

[5.1.1.64. メソッドgoBeginLine () 35](#_Toc438038415)

[5.1.1.65. メソッドfindComment () 35](#_Toc438038416)

[5.1.1.66. メソッドskipComment () 35](#_Toc438038417)

[5.1.1.67. メソッドcheckWord () 35](#_Toc438038418)

[5.1.1.68. メソッドcheckContinuousStatement () 36](#_Toc438038419)

[5.1.1.69. メソッドskipStatement () 36](#_Toc438038420)

[5.1.1.70. メソッドpushState () 36](#_Toc438038421)

[5.1.1.71. メソッドpopState () 36](#_Toc438038422)

[5.1.1.72. メソッドresetState () 36](#_Toc438038423)

[5.1.1.73. メソッドisEmpyState () 37](#_Toc438038424)

[5.1.1.74. メソッドgetBlock () 37](#_Toc438038425)

[5.1.1.75. メソッドgetPreamble () 37](#_Toc438038426)

[5.1.1.76. メソッドgetPostamble () 37](#_Toc438038427)

[5.1.1.77. メソッドparseCode () 37](#_Toc438038428)

[5.1.1.78. メソッドsortedBlockMap () 37](#_Toc438038429)

[5.1.1.79. メソッドoutputMacro () 37](#_Toc438038430)

[5.1.1.80. メソッドoutputCode () 38](#_Toc438038431)

[5.1.1.81. メソッドoutputBlock () 38](#_Toc438038432)

[5.1.1.82. メソッドusage () 38](#_Toc438038433)

[5.1.1.83. メソッドmain() 38](#_Toc438038434)

[5.1.2. Blockクラス 39](#_Toc438038435)

[5.1.2.1. Blockコンストラクタ 39](#_Toc438038436)

[5.1.2.2. メンバblock 39](#_Toc438038437)

[5.1.2.3. メンバsystem 39](#_Toc438038438)

[5.1.2.4. メンバname 39](#_Toc438038439)

[5.1.2.5. メンバblockName 39](#_Toc438038440)

[5.1.2.6. メンバsystemId 39](#_Toc438038441)

[5.1.2.7. メンバblockCount 39](#_Toc438038442)

[5.1.2.8. メンバupdateCount 40](#_Toc438038443)

[5.1.2.9. メンバid 40](#_Toc438038444)

[5.1.2.10. メソッドgetBlock() 40](#_Toc438038445)

[5.1.2.11. メソッドgetSystem() 40](#_Toc438038446)

[5.1.2.12. メソッドgetName() 40](#_Toc438038447)

[5.1.2.13. メソッドgetBlockName() 40](#_Toc438038448)

[5.1.2.14. メソッドgetSystemId() 40](#_Toc438038449)

[5.1.2.15. メソッドgetBlockCount() 41](#_Toc438038450)

[5.1.2.16. メソッドincrementBlockCount() 41](#_Toc438038451)

[5.1.2.17. メソッドgetUpdateCount() 41](#_Toc438038452)

[5.1.2.18. メソッドincrementUpdateCount() 41](#_Toc438038453)

[5.1.2.19. メソッドsetId() 41](#_Toc438038454)

[5.1.2.20. メソッドgetId() 41](#_Toc438038455)

[5.1.3. Rangeクラス 42](#_Toc438038456)

[5.1.3.1. Rangeコンストラクタ 42](#_Toc438038457)

[5.1.3.2. メンバstart 42](#_Toc438038458)

[5.1.3.3. メンバend 42](#_Toc438038459)

[5.1.3.4. メソッドgetStart() 42](#_Toc438038460)

[5.1.3.5. メソッドgetEnd() 42](#_Toc438038461)

[5.1.4. PointBaseクラス 43](#_Toc438038462)

[5.1.4.1. PointBaseコンストラクタ 43](#_Toc438038463)

[5.1.4.2. メンバtype 43](#_Toc438038464)

[5.1.4.3. メソッドgetType() 43](#_Toc438038465)

[5.1.4.4. メソッドgetPoint() 43](#_Toc438038466)

[5.1.4.5. メソッドcompareTo() 43](#_Toc438038467)

[5.1.5. PointBaseComparatorクラス 44](#_Toc438038468)

[5.1.5.1. メソッドcompare() 44](#_Toc438038469)

[5.1.6. SkipPointクラス 45](#_Toc438038470)

[5.1.6.1. SkipPointコンストラクタ 45](#_Toc438038471)

[5.1.7. ReplacePointクラス 46](#_Toc438038472)

[5.1.7.1. ReplacePointコンストラクタ 46](#_Toc438038473)

[5.1.7.2. メンバreplace 46](#_Toc438038474)

[5.1.7.3. メソッドgetReplace() 46](#_Toc438038475)

[5.1.8. InsertPointクラス 47](#_Toc438038476)

[5.1.8.1. InsertPointコンストラクタ 47](#_Toc438038477)

[5.1.8.2. メンバinsertText 47](#_Toc438038478)

[5.1.8.3. メソッドgetInsertText() 47](#_Toc438038479)

[5.1.9. DeletePointクラス 48](#_Toc438038480)

[5.1.9.1. DeletePointコンストラクタ 48](#_Toc438038481)

[5.1.10. MeasurePointクラス 49](#_Toc438038482)

[5.1.10.1. MeasurePointコンストラクタ 49](#_Toc438038483)

[5.1.10.2. メンバblock 49](#_Toc438038484)

[5.1.10.3. メンバupdate 49](#_Toc438038485)

[5.1.10.4. メソッドgetBlock() 49](#_Toc438038486)

[5.1.10.5. メソッドisUpdate() 49](#_Toc438038487)

[5.1.11. TIDInfoクラス 50](#_Toc438038488)

[5.1.11.1. TIDInfoコンストラクタ 50](#_Toc438038489)

[5.1.11.2. メンバtidIndex 50](#_Toc438038490)

[5.1.11.3. メンバstartOfCondition 50](#_Toc438038491)

[5.1.11.4. メンバendOfCondition 50](#_Toc438038492)

[5.1.11.5. メソッドgetTidIndex() 50](#_Toc438038493)

[5.1.11.6. メソッドgetStartOfCondtion() 50](#_Toc438038494)

[5.1.11.7. メソッドgetEndOfCondition() 50](#_Toc438038495)

[5.1.12. SaveStateクラス 51](#_Toc438038496)

[5.1.12.1. SaveStateコンストラクタ 51](#_Toc438038497)

[5.1.12.2. メンバstartComment 51](#_Toc438038498)

[5.1.12.3. メンバendComment 51](#_Toc438038499)

[5.1.12.4. メンバstartStatement 51](#_Toc438038500)

[5.1.12.5. メンバendStatement 51](#_Toc438038501)

[5.1.12.6. メンバstartBlock 52](#_Toc438038502)

[5.1.12.7. メンバendBlock 52](#_Toc438038503)

[5.1.12.8. メンバtidInfo 52](#_Toc438038504)

[5.1.12.9. メンバparsePoint 52](#_Toc438038505)

[5.1.12.10. メンバcodeEnd 52](#_Toc438038506)

[5.1.12.11. メンバstartBlockComment 52](#_Toc438038507)

[5.1.12.12. メンバcurrentBlock 52](#_Toc438038508)

[5.1.12.13. メンバisUpdate 52](#_Toc438038509)

[5.1.12.14. メンバhasIf 53](#_Toc438038510)

[5.1.12.15. メンバpreamble 53](#_Toc438038511)

[5.1.12.16. メンバpostamble 53](#_Toc438038512)

[5.1.12.17. メンバrestart 53](#_Toc438038513)

[5.2. SetMeasuringResultのクラス 54](#_Toc438038514)

[5.2.1. SetMeasuringResultクラス 56](#_Toc438038515)

[5.2.1.1. 静的メンバverbose 56](#_Toc438038516)

[5.2.1.2. 静的メンバblockResultMap 56](#_Toc438038517)

[5.2.1.3. 静的メソッドloadCSV() 56](#_Toc438038518)

[5.2.1.4. 静的メソッドprocess() 56](#_Toc438038519)

[5.2.1.5. 静的メソッドusage() 56](#_Toc438038520)

[5.2.1.6. 静的メソッドmain() 56](#_Toc438038521)

[5.2.2. MeasuringResultクラス 57](#_Toc438038522)

[5.2.2.1. コンストラクタMeasuringResult() 57](#_Toc438038523)

[5.2.2.2. メンバblockName 57](#_Toc438038524)

[5.2.2.3. メンバresult 57](#_Toc438038525)

[5.2.2.4. メンバcoreNumber 57](#_Toc438038526)

[5.2.2.5. メンバused 57](#_Toc438038527)

[5.2.2.6. メソッドgetBlockName() 57](#_Toc438038528)

[5.2.2.7. メソッドgetResult() 57](#_Toc438038529)

[5.2.2.8. メソッドgetCoreNumber() 58](#_Toc438038530)

[5.2.2.9. メソッドsetUsed() 58](#_Toc438038531)

[5.2.2.10. メソッドisUsed() 58](#_Toc438038532)

[6. RTW\_Measurement(CForest実行環境) 59](#_Toc438038533)

[6.1. measurement\_cforestの構成 60](#_Toc438038534)

[6.1.1. readme.txt 60](#_Toc438038535)

[6.1.2. MyEnv.sh(要変更) 60](#_Toc438038536)

[6.1.3. Tepllate-user.mk(要コピー・変更) 60](#_Toc438038537)

[6.1.4. Makefile(要コピー・変更) 61](#_Toc438038538)

[6.1.5. measurement.mk 61](#_Toc438038539)

[6.1.6. changedir.mk 61](#_Toc438038540)

[6.1.7. run.sh 61](#_Toc438038541)

[6.1.8. cforest.sh 61](#_Toc438038542)

[6.1.9. gdb.sh 61](#_Toc438038543)

[6.1.10. lib 62](#_Toc438038544)

[6.1.10.1. hwdep.c 62](#_Toc438038545)

[6.1.10.2. hwdep.h 62](#_Toc438038546)

[6.1.10.3. boot.s 62](#_Toc438038547)

[6.1.10.4. sample.ld 62](#_Toc438038548)

[6.1.11. include 62](#_Toc438038549)

[6.1.11.1. measure\_cforest.h 62](#_Toc438038550)

[6.1.12. sub 63](#_Toc438038551)

[6.1.12.1. gdb-command.x 63](#_Toc438038552)

[6.1.12.2. gdb.py 63](#_Toc438038553)

**図目次**

[図 2‑1 プログラムと入出力ファイル 3](#_Toc437884014)

[図 2‑2　ディレクトリ構成 5](#_Toc437884015)

**表目次**

[表 2‑1 ソフトウェア構成 3](#_Toc437884016)

[表 2‑2 ツール・ライブラリ一覧 4](#_Toc437884017)

[表 5‑1 AddMeasurementクラス 16](#_Toc437884018)

[表 5‑2 AddMeasureクラス内クラスの一覧 20](#_Toc437884019)

[表 5‑3 SetMeasuringResultクラス 54](#_Toc437884020)

[表 5‑4 SetMeasuringResultクラス内クラス一覧 55](#_Toc437884021)

[表 6‑1 RTW\_measurementのファイル構成 59](#_Toc437884022)

[表 6‑2 user.mkの変数 60](#_Toc437884023)

[表 6‑3 Makefileの変数 61](#_Toc437884024)

**参考資料**

|  |  |  |  |
| --- | --- | --- | --- |
| 項番 | 仕様書名 | 版数 | 発行元 |
|  |  |  |  |
|  |  |  |  |

**用語一覧**

|  |  |  |
| --- | --- | --- |
| 項番 | 用語 | 意味 |
| 1 | ブロックレベル構造XML  BLXML | Simulinkモデルから抽出したブロック構造を持つXMLファイル(名大様提供) |

# はじめに

本書では、以下の機能について説明します。

1. AddMeasurement
   * Mathworks 社ツールによって生成されたCコードを入力し、Simulink ブロックごとの性能を測定するターゲット非依存のプログラムの自動生成
2. RTW\_Measurement
   * 上記の生成プログラムをRenesas社CForest向けSDKでのビルド、およびCForestシミュレータにより計測する環境
3. SetMeasuringResult
   * 計測結果(CSV形式)を読み込み名古屋大学様のBLXMLのブロックに、計測結果を反映する

# ソフトウェア構成

　ソフトウェア構成は以下の通りです

表 2‑1 ソフトウェア構成

|  |  |  |
| --- | --- | --- |
| 項番 | 名称 | 説明 |
| 1 | AddMeasurement | MathWorks社ツールが生成したCコードから計測用Cコードを自動生成する |
| 2 | RTW\_Measurement | 計測用CコードからCForestシミュレータ用プログラムをビルドし、CForestシミュレータで実行し、計測結果(CSV)を出力する |
| 3 | SetMeasuringResult | 計測結果(CSV)と、入力BLXMLを入力し、計測結果をブロックに反映し、出力BLXMLへ出力する |
| 4 | Cコード | MathWorks社ツールが生成したCコード |
| 5 | Cヘッダ | MathWorks社ツールが生成したCヘッダ |
| 6 | 計測用Cコード | AddMeasurementが生成した計測用のCコード |
| 7 | 計測用マクロ | AddMeasurementが生成した計測用マクロ |
| 8 | 計測結果(CSV) | RTW\_Measurement機能により出力された計測結果のCSV形式のファイル |
| 9 | 入力BLXML | 計測結果を反映したい入力BLXML |
| 10 | 出力BLXML | 計測結果を反映した出力BLXML |

Cコード

AddMeasurement

計測用

Cコード

RTW\_Measurement

入力

BLXML

SetMeasuringResult

出力

BLXML

計測結果

(CSV)

計測用

マクロ

Cヘッダ

図 2‑1 プログラムと入出力ファイル

## 実行環境

　本プログラムはJava SE(JDK8)が実行可能なプラットフォームで動作します。また、ビルドおよび実行については表 2‑2のツール・ライブラリの最新版が必要です。

表 2‑2 ツール・ライブラリ一覧

|  |  |  |  |
| --- | --- | --- | --- |
| No. | 名称 | 説明 | 入手先 |
| 1 | JDK8 | Java開発環境 | <http://www.oracle.com/technetwork/jp/java/javase/overview/index.html> |
| 2 | Apache Ant | ビルドツール | <http://ant.apache.org/> |

## ファイル一覧

　以下はディレクトリ構成とファイルの一覧です

Simulink\_XSD/java

│ build.xml

│ SetMeasuringResult.sh

│ AddMeasurement.sh

│

├─src

│ └─jp

│ └─ertl

│ ├─simulink

│ │ └─BLXML

│ │ SchemaDefine.java

│ │

│ └─utils

│ Getopts.java

│ SimpleCSV.java

│

├─samples

│ └─RTW\_Measurement

│ ├─measurement\_cforest

│ │ │ run.sh

│ │ │ Makefile

│ │ │ readme.txt

│ │ │ gdb.sh

│ │ │ MyEnv.sh

│ │ │ cforest.sh

│ │ │ measurement.mk

│ │ │ Template-user.mk

│ │ │ changedir.mk

│ │ │

│ │ ├─lib

│ │ │ hwdep.c

│ │ │ sample.ld

│ │ │ hwdep.h

│ │ │ boot.s

│ │ │

│ │ ├─include

│ │ │ measure\_cforest.h

│ │ │

│ │ └─sub

│ │ gdb-command.x

│ │ gdb.py

│ │

│ ├─sample\_c\_gmres

│ │

│ └─sample\_synrmmodel

│

├─AddMeasurement

│ AddMeasurement.java

│

├─SetMeasuringResult

│ SetMeasuringResult.java

│

└─schema

readme-schema.txt

jaxb-prop.xjb

図 2‑2　ディレクトリ構成

* build.xmlはantでビルドするための設定ファイルです。ただし、本ドキュメント外の機能と共用しているため、本ドキュメント外の機能のビルド設定も含まれます
* 実際にビルドする際にはSimulinkModel.xsdをschemaに配置する必要があります。schema\readme-schema.txtはその説明です
* jaxb-prop.xjbはJAXBにより生成されるJavaコードのパッケージ名の指定と、JAXBで衝突が発生するXMLの属性の”value”をValueAttributeに変更するために用います。なおパッケージ名はjp.ertl.simulink.SimulinkModelです
* 実際にRTW\_Measurementでの計測を行う場合は、Renesas社のCForest向けのSDKと、CForest MPシミュレータが別途必要です

# ビルド手順

## ビルド準備

### JDK8のインストール

最新のJDK8をダウンロードサイトからダウンロードし、インストールしてください。

<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

* インストールしたJDKのパスをJAVA\_HOME環境変数に適切に設定してください
* 環境変数PATHにjavac,xjcが実行可能なように設定してください。具体的には $JAVA\_HOME/binをPATHに追加設定します

### Apache Antのインストール

最新のantをダウンロードし、インストールしてください。

<https://ant.apache.org/bindownload.cgi>

antが実行できるようantコマンドがあるディレクトリを環境変数にPATHに追加設定してください。

### SimulinkModel.xsdの配置

最新の状態のSimulink\_XSD/SimulinkModel.xsdをschemaにコピーしてください

## ビルド

Simulink\_XSD/javaのディレクトリにて以下のようにantを実行してください。

$ ant

ただし、本ドキュメント外の機能のビルドも含まれるため、本ドキュメント外の機能のビルドも同時に行われます。

## javadocの生成(オプショナル)

ドキュメント(javadoc)を生成するには以下のようにantを実行してください。

$ ant javadoc

javadoc以下にHTMLファイルが生成されますので、内容については、任意のWeb/HTMLブラウザで参照してください。なお、本ドキュメント外の機能のビルドも含まれるため、本ドキュメント外の機能のビルドも同時に行われます。

# 実行

## AddMeasurementコマンド仕様(javaコマンド使用時)

　javaコマンドにて実行する場合、以下のように指定します。

$ java –cp “***クラスパス***” AddMeasurement [-v] [-p マクロプレフィクス] \

[-b ブロック情報ファイル] [-i インクルードファイル] \

[-o ***計測用Cコード***] [-m ***計測用マクロ***]-h ***Cヘッダ*** ***Cコード***

* ***クラスパス***は次のものを指定する必要があります。
  + JAXBにより生成され、コンパイルされたSimulinkModelのclassがあるディレクトリ
  + AddMeasurement.classがあるディレクトリ
* **-v**オプションは実行時に様々な情報を標準出力に表示します。このため、出力ファイルを指定しない(**-o**オプションを指定しない)場合には指定していても無視します
* **-p**オプションは、AddMeasurementが出力するCのマクロのプレフィクスを指定します。デフォルトでは空文字列です
* **-b**オプションは、ブロックの情報をCSVファイルに出力します。指定しない場合は出力しません。このファイルの用途は、デバッグやブロックの情報を他の目的に使用することです。「4.4AddMeasurementが出力するブロック情報のファイル形式」を参照してください。
* **-i**オプションは**計測用Cコード**に出力するインクルードファイルを指定します。複数指定可能で、計測用Cコードに指定した順序でインクルード文を出力します。「4.3AddMeasurementによる計測Cコードのマクロ」で述べるターゲット依存マクロを定義するヘッダファイルはこのオプションを指定してCコードにヘッダファイルのインクルード文を出力することを推奨します
* **-o**オプションは出力する**計測用Cコード**を指定します。**-o**オプションを指定しない場合、標準出力に出力します。**-v**オプションを指定する場合は、**-o**オプションの使用を推奨します
* **-m**オプションは、**計測用マクロ**を指定します。**-m**オプションを指定しない場合は”macro.h”がデフォルトです。このマクロファイル名は、計測用Cコードには最初にインクルードされるように自動生成されます
* **-h**オプションはMathWorks社のツールが生成したCのヘッダファイルを指定します。ブロックの階層情報を読み込むため、このオプションは必須です
* ***Cコード***はMathWorks社のツールが生成したCのコードです。この指定は必須です

## AddMeasurement.shコマンド仕様(shellスクリプト使用時)

AddMeasurement.shは「4.1AddMeasurementコマンド仕様(javaコマンド使用時)」を簡易に実行するためのスクリプトなので、オプションは同じです。

$ AddMeasuremen.sht [-v] [-p マクロプレフィクス] \

[-b ブロック情報ファイル] [-i インクルードファイル] \

[-o ***計測用Cコード***] [-m ***計測用マクロ***]-h ***Cヘッダ*** ***Cコード***

なお、本コマンドは、実行時にコマンドが配置されているディレクトリから相対的なファイル参照を行うため、次のようなディレクトリ構造である必要があります。

<AddMeasurement.shのあるディレクトリ>

├ AddMeasurement.sh

└ build

├ AddMeasurement.class

├ (AddMeasurement内クラスのclassファイル)

　 └ (SimulinkModelのclassファイル)

## AddMeasurementの制限事項

AddMeasurementはCコード中のSimulinkブロックのコードに対し、以下の制限があります。

* 単独のif文やfor文に単一のCの構文ブロックがあり、その構文ブロック内に、複数のSimulinkブロックのコードが含まれる場合は、含まれるSimulinkブロックそれぞれをif文やfor文を前置したコードに変換・分離し、計測コードとします。なお、else部があるといった、複数の構文ブロックを持つ場合や、後述のDot Productに関わるfor文の場合、あるいはswitch-case文のように、Simulinkブロックの分離が困難な構文ブロック内のSimulinkブロックは計測対象となりません
* Iterator SubSystemは、そのSimulinkブロックの終端を示すコメントまでをSubSystemのコードとして計測します。Iterator SubSystemに含まれる個々のブロックは計測対象になりません
* Dot Productは、そのSimulinkブロックの終端を示すコメントまでをDot Productのコードとして計測します。なお、for文内に他のSimulinkブロックが含まれていてもそのブロックは計測対象になりません

## AddMeasurementによる計測Cコードのマクロ

### ブロック番号

ブロック番号のマクロは、以下のようにBLXMLのブロック名を大文字としたマクロ名です。0から昇順に割り当て、「4.3.2最大ブロック番号」のMAX\_BLOCK\_ID未満の範囲です。このマクロは、「4.3.10step関数の開始マクロ」および「4.3.11step関数の終端マクロ」のマクロの第1パラメタに使用され、4.3.4ブロック名の配列のインデクスにも使用可能です。

#define SYNRMMODEL\_MT\_RTW\_MAKETARGETVALUE1 0

このマクロ名を他の目的の機能から参照したい場合、「4.4AddMeasurementが出力するブロック情報のファイル形式」が有用です。

### 最大ブロック番号

最大ブロック番号のマクロは以下の通りです。

/\* Maximum Block ID \*/

#define MAX\_BLOCK\_ID 157

### ブロック固有パラメタ(ターゲット依存)

任意のブロック固有のパラメタを設定可能です。パラメタは、BLXMLのブロック名を大文字とし、”\_PARAM”をサフィックスとしたものです。このマクロは、「4.3.10ブロックの計測開始マクロ」および4.3.11ブロックの計測停止マクロのマクロの第4パラメタに使用されます。デフォルトでは0です。

/\* Block Parameters \*/

#ifndef SYNRMMODEL\_MT\_RTW\_MAKETARGETVALUE1\_PARAM

#define SYNRMMODEL\_MT\_RTW\_MAKETARGETVALUE1\_PARAM 0

#endif

本マクロは、-iオプションによって指定したヘッダファイルで定義することを推奨します。

### ブロック名の配列

BLXMLのブロック名の文字列の配列です。配列のインデクスは、「4.3.1ブロック番号」のマクロか、「4.3.10ブロックの計測開始マクロ」および「4.3.11ブロックの計測停止マクロ」の第1パラメタを用います。

/\*

\* Block Names

\*/

const char \*BLOCK\_NAMES[];

### step関数およびinitialize関数名のマクロ

step関数とinitialize関数は、マクロ定義されます。このマクロを使うことで、モデルに依存しない計測環境を作成できます。

#define MEASUREMENT\_STEP\_FUNCTION synrmmodel\_mt\_RTW\_step

#define MEASUREMENT\_INITIALIZE\_FUNCTION synrmmodel\_mt\_RTW\_initialize

### if文の条件式のラップマクロ(ターゲット依存)

ブロックが例えばTriggered SubSystemのブロックの場合、トリッガ―により実行制御されるので、計測のタイミングではブロックのコードが実行されない場合があります。そのため、AddMeasurementでは、その実行判定のif文の条件を制御できるようにするため、条件式をマクロでラップして出力します。マクロが定義されない場合、以下の定義が使用され、ラップされた条件式がそのまま使用されます。

#ifndef MEASUREMENT\_IF\_CONDITION

#define MEASUREMENT\_IF\_CONDITION(X) X

#endif

例えば、本マクロ定義を1にすれば、Triggered SubSystemのブロックであっても、常にブロックのコードの実行を計測することが可能です。

本マクロは、-iオプションによって指定したヘッダファイルで定義することを推奨します。

### TIDの条件式のラップマクロ(ターゲット依存)

ブロックがベースレートと異なるサンプル時間を持つ場合、サンプル時間のタイミングで実行制御されるので、計測のタイミングではブロックのコードが実行されない場合があります。そのため、AddMeasurementでは、その実行判定のif文の条件を制御できるようにするため、条件式をマクロでラップして出力します。マクロが定義されない場合、以下の定義が使用され、ラップされた条件式がそのまま使用されます。

#ifndef MEASUREMENT\_TID\_CONDITION

#define MEASUREMENT\_TID\_CONDITION(X) X

#endif

例えば、本マクロ定義を1にすれば、ベースレートと異なるサンプル時間のブロックであっても、常にブロックのコードの実行を計測することが可能です。

本マクロは、-iオプションによって指定したヘッダファイルで定義することを推奨します。

### step関数の開始マクロ(ターゲット依存)

step関数の実行開始位置に生成されるマクロです。デフォルトでは何もしないマクロとなります。

#ifndef MEASUREMENT\_STEP\_BEGIN

#define MEASUREMENT\_STEP\_BEGIN() do { ; } while (0)

#endif

step関数の開始時に行う処理を記述します。例えば、計測コードの計測オーバヘッドや、実行カウンタの更新、計測処理の初期化などです。

本マクロは、-iオプションによって指定したヘッダファイルで定義することを推奨します。

### step関数の終端マクロ(ターゲット依存)

step関数の実行終了位置に生成されるマクロです。デフォルトでは何もしないマクロとなります。

#ifndef MEASUREMENT\_STEP\_END

#define MEASUREMENT\_STEP\_END() do { ; } while (0)

#endif

step関数の終了時に処理を記述します。例えば、計測結果全体に対する処理や、実行カウンタの更新、計測結果のコミットなどです。

本マクロは、-iオプションによって指定したヘッダファイルで定義することを推奨します。

### ブロックの計測開始マクロ(ターゲット依存)

ブロックのコードの計測開始位置に生成されるマクロです。デフォルトでは何もしないマクロとなります。

#ifndef MEASUREMENT\_START

#define MEASUREMENT\_START(ID,UPD,INDEX,PARAM) do { ; } while (0)

#endif

**ID**は「4.3.1ブロック番号」のブロック番号、**UPD**はコードがアップデートであれば1、そうでなければ0です。**INDEX**は0から始まるブロックのコードの計測位置を示し、ブロックのコードが異なる場所に出現していることを識別します。**PARAM**は「4.3.3ブロック固有パラメタ」として任意に設定可能です。

なお、コンパイル時の最適化により、計測コードの一部または全部が本マクロを超えて巻き上げられることのないようにする必要があります。具体的にはインライン展開を禁止した関数にするなど、コンパイラに副作用があることを認識させます。

本マクロは、-iオプションによって指定したヘッダファイルで定義することを推奨します。

### ブロックの計測停止マクロ(ターゲット依存)

ブロックのコードの計測終了位置に生成されるマクロです。デフォルトでは何もしないマクロとなります。パラメタについては「4.3.10ブロックの計測開始マクロ」と同じです。

#ifndef MEASUREMENT\_STOP

#define MEASUREMENT\_STOP(ID,UPD,INDEX,PARAM) do { ; } while (0)

#endif

なお、コンパイル時の最適化により、計測コードの一部または全部が本マクロを超えて後置されることがないようにする必要があります。具体的にはインライン展開を禁止した関数にするなど、コンパイラに副作用があることを認識させます。

本マクロは、-iオプションによって指定したヘッダファイルで定義することを推奨します。

## AddMeasurementが出力するブロック情報のファイル形式

ブロック情報のファイル形式は以下のCSV形式のファイルです。このファイルはAddMeasurementの-bオプションで指定します。

ブロックヒエラルキー,ブロックID,BLXMLブロック名,ブロックIDのマクロ名

**ブロックヒエラルキー**は、”<Root>/A”や”<S11>/B”といったブロックのヒエラルキーです。**ブロックID**および**ブロックID**のマクロ名はブロックに割り振られた「4.3.1ブロック番号」の値とマクロ名です。**BLXMLブロック名**はBLXMLにおけるブロック名です。

このファイルは、計測Cコード内のブロックのIDやブロックIDのマクロ名をヒエラルキーのブロックやBLXMLのブロック名とを関連付けし、外部機能での計測用マクロの生成やデバッグ、その他の目的に使用することができます。

## SetMeasuringResultコマンド仕様(javaコマンド使用時)

javaコマンドにて実行する場合、以下のように指定します。

$ java –cp “***クラスパス***” SetMeasuringResult [-v] \

-i ***入力BLXML*** [-o ***出力BLXML***] [***計測結果CSV***]

* ***クラスパス***は次のものを指定する必要があります。
  + JAXBにより生成され、コンパイルされたSimulinkModelのclassがあるディレクトリ
  + SetMeasuringResult.classがあるディレクトリ
* **-v**オプションは実行時に様々な情報を標準出力に表示します。このため、出力ファイルを指定しない(**-o**オプションを指定しない)場合には指定していても無視します
* **-i**オプションは、***入力BLXML***を指定します。これは必須オプションです
* **-o**オプションは、***出力BLXML***を指定します。**-o**オプションを指定しない場合、標準出力に出力します

***計測結果CSV***は「4.7SetMeasuringResult計測結果CSVファイル」の形式の計測結果のCSVファイルです。***計測結果CSV***が省略された場合、標準入力から入力します

## SetMeasuringResultコマンド仕様(shellスクリプト使用時)

SetMeasuringResult.shは「4.5SetMeasuringResultコマンド仕様(javaコマンド使用時)」を簡易に実行するためのスクリプトなので、オプションは同じです。

$ SetMeasuringResult.sh [-v] -i ***入力BLXML*** [-o ***出力BLXML***] ***計測結果CSV***

なお、本コマンドは、実行時にコマンドが配置されているディレクトリから相対的なファイル参照を行うため、次のようなディレクトリ構造である必要があります。

<SetMeasuringResult.shのあるディレクトリ>

├ SetMeasuringResult.sh

└ build

├ SetMeasuringResult.class

├ (SetMeasuringResult内クラスのclassファイル)

　 └ (SimulinkModelのclassファイル)

## SetMeasuringResult計測結果CSVファイル

SetMeasuringResultの入力の計測結果ファイル形式は以下のCSV形式のファイルです。このファイルはSetMeasuringResultのコマンド引数で指定します。

***BLXMLブロック名***,***計測結果***,***コア番号***

***BLXMLブロック名***はBLXMLのブロック名です。***計測結果***はBLXMLのブロックの属性”measuringResult”に設定する値です。***コア番号***はBLXMLのブロックの属性”peinfo”に設定する値です。

***BLXMLブロック名***が空の場合は、計測結果は計測のオーバヘッドを示します。また、***計測結果***が空の場合は、そのブロックを計測していないことを示し、属性”measuringResult”はBLXMLに出力されません。***コア番号***が空の場合は-1が指定されいるとみなします。

# モジュール仕様

## AddMeasurementのクラス

AddMeasurement.java内で定義されるクラスは表 5‑1、AddMeasurementクラスの内部クラスは表 5‑2の通りです。

表 5‑1 AddMeasurementクラス

|  |  |  |
| --- | --- | --- |
| クラス名 | 内容 | |
| AddMeasurement | Cコードから計測用のコードを生成するクラス | |
| 列挙 | 説明 |
| SubsystemType | SubSystemのタイプ |
| PointType | テキストの変更ポイント処理タイプ |
| メンバ | 説明 |
| verbose | 実行時のverboseフラグ |
| outputMacroName | 出力マクロファイル名 |
| macroPrefix | 出力するマクロのプレフィクス |
| includeFiles | インクルードファイル名のリスト |
| beginHierarchy | ヘッダファイルのヒエラルキー情報開始の正規表現パターン |
| blockHierarchy | ヘッダファイルのブロックのヒエラルキーの正規表現パターン |
| stepFunction | step関数の正規表現パターン |
| tidPattern | if文中のTIDの正規表現パターン |
| ifPattern | if文にマッチするための正規表現パターン |
| blockMark | Cコード中のブロックのコメントの正規表現パターン |
| indexEndOf | blockMarkのパターンにマッチしたMatcherの”End Of”のインデクス |
| indexUpdateOf | 同、”Update for”のインデクス |
| indexType | 同、Blockの前置タイプのインデクス |
| indexSfUpdate | 同、S-Functionのアップデートパタンのインデクス |
| indexSubSys | 同、SubSystemのインデクス |
| indexBlock | 同、ブロックのヒエラルキーのインデクス |
| indexSystem | 同、ヒエラルキーのブロックの前半のSubSystem部のインデクス |
| indexCCB | 同、”ConcatBufferAt”のインデクス |
| indexSC | 同、”TmpSignal ConversionAt”のインデクス |
| indexName | 同、ヒエラルキーのブロックの後半のブロック名のインデクス |
| blockEndMark | Cコード中のブロックのコードの終端の正規表現パターン |
| hierarchyMap | ヒエラルキーのSubSystemをキーとしたブロック名の連想配列 |
| blockNameMap | ヒエラルキーのブロック名をキーとしたBLXMLのブロック名の連想配列 |
| blockMap | ヒエラルキーのブロック名をキーとしたBlockのインスタンスの連想配列 |
| measurePointList | 計測および変更ポイントのリスト |
| startComment | 次に処理するコメントの開始位置 |
| endComment | 次に処理するコメントの終端位置 |
| startStatement | 次に処理する文の開始位置 |
| endStatement | 次に処理する文の終端位置 |
| startBlock | ブロックに依存しないCの構文ブロックの開始位置 |
| endBlock | ブロックに依存しないCの構文ブロックの終端位置 |
| tidInfo | 参照するTIDInfo |
| parsePoint | パース中のテキスト位置 |
| codeEnd | 処理中のテキストの終端位置 |
| startBlockComment | ブロックのコメントの開始位置 |
| currentBlock | 処理中のブロック情報 |
| isUpdate | コードがアップデートかいなか |
| hasIf | ifを持つブロックか(Trigger等) |
| preamble | 文の前に必要なコード |
| postamble | 文の後に必要なコード |
| restart | 再開するためのフラグ |
| stateStack | パース中の状態スタック |
| stepFunctionName | ステップ関数の関数名 |
| initializeFunctionName | 初期化関数の関数名 |
| stepFunctionMacroName | ステップ関数名に対応するマクロ名 |
| initializeFunctionMacroName | 初期化関数名に対応するマクロ名 |
| ifCondtionMacroName | if文の条件をラップするマクロ名 |
| tidCondtionMacroName | TIDの条件をラップするマクロ名 |
| startMacroName | ブロック計測開始のマクロ名 |
| stopMacroName | ブロック計測終了のマクロ名 |
| stepBeginMacroName | ステップ関数開始のマクロ名 |
| stepEndMacroName | ステップ関数終端のマクロ名 |
| メソッド | 説明 |
| addSkipPoint() | スキップする範囲を登録する |
| addReplacePoint() | 置換する範囲とテキストを登録する |
| addInsertPoint() | 挿入するポイントとテキストを登録する |
| addDeletePoint() | 削除範囲を登録する |
| addMeasurePoint() | 計測ポイントと計測のタイプ、ブロック情報、アップデートかどうかを登録する |
| canonicalize() | BLXMLで使用するブロック名に正規化する |
| mapFile() | ファイルの内容をマッピングしCharBufferを返す |
| parseHeader() | ヘッダファイルを処理し、ヒエラルキーのブロックの情報を取得する |
| goNextLine() | 次の行の位置を取得する |
| goBeginLine() | 行頭の位置を取得する |
| findComment() | コメントの位置を取得する |
| skipComment() | コメントを読み飛ばした位置を取得する |
| checkWord() | 単語が出現する場合単語の終わりを取得する |
| checkContinuousStatement() | elseやwhileかどうか調べる |
| skipStatement() | 文を読み飛ばした位置を取得する |
| pushState() | パース中の状態をセーブする |
| popState() | パース中の状態を戻す |
| resetState() | パース中の状態を初期化する |
| isEmpyState() | パース中の状態のスタックが空か調べる |
| getBlock() | blockMarkにマッチしたコメント中のブロック情報を取得する |
| getPreamble() | パース状態のスタックから、現在の文の前に必要なコードを取得する |
| getPostamble() | パース状態のスタックから、現在の文の後に必要なコードを取得する |
| parseCode() | Cコードをパースする |
| sortedBlockMap() | blockMapをヒエラルキーおよびブロックの名前でソートしたリストを返す |
| outputMacro() | 計測用マクロを出力する |
| outputCode() | 計測用Cコードを出力する |
| outputBlock() | ブロック情報を出力する |
| usage() | コマンドの使い方を表示し、exit(1)で終了する |
| main() | AddMeasurementのmain関数 |
| 内部クラス | 説明 |
| Block | Simulinkのブロックの情報を格納するクラス |
| Range | テキストの開始と終端の範囲を保持するスーパークラス |
| PointBase | テキストの変更ポイントのスーパークラス(Rangeを継承) |
| PointBaseComparator | PointBaseの比較用クラス |
| SkipPoint | スキップする範囲を保持するクラス(PointBaseを継承) |
| ReplacePoint | 置換する範囲を保持するクラス(pointBaseを継承) |
| InsertPoint | 文字列挿入を行うポイントを保持するクラス(PointBaseを継承) |
| DeletePoint | 削除を行う範囲を保持するクラス(PointBaseを継承) |
| MeasurePoint | 計測コードを出力するポイントを保持するクラス(PointBaseを継承) |
| TIDInfo | マルチレートのTIDの情報を保持するクラス |
| SaveState | Cコードのパース中の状態を保持するクラス |

表 5‑2 AddMeasureクラス内クラスの一覧

|  |  |  |
| --- | --- | --- |
| クラス名 | 内容 | |
| Block | Simulinkのブロックの情報を格納するクラス | |
| コンストラクタ | 説明 |
| Block | Blockのメンバの情報を指定したコンストラクタ |
| メンバ | 説明 |
| block | ブロック名(ヒエラルキー) |
| system | ヒエラルキーのブロック名の前半部 |
| name | ヒエラルキーのブロック名の後半部 |
| blockName | BLXMLのブロック名 |
| systemId | systemの数値 |
| blockCount | ブロックのコードのカウント |
| updateCount | ブロックのUpdateコードのカウント |
| id | ブロックに割り付けたID番号 |
| 公開メソッド | 説明 |
| getBlock() | ブロック名(ヒエラルキー)の取得 |
| getSystem() | ヒエラルキーのブロック名の前半部の取得 |
| getName() | ヒエラルキーのブロック名の後半部の取得 |
| getBlockName() | BLXMLのブロック名の取得 |
| getSystemId() | systemの数値の取得 |
| getBlockCount() | ブロックのコードのカウントの取得 |
| incrementBlockCount() | ブロックのコードのカウントのインクリメント |
| getUpdateCount() | ブロックのUpdateコードのカウントの取得 |
| incrementUpdateCount() | ブロックのUpdateコードのカウントのインクリメント |
| setId() | ブロックに割り付けたID番号の設定 |
| getId() | ブロックに割り付けたID番号の取得 |

|  |  |  |
| --- | --- | --- |
| Rage | テキストの開始と終端の範囲を保持するスーパークラス | |
| コンストラクタ | 説明 |
| Range(int,int) | 範囲の開始と終端を指定したコンストラクタ |
| メンバ | 説明 |
| start | 範囲の開始 |
| end | 範囲の終端 |
| 公開メソッド | 説明 |
| getStart() | 範囲の開始の取得 |
| getEnd() | 範囲の終端の取得 |
| PointBase | テキストの変更ポイントのスーパークラス(Rangeを継承) | |
| コンストラクタ | 説明 |
| PointBase | PointBaseの変更ポイントのタイプ、開始、終端を指定したコンストラクタ |
| メンバ | 説明 |
| type | 変更ポイントのタイプ |
| 公開メソッド | 説明 |
| getType() | 変更ポイントのタイプを取得 |
| getPoint() | 変更ポイントを取得 |
| compareTo() | ポイントの位置および変更処理優先度での比較 |
| PointBaseComparator | PointBaseの比較用クラス | |
| 公開メソッド | 説明 |
| compare() | 比較メソッド |
| SkipPoint | スキップする範囲を保持するクラス(PointBaseを継承) | |
| コンストラクタ | 説明 |
| SkipPoint(int,int) | スキップする開始位置と終端を指定したコンストラクタ |
| ReplacePoint | 置換する範囲を保持するクラス(pointBaseを継承) | |
| コンストラクタ | 説明 |
| ReplacePoint(int,int,String) | 置換する開始位置と終端、置換テキストを指定したコンストラクタ |
| メンバ | 説明 |
| replace | 置換テキスト |
| 公開メソッド | 説明 |
| getReplace() | 置換テキストの取得 |

|  |  |  |
| --- | --- | --- |
| InsertPoint | 文字列挿入を行うポイントを保持するクラス(PointBaseを継承) | |
| コンストラクタ | 説明 |
| InsertPoint(int,String) | 挿入位置と挿入テキストを指定したコンストラクタ |
| メンバ | 説明 |
| insertText | 挿入テキスト |
| 公開メソッド | 説明 |
| getInsertText() | 挿入テキストの取得 |
| DeletePoint | 削除を行う範囲を保持するクラス(PointBaseを継承) | |
| コンストラクタ | 説明 |
| DeletePoint(int,int) | 削除する範囲の開始と終端を指定したコンストラクタ |
| MeasurePoint | 計測コードを出力するポイントを保持するクラス(PointBaseを継承) | |
| コンストラクタ | 説明 |
| MeasurePoint(PointType,  int,Block,Boolean) | 計測のタイプ、位置、ブロック情報、アップデートかどうかを指定したコンストラクタ |
| メンバ | 説明 |
| block | 計測ポイントのブロック情報 |
| update | アップデートかどうかのフラグ |
| 公開メソッド | 説明 |
| getBlock() | ブロック情報の取得 |
| isUpdate() | アップデートかどうか検査 |
| TIDInfo | マルチレートのTIDの情報を保持するクラス | |
| コンストラクタ | 説明 |
| TIDInfo(int,int,int) | TIDのインデクスと、if文内の条件の開始・終端を指定したコンストラクタ |
| メンバ | 説明 |
| tidIndex | TIDのインデクス |
| startOfCondition | if文の条件式の開始位置 |
| endOfCondigion | if文の条件式の終端位置 |
| 公開メソッド | 説明 |
| getTidIndex() | TIDのインデクスの取得 |
| getStartOfCondtion() | if文の条件式の開始位置の取得 |
| getEndOfCondition() | if文の条件式の終端位置の取得 |

|  |  |  |
| --- | --- | --- |
| SaveState | Cコードのパース中の状態を保持するクラス | |
| コンストラクタ | 説明 |
| SaveState | SaveStateのメンバ情報を全部または一部を指定したコンストラクタ |
| メンバ | 説明 |
| startComment | 次に処理するコメントの開始位置 |
| endComment | 次に処理するコメントの終端位置 |
| startStatement | 次に処理する文の開始位置 |
| endStatement | 次に処理する文の終端位置 |
| startBlock | ブロックに依存しないCの構文ブロックの開始位置 |
| endBlock | ブロックに依存しないCの構文ブロックの終端位置 |
| tidInfo | 参照するTIDInfo |
| parsePoint | パース中のテキスト位置 |
| codeEnd | 処理中のテキストの終端位置 |
| startBlockComment | ブロックのコメントの開始位置 |
| currentBlock | 処理中のブロック情報 |
| isUpdate | コードがアップデートかいなか |
| hasIf | ifを持つブロックか(Trigger等) |
| preamble | 文の前に必要なコード |
| postamble | 文の後に必要なコード |
| restart | 再開するためのフラグ |

### AddMeasureクラス

Mathworks 社ツールによって生成されたCコードを入力し、Simulink ブロックごとの性能を測定するターゲット非依存のプログラムを自動生成する主処理のクラスです。

#### 列挙子SubsystemType

private static enum SubsystemType { NONE, ATOMIC, ENABLED, TRIGGERED,

ENABLED\_TRIGGERED, ACTION, ITERATOR };

SubSystemのタイプを表す列挙子です。

#### 列挙子PointType

private static enum PointType { INCLUDE,

STEP\_BEGIN,

START, INSERT, STOP, DELETE,

REPLACE, SKIP,

STEP\_END };

テキストの変更ポイントにおける処理タイプを表す列挙子です。この順番は処理の優先度も表します(先に出現しているものが優先されます)。

#### 静的メンバverbose

private static boolean verbose = false;

プログラム実行時のverboseメッセージの表示有無のフラグです。-vオプションを指定するとtrueになります。

#### 静的メンバoutputMacroName

private static String outputMacroName = "macro.h";

評価用マクロのファイル名です。-mオプションにより設定されます。

#### 静的メンバmacroPrefix

private static String macroPrefix = "";

出力するCのマクロ名のプレフィクスです。-pオプションにより設定されます。

#### 静的メンバincludeFiles

private static List<String> includeFiles = new ArrayList<String>();

評価用Cコードに出力するインクルードファイルのリストです。-iオプションにより設定されます。

#### 静的メンバbeginHierarchy

private static Pattern

beginHierarchy = Pattern.compile("^\\s\*\\\*\\s\*Here\\s+is\\s+the"

+ "\\s+system\\s+hierarchy\\s+"

+ "for\\s+this\\s+model\\s\*$",

Pattern.MULTILINE);

MathWorks社のツールが生成したヘッダファイルのSubsystemのヒエラルキーの情報の開始を示す正規表現パターンです。

#### 静的メンバblockHierarchy

private static Pattern

blockHierarchy = Pattern.compile("^\\s\*\\\*\\s\*("

+ "'(<(Root|S\\d+)>)'"

+ "\\s\*"

+ ":"

+ "\\s+"

+ "'([^']+)'\\s\*)?$",

Pattern.MULTILINE);

MathWorks社のツールが生成したヘッダファイルのSubsystemとブロックのヒエラルキーを取得するための正規表現パターンです。

#### 静的メンバstepFunction

private static Pattern

stepFunction = Pattern.compile("^(void\\s+"

+ "([\_\\p{Alpha}][\_\\p{Alnum}]+\_step)"

+ "\\s\*\\(\\s\*void\\s\*\\)\\s\*$\\v"

+ "^\\{$\\v)"

+ "|"

+ "(\\s\*#\\s\*include\\s+.\*$\\v)"

+ "|"

+ "^(void\\s+"

+ “([\_\\p{Alpha}][\_\\p{Alnum}]+\_initialize)"

+ "\\s\*\\(\\s\*void\\s\*\\)\\s\*$\\v"

+ "^\\{$\\v)",

Pattern.MULTILINE);

MathWorks社のツールが生成したCコードの、step関数の正規表現パターンです。

#### 静的メンバtidPattern

private static Pattern

tidPattern = Pattern.compile("^\\s\*if\\s\*\\("

+ "("

+ "[\_\\p{Alpha}][\_\\p{Alnum}]\*"

+ "->Timing\\.TaskCounters\\."

+ "(TID\\[(\\d+)\\])[^\\)]\*"

+ ")"

+ "\\)",

Pattern.MULTILINE);

MathWorks社のツールが生成したCコードの、サンプル時間が指定されているブロックの実行を制御するif文のTIDを参照する正規表円パターンです。

#### 静的メンバifPattern

private static Pattern

ifPattern = Pattern.compile("^\\s\*if\\s\*\\("

+ "(.\*)"

+ "\\)\\s\*\\{\\s\*$\\v",

Pattern.MULTILINE | Pattern.DOTALL);

MathWorks社のツールが生成したCコードの、Triggered Subsystem等の実行を制御するif文の条件式の正規表現パターンです。

#### 静的メンバblockMark

private static Pattern

blockMark = Pattern.compile("^\\s\*/?\\\*\\s\*"

+ "(" // 1

+ "(" // 2

+ "(End\\s+of)" // 3

+ "|"

+ "(Update\\s+for)" // 4

+ ")"

+ "\\s+"

+ ")?"

+ "(" // 5

+ "S-Function\\s+\"[^\"]\*\""

+ "([\\s+Block)](file:///\\s+Block))?" // 6

+ "|"

+ "Outputs\\s+for\\s+(.\*)[\\s+SubSystem](file:///\\s+SubSystem)" // 7

+ "|"

+ "[\\p{Alpha}][^:]\*"

+ ")"

+ "\\s\*:\\s+'?"

+ "(" // 8

+ "(<" // 9

+ "(Root|S\\d+)" // 10

+ ">)"

+ "/"

+ "(" // 11

+ "ConcatBufferAt(.\*)In\\d+" // 12

+ "|"

+ "TmpSignal\\s+"

+ "ConversionAt(.\*)Inport\\d+" // 13

+ "|"

+ "([^':]\*[\\p{Alnum}])" // 14

+ ")"

+ ")'?"

+ "\\s+"

+ "(incorporates:|\\\*/?)\\s\*$\\v",

Pattern.MULTILINE);

MathWorks社のツールが生成したCコードの、ブロックのコードを示すコメントの正規表現パターンです。

#### 静的メンバindexEndOf

private static final int indexEndOf = 3;

5.1.1.12静的メンバblockMarkのブロックのコメントにマッチした場合のEnd Of”を指すMatcherのインデクスです。

#### 静的メンバindexUpdateOf

private static final int indexUpdateOf = 4;

5.1.1.12静的メンバblockMarkのブロックのコメントにマッチした場合の”Update for”を指すMatcherのインデクスです。

#### 静的メンバindexType

private static final int indexType = 5;

5.1.1.12静的メンバblockMarkのブロックのコメントにマッチした場合のブロック種別やサブシステム等の前置部を指すMatcherのインデクスです。

#### 静的メンバindexSfUpdate

private static final int indexSfUpdate = 6;

5.1.1.12静的メンバblockMarkのブロックのコメントにマッチした場合のS-FunctionのUpdateを指すMatcherのインデクスです。

#### 静的メンバindexSubSys

private static final int indexSubSys = 7;

5.1.1.12静的メンバblockMarkのブロックのコメントにマッチした場合のSubSystemを示すMatcherのインデクスです。

#### 静的メンバindexBlock

private static final int indexBlock = 8;

5.1.1.12静的メンバblockMarkのブロックのコメントにマッチした場合のブロックのヒエラルキーを指すMatcherのインデクスです。

#### 静的メンバindexSystem

private static final int indexSystem = 9;

5.1.1.12静的メンバblockMarkのブロックのコメントにマッチした場合の、ヒエラルキーの前半部のSubSystem部を指すMatcherのインデクスです。

#### 静的メンバindexCCB

private static final int indexCCB = 12;

5.1.1.12静的メンバblockMarkのブロックのコメントにマッチした場合の”ConcatBufferAt”を指すMatcherのインデクスです。

#### 静的メンバindexSC

private static final int indexSC = 13;

5.1.1.12静的メンバblockMarkのブロックのコメントにマッチした場合の”TmpSignal ConversionAt”を指すMatcherのインデクスです。

#### 静的メンバindexName

private static final int indexName = 14;

5.1.1.12静的メンバblockMarkのブロックのコメントにマッチした場合の、ビエラルキーの後半のブロック名を指すMatcherのインデクスです。

#### 静的メンバblockEndMark

private static Pattern

blockEndMark = Pattern.compile("^\\s\*"

+ "/\\\*"

+ "\\s\*Update\\s+absolute\\s+"

+ "time\\s+for\\s+base\\s+rate\\s\*"

+ "\\\*/");

MathWorks社のツールが生成したCコードの、ブロックのコードの終わりの正規表現パターンです。

#### 静的メンバhierarchyMap

private static Map<String, String>

hierarchyMap = new HashMap<String, String>();

ヒエラルキーのSubSystemをキーとしたSubSystemのBLXMLのブロック名の連想配列です。これは<S1>などの表記とSubSystemのブロック名のマッピングになります。

#### 静的メンバblockNameMap

private static Map<String, String>

blockNameMap = new HashMap<String, String>();

ブロックのヒエラルキーをキーとしたBLXMLのブロック名の連想配列です。

#### 静的メンバblockMap

private static Map<String, Block>

blockMap = new HashMap<String, Block>();

ブロックのヒエラルキーをキーとしたBlockのインスタンスの連想配列です。

#### 静的メンバmeasurePointList

private static List<PointBase>

measurePointList = new ArrayList<PointBase>();

計測ポイントや変更ポイントのリストです。最終的にPointBaseComparatorを使用してソートした順に処理します。

#### 静的メンバstartComment

private static int startComment = -1;

次に処理するコメントの開始位置を保持します。

#### 静的メンバendComment

private static int endComment = -1;

次に処理するコメントの終端位置を保持します。

#### 静的メンバstartStatement

private static int startStatement = -1;

次に処理する文の開始位置を保持します。

#### 静的メンバendStatement

private static int endStatement = -1;

次に処理する文の終端位置を保持します。

#### 静的メンバstartBlock

private static int startBlock = -1;

ブロックに依存しないCの構文ブロックの開始位置を保持します。

#### 静的メンバendBlock

private static int endBlock = -1;

ブロックに依存しないCの構文ブロックの終端位置を保持します。

#### 静的メンバtidInfo

private static TIDInfo tidInfo = null;

参照するTIDInfoのインスタンスを保持します。

#### 静的メンバparsePoint

private static int parsePoint = -1;

パース中のテキスト位置を保持します。

#### 静的メンバcodeEnd

private static int codeEnd;

処理中のテキストの終端位置を保持します。

#### 静的メンバstartBlockComment

private static int startBlockComment = -1;

ブロックのコメントの開始位置を保持します。

#### 静的メンバcurrentBlock

private static Block currentBlock = null;

処理中のブロックの情報を保持します。

#### 静的メンバisUpdate

private static boolean isUpdate = false;

コードがアップデートかどうかを保持します。

#### 静的メンバhasIf

private static boolean hasIf = false;

ifを持つブロック(Trigger等)かどうかを保持します。

#### 静的メンバpreamble

public static CharBuffer preamble = null;

文の前に必要なコードを保持します。これはfor文やif文等です。

#### 静的メンバpostamble

public static CharBuffer postamble = null;

文の後に必要なコードを保持します。これはfor文やif文の構文ブロックの終端です。

#### 静的メンバrestart

private static boolean restart = false;

前回の処理の再開か新しいパースかを保持します。Cの構文中にブロックのコードがある場合はこれをtrueにしてパースの状態をスタックにプッシュし、構文の内部を処理します。状態をスタックから回復した後、その状態から再開します。

#### 静的メンバstateStack

private static List<SaveState> stateStack = new ArrayList<SaveState>();

パース中の状態の状態スタックです。

#### 静的メンバstepFunctionName

private static String stepFunctionName;

step関数の関数名です。

#### 静的メンバinitializeFunctionName

private static String initializeFunctionName;

initialize関数の関数名です。

#### 静的メンバstepFunctionMacroName

private static final String

stepFunctionMacroName = "MEASUREMENT\_STEP\_FUNCTION";

step関数を示すマクロ名です。

#### 静的メンバinitializeFunctionMacroName

private static final String

initializeFunctionMacroName = "MEASUREMENT\_INITIALIZE\_FUNCTION";

initialize関数を示すマクロ名です。

#### 静的メンバifCondtionMacroName

private static final String

ifCondtionMacroName = "MEASUREMENT\_IF\_CONDITION";

if文の条件式をラップするマクロ名です。

#### 静的メンバtidCondtionMacroName

private static final String

tidCondtionMacroName = "MEASUREMENT\_TID\_CONDITION";

TIDの条件式をラップするマクロ名です。

#### 静的メンバstartMacroName

private static final String startMacroName = "MEASUREMENT\_START";

ブロック計測開始のマクロ名です。

#### 静的メンバstopMacroName

private static final String stopMacroName = "MEASUREMENT\_STOP";

ブロック計測終了のマクロ名です。

#### 静的メンバstepBeginMacroName

private static final String stepBeginMacroName = "MEASUREMENT\_STEP\_BEGIN";

step関数の開始に出力するマクロ名です。

#### 静的メンバstepEndMacroName

private static final String stepEndMacroName = "MEASUREMENT\_STEP\_END";

step関数の終端に出力するマクロ名です。

#### メソッドaddSkipPoint ()

private static void addSkipPoint(int start, int end)

スキップするテキストの範囲の開始**start**と終端**end**を登録します。

#### メソッドaddReplacePoint ()

private static void addReplacePoint(int start, int end, String replace)

置換するテキストの範囲の開始**start**と終端**end**と、置換するテキスト**replace**を登録します。

#### メソッドaddInsertPoint ()

private static void addInsertPoint(int point, String insertText)

挿入するテキストの位置pointと挿入するテキスト**insertText**を登録します。

#### メソッドaddDeletePoint ()

private static void addDeletePoint(int start, int end)

削除するテキストの範囲の開始**start**と終端**end**を登録します。

#### メソッドaddMeasurePoint ()

private static void addMeasurePoint(PointType type, int point,

Block block, boolean update)

計測のタイプ**type**と計測ポイント**point**と、計測するブロック**block**とアップデートか否か**update**を登録します。

#### メソッドcanonicalize ()

private static String canonicalize(String name)

文字列**name**をBLXMLで使用するブロック名に正規化します。

#### メソッドmapFile ()

private static CharBuffer mapFile(String file)

指定したファイル**file**をマッピングしたCharBufferを返します。

#### メソッドparseHeader ()

private static boolean parseHeader(String inputHeader)

指定したヘッダファイル**inputHeader**をパースし、ヒエラルキーの情報を取り込みます。

#### メソッドgoNextLine ()

private static int goNextLine(CharBuffer buffer, int point, int end)

CharBuffer **buffer**内でポイント**point**から終端**end**までの間で、行の開始位置を取得します。

#### メソッドgoBeginLine ()

private static int goBeginLine(CharBuffer buffer, int point)

CharBuffer **buffer**内でポイント**point**を含む行の開始位置を取得します。

#### メソッドfindComment ()

private static Range findComment(CharBuffer buffer, int start, int end)

CharBuffer **buffer**内で開始位置**start**と終端位置**end**の範囲の最初のコメントの位置を取得します。

#### メソッドskipComment ()

private static int skipComment(CharBuffer buffer, int start, int end)

CharBuffer **buffer**内で開始位置**start**と終端位置**end**の範囲の最初のコメントと空白文字をスキップした後の位置を取得します(識別子や予約語の開始、特殊文字の位置)。

#### メソッドcheckWord ()

private static int checkWord(CharBuffer buffer,

int start, int end, String word)

CharBuffer **buffer**内で開始位置**start**と終端位置**end**の範囲で指定した単語**word**が出現した場合、単語の終端位置を取得します。

#### メソッドcheckContinuousStatement ()

private static int checkContinuousStatement(CharBuffer buffer,

int start, int end)

checkWord()を用いて、CharBuffer **buffer**内で開始位置**start**と終端位置**end**の範囲でelseやwhileが出現した場合、その単語の終端位置を取得します(elseやwhileの前のセミコロンは文の終端にはならないのでそれを調べる)。

#### メソッドskipStatement ()

private static int skipStatement(CharBuffer buffer, int start, int end,

List<Range> list)

CharBuffer **buffer**内で開始位置**start**と終端位置**end**の範囲の最初の文を読み飛ばします。なお、文の中にCの構文ブロックが一つある場合、その構文ブロックの範囲を**list**に返します。

#### メソッドpushState ()

private static void pushState(int p, boolean r)

private static void pushState(boolean r)

private static void pushState()

パースの状態を状態スタックへ保存します。最初の形式は、再開後のparsePointを更新するためのポイント**p**と再開フラグ**restart**を指定した形式で、次は再開フラグ**restart**のみを指定した形式、最後は完全な状態保存を行うものです。Cの構文内のパースを行う準備として使用します。

#### メソッドpopState ()

private static boolean popState()

パースの状態を状態スタックから回復します。Cの構文内のパースから復帰するために使用します。

#### メソッドresetState ()

private static void resetState(int p, int e)

パースの開始位置**p**と終端**e**を指定してパースの状態を初期化します。Cの構文内のパースを行う目的に使用します。

#### メソッドisEmpyState ()

private static boolean isEmpyState()

パースの状態スタックが空か調べます。状態スタックが空の場合は、step関数内の最も外側のパースを実行中です。

#### メソッドgetBlock ()

private static Block getBlock(Matcher blockMatcher)

ブロックのコメントの正規表現**blockMark**にマッチしたコメントのBlockインスタンスを取得します。ブロックが最初に出現した場合はBlockインスタンスを生成します。

#### メソッドgetPreamble ()

private static String getPreamble()

現在の文の前に必要なコードを取得します。これはfor文やif文等です。

#### メソッドgetPostamble ()

private static String getPostamble()

現在の文の後に必要なコードを取得します。これはfor文やif文の構文ブロックの終端です。

#### メソッドparseCode ()

private static boolean parseCode(CharBuffer code, String inputSource)

ファイル名**inputSource**をマッピングしたCharBuffer **code**内のCのコードをパースします。step関数を見つけ、その内部の計測ポイント・変更ポイントを生成し、また、initialize関数の名前等も抽出します。

#### メソッドsortedBlockMap ()

private static List<Entry<String, Block>> sortedBlockMap()

blockMapをヒエラルキーのSubSystemのIDおよびブロックの名前でソートしたリストを取得します。これはブロックのID割り当てや計測マクロの生成の順番に使用します。

#### メソッドoutputMacro ()

private static void outputMacro(List<Entry<String, Block>> entries)

sortedBlockMap()の結果**entries**の順に計測用のマクロを計測マクロのファイルに出力します。

#### メソッドoutputCode ()

private static void outputCode(String outputCode, CharBuffer inputBuffer)

計測Cコードのファイル名**outputCode**とCコードのファイルをmapFile()でマップしたCharBuffer **inputBuffer**を指定して、計測コードの埋め込み、コード置換を行った計測Cコードを出力します。**outputCode**がnullの場合は標準出力に出力します。

#### メソッドoutputBlock ()

private static void outputBlock(String file,

List<Entry<String, Block>> entries)

出力ファイル**file**に対し、sortedBlockMap()の結果**entries**の順にブロックの情報をCSV形式で出力します。このファイルはデバッグや他の用途に使用することができます。

#### メソッドusage ()

private static void usage()

コマンドの使い方を表示し、exit(1)で終了します。

#### メソッドmain()

public static void main(String args[])

AddMeasurementの主処理です。コマンドラインのオプション処理、ヘッダファイルおよびCコードのパース、計測コード埋め込みやコード置換を行った計測Cコードの出力、計測マクロの出力、ブロックの情報出力を行います。

### Blockクラス

Simulinkのブロックの情報を格納するクラスです

#### Blockコンストラクタ

Block(String block, String system, String prefix, String name)

ヒエラルキーのブロック名**block**、ヒエラルキーのブロック名の前半のシステム部**system**、BLXMLにおけるサブシステム名**prefix**、Simulinkのブロック名**name**を指定し、Blockのインスタンスを生成します。

#### メンバblock

private String block;

ヒエラルキーのブロック名を保持します。

#### メンバsystem

private String system;

ヒエラルキーのブロック名の前半部を保持します。

#### メンバname

private String name;

ヒエラルキーのブロック名の後半部を保持します。

#### メンバblockName

private String blockName;

BLXMLのブロック名を保持します。

#### メンバsystemId

private int systemId;

systemの数値部の値を保持します。例えばS11であれば11です。また、Rootは0です。

#### メンバblockCount

private int blockCount = -1;

ブロックのコードの出力カウントを保持します。

#### メンバupdateCount

private int updateCount = -1;

ブロックのUpdateコードの出現カウントを保持します。

#### メンバid

private int id = -1;

ブロックに割り付けたID番号を保持します。IDはsortedBlockMap()でソートされたリスト順に、0から設定します。

#### メソッドgetBlock()

public String getBlock()

ヒエラルキーのブロック名を取得します。この名前は<Root>/Aとか<S11>/Bとかです。

#### メソッドgetSystem()

public String getSystem()

ヒエラルキーのブロック名の前半部を取得します。<Root>や<S11>といったものになります。

#### メソッドgetName()

public String getName()

ヒエラルキーのブロック名の後半部を取得します。Simulinkのブロック名です。

#### メソッドgetBlockName()

public String getBlockName()

BLXMLのブロック名を取得します。

#### メソッドgetSystemId()

public int getSystemId()

systemの数値部の値を取得します。例えばS11であれば11です。また、Rootは0です。

#### メソッドgetBlockCount()

public int getBlockCount()

ブロックのコードのカウントを取得します。

#### メソッドincrementBlockCount()

public int incrementBlockCount()

ブロックのコードのカウントのインクリメントを行います。カウンタはブロックのコードが分割されている場合、ブロックのコードが出現の都度インクリメントされます。

#### メソッドgetUpdateCount()

public int getUpdateCount()

ブロックのUpdateコードのカウントを取得します。

#### メソッドincrementUpdateCount()

public int incrementUpdateCount()

ブロックのUpdateコードのカウントのインクリメントを行います。カウンタはブロックのアップデートのコードが分割されている場合、ブロックのアップデートのコードが出現の都度インクリメントされます。

#### メソッドsetId()

public void setId(int id)

ブロックに指定したID番号を割り付けます。idは0から開始されます。

#### メソッドgetId()

public int getId()

ブロックに割り付けたID番号を取得します。

### Rangeクラス

テキストの開始と終端の範囲を保持するスーパークラスです。

#### Rangeコンストラクタ

public Range(int start, int end)

範囲の開始**start**と終端**end**を指定しRangeクラスのインスタンスを生成します。

#### メンバstart

protected int start;

範囲の開始位置を保持します。

#### メンバend

private int end;

範囲の終端位置を保持します。

#### メソッドgetStart()

public int getStart()

範囲の開始位置を取得します。

#### メソッドgetEnd()

public int getEnd()

範囲の終端位置を取得します。

### PointBaseクラス

スーパークラスRangeを継承したテキストの変更ポイントのスーパークラスです。

#### PointBaseコンストラクタ

public PointBase(PointType type, int start, int end)

public PointBase(PointType type, int point)

PointBaseの変更ポイントのタイプ**type**、開始**start**、終端**end**を指定しPointBaseのインスタンスを生成します。**end**を指定しない場合は、**end**に-1が指定されたことと同じです。

#### メンバtype

private PointType type;

テキストの変更ポイントに対する処理のタイプを保持します。

#### メソッドgetType()

public PointType getType()

テキストの変更ポイントに対する処理のタイプを取得します。

#### メソッドgetPoint()

public int getPoint()

変更ポイントの位置を取得します。これはgetStart()と同じです。

#### メソッドcompareTo()

public int compareTo(PointBase a)

PointBaseの比較メソッドです。ポイントの位置および変更処理の優先度で比較します。

### PointBaseComparatorクラス

PointBaseの比較用クラスです。

#### メソッドcompare()

public int compare(PointBase a, PointBase b)

PointBase間の比較関数です。PointBaseのソートに使用します。

### SkipPointクラス

スーパークラスPointBaseを継承した、スキップする範囲を保持するクラスです。

#### SkipPointコンストラクタ

public SkipPoint(int start, int end)

スキップする範囲の開始位置**start**と終端位置**end**を指定しSkipPointクラスのインスタンスを生成します。

### ReplacePointクラス

スーパークラスPointBaseを継承した、置換する範囲を保持するクラスです。

#### ReplacePointコンストラクタ

public ReplacePoint(int start, int end, String replace)

置換範囲の開始位置**start**と終端位置**end**と置換テキスト**replace**を指定しReplacePointクラスのインスタンスを生成します。

#### メンバreplace

private String replace;

置換テキストを保持します。

#### メソッドgetReplace()

public String getReplace()

置換テキストを取得します。

### InsertPointクラス

スーパークラスPointBaseを継承した、文字列挿入を行うポイントを保持するクラスです。

#### InsertPointコンストラクタ

public InsertPoint(int point, String insertText)

挿入位置と挿入テキストを指定しInsertPointクラスのインスタンスを生成します。

#### メンバinsertText

private String insertText;

挿入テキストを保持します。

#### メソッドgetInsertText()

public String getInsertText()

挿入テキストを取得します。

### DeletePointクラス

スーパークラスPointBaseを継承した、削除を行う範囲を保持するクラスです。

#### DeletePointコンストラクタ

public DeletePoint(int start, int end)

削除する範囲の開始位置**start**と終端位置**end**を指定しDeletePointクラスのインスタンスを生成します。

### MeasurePointクラス

スーパークラスPointBaseを継承した、計測コードを出力するポイントを保持するクラスです。

#### MeasurePointコンストラクタ

public MeasurePoint(PointType type, int point, Block block, boolean update)

変更ポイントの処理タイプ(計測のタイプ)**type**、位置**point**、ブロック情報**block**、アップデートかどうか**update**を指定しMeasurePointクラスのインスタンスを生成します。

#### メンバblock

private Block block;

計測ポイントのコードのブロックの情報を保持します。

#### メンバupdate

private boolean update;

計測ポイントのコードがアップデートかどうかを保持します。

#### メソッドgetBlock()

public Block getBlock()

計測ポイントのブロックの情報を取得します。

#### メソッドisUpdate()

public boolean isUpdate()

計測ポイントがアップデートかどうか検査します。

### TIDInfoクラス

マルチレートのTIDの情報を保持するクラスです。

#### TIDInfoコンストラクタ

public TIDInfo(int tidIndex, int startOfCondition, int endOfCondition)

TIDのインデクス**tidIndex**とそれを判定するif文の条件式の開始位置**startOfCondition**と終端位置**endOfCondition**を指定しTIDInfoクラスのインスタンスを生成します。

#### メンバtidIndex

private int tidIndex;

TIDのインデクスを保持します。

#### メンバstartOfCondition

private int startOfCondition;

TIDを判定するif文の条件式の開始位置を保持します。

#### メンバendOfCondition

private int endOfCondition;

TIDを判定するif文の条件式の終端位置を保持します。

#### メソッドgetTidIndex()

public int getTidIndex()

TIDのインデクスを取得します。

#### メソッドgetStartOfCondtion()

public int getStartOfCondtion()

TIDを判定するif文の条件式の開始位置を取得します。

#### メソッドgetEndOfCondition()

public int getEndOfCondition()

TIDを判定するif文の条件式の終端位置を取得します。

### SaveStateクラス

Cコードのパース中の状態を保持するクラスです。

#### SaveStateコンストラクタ

public SaveState(int startComment, int endComment,

int startStatement, int endStatement,

int startBlock, int endBlock,

TIDInfo tidInfo, int parsePoint, int codeEnd,

int startBlockComment,

Block currentBlock, boolean isUpdate,

boolean hasIf,

CharBuffer preamble, CharBuffer postamble,

boolean restart)

public SaveState(int parsePoint, int codeEnd)

パース中の各情報を個別指定するか、または、パースの範囲を**parsePoint**と**codeEnd**で指定し、SaveStateのインスタンスを生成します。

#### メンバstartComment

public int startComment;

次に処理するコメントの開始位置を保持します。

#### メンバendComment

public int endComment;

次に処理するコメントの終端位置を保持します。

#### メンバstartStatement

public int startStatement;

次に処理する文の開始位置を保持します。

#### メンバendStatement

public int endStatement;

次に処理する文の終端位置を保持します。

#### メンバstartBlock

public int startBlock;

ブロックに依存しないCの構文ブロックの開始位置を保持します。

#### メンバendBlock

public int endBlock;

ブロックに依存しないCの構文ブロックの終端位置を保持します。

#### メンバtidInfo

public TIDInfo tidInfo;

参照するTIDInfoのインスタンスを保持します。

#### メンバparsePoint

public int parsePoint;

パース中のテキスト位置を保持します。

#### メンバcodeEnd

public int codeEnd;

処理中のテキストの終端位置を保持します。

#### メンバstartBlockComment

public int startBlockComment;

ブロックのコメントの開始位置を保持します。

#### メンバcurrentBlock

public Block currentBlock;

処理中のブロックの情報を保持します。

#### メンバisUpdate

public boolean isUpdate;

コードがアップデートかどうかを保持します。

#### メンバhasIf

public boolean hasIf;

ifを持つブロック(Trigger等)かどうかを保持します。

#### メンバpreamble

public CharBuffer preamble;

文の前に必要なコードを保持します。これはfor文やif文等です。

#### メンバpostamble

public CharBuffer postamble;

文の後に必要なコードを保持します。これはfor文やif文の構文ブロックの終端です。

#### メンバrestart

public boolean restart;

前回の処理の再開か新しいパースかを保持します。Cの構文中にブロックのコードがある場合はこれをtrueにしてパースの状態をスタックにプッシュし、構文の内部を処理します。状態をスタックから回復した後、その状態から再開します。

## SetMeasuringResultのクラス

SetMeasuringResult.java内で定義されるクラスは表 5‑3、SetMeasuringResultクラスの内部クラスは表 5‑4の通りです。

表 5‑3 SetMeasuringResultクラス

|  |  |  |
| --- | --- | --- |
| クラス名 | 内容 | |
| SetMeasuringResult | 計測結果CSVを入力し、計測結果をBLXMLのブロックに反映するクラス | |
| 静的メンバ | 説明 |
| verbose | 実行時のverboseフラグ |
| blockResultMap | ブロック名をキーとしたMeasuringResultのインスタンスの連想配列 |
| 静的メソッド | 説明 |
| loadCSV() | 計測結果CSVを入力し、blockResultMapを構築する |
| process() | BLXMLの入力ブロックを処理し、計測結果があればブロックの属性に反映する |
| usage() | コマンドの使い方を表示し、exit(1)で終了する |
| main() | SetMeasuringResultのmain関数 |
| 内部クラス | 説明 |
| MeasuringResult | 計測結果を格納するクラス |

表 5‑4 SetMeasuringResultクラス内クラス一覧

|  |  |  |
| --- | --- | --- |
| クラス名 | 内容 | |
| MesasuringResult | 計測結果を格納するクラス | |
| コンストラクタ | 説明 |
| MeasuringResult | ブロック名、計測結果、コア番号を指定したコンストラクタ |
| メンバ | 説明 |
| blockName | BLXMLのブロック名 |
| result | 計測値のインスタンス |
| coreNumber | コア番号 |
| used | この結果を使用したかどうか |
| 公開メソッド | 説明 |
| getBlockName() | ブロック名を取得する |
| getResult() | 計測値のインスタンスを取得する |
| getCoreNumber() | コア番号を取得する |
| setUsed() | 使用したことを設定する |
| isUsed() | この結果を使用したかどうかを検査する |

### SetMeasuringResultクラス

#### 静的メンバverbose

private static boolean verbose = false;

プログラム実行時のverboseメッセージの表示有無のフラグです。-vオプションを指定するとtrueになります。

#### 静的メンバblockResultMap

private static HashMap<String, MeasuringResult>

blockResultMap = new HashMap<String, MeasuringResult>();

BLXMLのブロック名をキーとしたMeasuringResultのインスタンスの連想配列です。

#### 静的メソッドloadCSV()

private static void loadCSV(String csvFile)

計測結果CSVを入力し、blockResultMapを構築します。

#### 静的メソッドprocess()

private static void process(BlocksT blocks)

入力BLXMLを処理し、対応するブロックの計測結果のMeasurintResultのインスタンスがあればブロックの属性に計測値とコア番号を反映します。

#### 静的メソッドusage()

private static void usage()

コマンドの使い方を表示し、exit(1)で終了します。

#### 静的メソッドmain()

public static void main(String args[])

SetMeasuringResultのメイン関数です。計測結果CSVファイルの読み込み、入力BLXMLファイルの読み込んで、ブロックへ計測結果とコア番号を反映し、出力BLXMLへ出力します。

### MeasuringResultクラス

#### コンストラクタMeasuringResult()

public MeasuringResult(String blockName, Integer result, int coreNumber)

BLXMLのブロック名**blockName**と計測結果のインスタンス**result**とコア番号**coreNumber**を指定しmeasuringResultのインスタンスを生成します。

#### メンバblockName

private String blockName;

BLXMLのブロック名を保持します。

#### メンバresult

private Integer result;

計測結果(Integer)のインスタンスを保持します。nullの場合、計測結果がありません。

#### メンバcoreNumber

private int coreNumber;

コア番号を保持します。

#### メンバused

private boolean used;

この計測結果を使用したかどうかのフラグです。使用した場合はtrueです。

#### メソッドgetBlockName()

public String getBlockName()

BLXMLのブロック名を取得します。

#### メソッドgetResult()

public Integer getResult()

計測結果を取得します。nullの場合は計測結果がありません。

#### メソッドgetCoreNumber()

public int getCoreNumber()

コア番号を取得します。

#### メソッドsetUsed()

public void setUsed()

この計測結果を使用したことを設定します。

#### メソッドisUsed()

public boolean isUsed()

この計測結果が使用されたかどうかを検査します。trueの場合は使用されたことを示します。

# RTW\_Measurement(CForest実行環境)

RTW\_Measurementは、CForest MP向け実行環境です。ファイル構成は表 6‑1の通りです。ここでは、measurement\_cforestについて説明します。

表 6‑1 RTW\_measurementのファイル構成

|  |  |  |  |
| --- | --- | --- | --- |
| ディレクトリ | サブディレクトリ  ファイル | | 説明 |
| cforest用計測環境 | | | |
| measurement\_cforest | reame.txt | | cforest計測環境に関するメモ |
| MyEnv.sh | | 環境設定ファイル(要変更) |
| Temlate-user.mk | | ユーザ用makefileテンプレート |
| Makefile | | 性能計測実行用makefile |
| measurement.mk | | 環境設定・ビルド用makefile |
| changedir.mk | | ディレクトリ変更用makefile |
| run.sh | | 性能計測実行スクリプト |
| cforest.sh | | cforest実行スクリプト |
| gdb.sh | | gdb実行スクリプト |
| lib | hwdep.c | cforest依存関数のコード |
| hwdep.h | cforest依存関数のヘッダ |
| boot.s | cforest用ブートコード |
| sample.ld | cforest用リンカスクリプト |
| include | measure\_cforest.h | cforest向け計測用ヘッダ |
| sub | gdb-command.x | gdb実行コマンド |
| gdb.py | cforest用設定スクリプト |
| サンプルコード | | | |
| sample\_c\_gmres | c\_gmres\_measure.c | | 計測Cコード |
| c\_gmres\_meashre.h | | 計測マクロ |
| Makefile | | メインのmakefile |
| user.mk | | c\_gmres用makefile |
| sample\_synrmmodel | measure.c | | 計測Cコード |
| measure.h | | 計測マクロ |
| Makefile | | メインのmakefile |
| user.mk | | synrmmodel用makefile |

## measurement\_cforestの構成

measurement\_cforestは、CForest MP向けの性能計測用環境です。プログラムのビルド、CForestでの計測実施と計測結果の出力を行います。

基本的に全体的な設定として「6.1.2MyEnv.sh」のCForest用のツール環境の設定を一度行い、各モデルの計測の際には、個別のディレクトリに「6.1.3Tepllate-user.mk」「6.1.4Makefile」をコピー修正して利用になります。

### readme.txt

CForest MPの性能計測環境に関するメモです。

### MyEnv.sh(要変更)

CForest MP用のコンパイラのパスを**TOOL\_PATH**に設定し、CForestシミュレータのパスを**CFOREST\_PATH**に設定します。基本的に一度だけ設定を行うものですが、各ツールのパスを変更した場合は合わせてこのファイルを変更してください。

なお、このファイルは計測プログラムのビルド時に、モデル個別のディレクトリへ、シンボリックリンクされます。

### Tepllate-user.mk(要コピー・変更)

各モデル毎に個別に用意するuser.mkのテンプレートです。本ファイルをuser.mkという名前でコピーし、設定する必要があります。設定が必要な変数は表 6‑2の通りです。

表 6‑2 user.mkの変数

|  |  |
| --- | --- |
| 変数 | 説明 |
| AUX\_SRC\_DIR | 計測Cコード以外のCソース、ヘッダがあるディレクトリ |
| AUX\_C\_SRCS | 計測Cコード以外にコンパイルするCソースファイル |
| AUX\_S\_SRCS | コンパイルするアセンブラソースファイル |
| TARGET\_SRC | 計測Cコード(デフォルトでmeasure.cが設定) |
| TARGET\_HDRS | 計測マクロおよびCForest用ヘッダ |
| オプション設定 | |
| CFLAGS | Cコードをコンパイルする際のオプション(最適化オプション等) |
| CPPFLAGS | Cプリプロセッサ用オプション(-Iや-D) |
| LDFLAGS | リンカ用オプション(-Lオプション等) |
| LDLIBS | ライブラリ指定(-lオプション) |

user.mkは、「6.1.5measurement.mk」で入力され、ビルドのルールに使用されます。

### Makefile(要コピー・変更)

各モデル毎に個別に用意するMakefileです。本ファイルをコピーし、設定を行う必要があります。設定が必要な変数は表 6‑3の通りです。なお、コピー先がmeasure\_cforestと同列のディレクトリであれば、変更は不要です。

表 6‑3 Makefileの変数

|  |  |
| --- | --- |
| 変数 | 説明 |
| MEASUREMENT\_TOOL\_DIR | measurement\_cforestの相対または絶対パスを設定  デフォルト設定は“../measurement\_cforest”である |

Makefileは**MEASUREMENT\_TOOL\_DIR**で指定されたパスにある、「6.1.6changedir.mkchangedir.mk」をインクルードします。

### measurement.mk

環境コピーや計測プログラムのビルドを行うためのmakefile本体です。共用され、変更の必要はありません。

### changedir.mk

設定した**MEASUREMENT\_TOOL\_DIR**でmakeのmeasurement.mkの各ターゲットを実行するmakefileです。共用され、変更の必要はありません。

### run.sh

利用者が、計測を実行し測定結果のCSVファイルを出力するために用いるスクリプトです。内部的にcforest.shとgdb.shを呼び出し、実行後、シミュレータのログから計測結果を切り出します。また、共用され、変更の必要はありません。

なお、このファイルは計測プログラムのビルド時に、モデル個別のディレクトリへ、シンボリックリンクされます。

### cforest.sh

CForest MPシミュレータを起動するスクリプトです。run.shから呼び出され、利用者は直接使用しません。また、共用され、変更の必要はありません。

なお、このファイルは計測プログラムのビルド時に、モデル個別のディレクトリへ、シンボリックリンクされます。

### gdb.sh

CForest MPシミュレータを制御するgdbを起動するスクリプトです。run.shから呼び出され、利用者は直接使用しません。また、共用され、変更の必要はありません。

なお、このファイルは計測プログラムのビルド時に、モデル個別のディレクトリへ、シンボリックリンクされます。

### lib

CForest用機種依存コードおよびリンカスクリプトを置くディレクトリです。

#### hwdep.c

CForestのHW依存コードです。hwdep\_printf()のようなテキスト出力APIを使用しています。共用され、変更の必要はありません。

#### hwdep.h

CForestのHW依存関数やマクロを定義したヘッダファイルです。サイクルカウンタへのアクセスマクロ等を使用しています。共用され、変更の必要はありません。

#### boot.s

ブートストラップ用のアセンブラコードです。共用され、変更の必要はありません。

#### sample.ld

CForestのメモリマップに対応したリンカスクリプトです。共用され、変更の必要はありません。

### include

計測Cコードで利用するマクロを定義した計測用のヘッダファイルを配置しているディレクトリです。

#### measure\_cforest.h

「4.3AddMeasurementによる計測Cコードのマクロ」のCForest用の性能計測用マクロ定義です。共用され、変更の必要はありません。

計測したサイクルを保存するための領域measure\_cycles[]を定義し、計測用の各マクロは、サイクル数を取得する際に、inline展開やコピーを禁止した共通の関数を呼び出すことで、最適化時においても、均一な計測オーバヘッドになることと、ブロックのコードが計測用のマクロを跨いでの移動を抑止するよう可能な範囲で対応しています。

main関数では、10回step関数を実行し、各ブロックの計測結果の平均値を求め、run.shで切り出し可能なように、CSV形式で表示します。

### sub

性能計測実行時に必要なサブファイルを配置しています。

#### gdb-command.x

gdbのコマンドを記述したファイルです。共用され、変更の必要はありません。

#### gdb.py

gdbを使用する設定を行ったCForestの設定ファイルです。共用され、変更の必要はありません。

以上