Docgen

(Document Generator)

User Guide

version 1.6

May 17, 2024

Important Notice

『Freely available under the terms of the 3-Clause BSD License』

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)

HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Document Revision History

Doc Revision Number	Date	Description
1.6	May 17, 2024	Markdown can be specified as the input source file
1.5	April 9, 2024	add lua snippet option on command line
1.4	February 5, 2024	add lua function call in paragraph
1.3	November 12, 2021	support excel table
1.2	July 3, 2021	add word's @**<property>** tag variable
1.1	April 2, 2020	add Visio page to picture & bookmark automation
1.0	February 4, 2019	Initial relase

Table of Contents

List of Tables

Table **Title** **Page**

List of Figures

Figure **Title** **Page**

List of Terms

List	Description
TestDrive	TestDrive Profiling Master (https://testdrive-profiling-master.github.io/)
Lua	Lua script language (Wiki, Homepage)
WORD	Microsft Office's word processor
Visio	Microsoft Office's Visio - Diagramming and vector graphics application
PDF	Portable Document Format (Wiki)
markdown	Mark-up language (Wiki)
VBA	Microsoft's Visual Basic for Application

1. 소개

만약 소개 과정을 생략하고 바로 사용하고자 한다면, 반드시 '1.4.3. 선행 과정'을 참고하여 진행해 주시기 바랍니다.

신규 기능이 필요하거나 개선사항이나 버그를 발견하시면, 정형기(clonextop@gmail.com)에게 건의 부탁드리며, 이 문서 또한 docgen으로 작성 및 생성 되었음을 알려드립니다.

1.1 주요 기능

docgen 은 Template 워드(.docx) 파일을 참조하여 새로운 WORD 문서(.docx)와 PDF 문서(.pdf)를 자동 생성 해주는 툴입니다.

기본적으로 CodeGen을 통해 범용적인 Lua 프로그래밍 환경과 markdown에 호환되는 문법을 통해 text 형태로 문서를 작성할 수 있습니다. 주요 기능을 나열하면 아래와 같습니다.

- Lua 스크립트를 통해 다양한 문자 변조 기능 지원
- WORD 및 PDF 파일 자동 생성
- 자동화된 절/캡션(Caption)/상호참조 생성 기능
- 그림, 표, 스타일 서식, 코드 인용, 수학 수식 표현 기능
- 워터마크(water mark) 삽입 기능
- 워드 템플릿 문서 기반으로 사용자가 손쉽게 다양한 스타일 변경 구축과 일관된 서식 적용

1.2 문서 자동화 생성 과정

문서 자동화 생성 과정은 아래와 같은 단계로 진행됩니다.

1. 템플릿 문서(template.docx) 열기
2. 사용자 lua 코드로 부터 아래 내용 추가 하기
3. 워드(.docx) 문서 생성
4. .docx 문서 필드 갱신 및 PDF(.pdf) 파일 생성

1.3 실행 방법

docgen 을 실행하기 위해서는 아래와 같이 실행합니다.

```
> docgen
```

```
Document Generator for TestDrive Profiling Master. v1.6
Usage: docgen  [--help] [-t template] [-l language] [-r|--run=lua_code] input_file
[output_file]

      --help                display this help and exit
  -t template               Document template name/file.
                            *** Installed docgen template list ***
                            testdrive      : TestDrive Profiling Master
                            (default : testdrive)
  -l language               Document language code string.
                            'docgen_language' variable in Lua
                            (default : 'en')
  -r, --run=lua_code        Run Lua snippet code
  input_file                input Lua or .md(markdown) file
  output_file               output Microsoft Word(.docx) file
```

실행 명령 : docgen INPUT_LUA_FILE OUTPUT_DOCX_FILE

INPUT_LUA_FILE에 해당하는 Lua 스크립트 또는 Markdown 문서를 입력 소스 파일로 지정하며, OUTPUT_DOCX_FILE을 지정하지 않을 경우 주어진 property를 참조하여 자동으로 알맞게 생성합니다. '-t' 옵션은 기본 바탕이 될 템플릿 문서를 지정하게 되는데, 템플릿 문서를 지정하지 않을 경우 기본 docgen_template.docx로 지정하도록 되어 있으나 이를 참조/변경하여 다양한 문서 형태를 만들 수 있습니다.

'-l' 옵션은 임의의 언어코드를 지정합니다. Lua 에서는 'docgen_language' 변수로 확인할 수 있고, 문장에서는 **'3.12. 문서 활성화'** 표현식을 사용하여, 원하는 언어 코드를 선택할 수 있습니다.

'-r' 옵션은 임의의 선행 Lua 코드를 지정합니다. 이 코드는 문서 구조를 변경을 위한 조건을 넣을 수 있습니다.

NOTE: 기본 설정은 'en'(english) 이며, 이 'language' 설정을 사용하지 않는 문서에는 영향을 주지 않습니다.

1.3.1 프로젝트 생성 예

아래와 같이 명령어를 입력하여, 간소화된 프로젝트를 생성하고 빌드 할 수 있습니다.

```
> create_project docgen_simplified example
*I: Create DocGen simplified project : 'example'
Run 'build.bat' to build document.

> cd example

> ls
build.bat  main.md  media

> build.bat
1. Introduction
   Main functions
2. first Contents
   media contents
*I: Link all bookmarks.
*I: Build document : Header_Name_userguide_rev1.0.docx
*I: Fields calculation & Saving to PDF output : Header_Name_userguide_rev1.0.pdf
```

1.4 사용상 제한 및 라이센스 허가와 선행 과정

1.4.1 제한

WORD의 VBA가 사용되므로 WORD가 설치된 윈도우즈 PC 환경이 필요합니다. 또한 그림 삽입에 Visio 파일(vsd/vsdx)을 바로 지정하기 위해서는 Visio 설치가 별도로 필요합니다. 리눅스 상에서도 생성은 가능하나, 문서 전체 필드 업데이트(WORD에서 직접 열어 수동으로 업데이트는 가능)/워터마킹/pdf변환 등의 기능이 수행되지 않습니다.

1.4.2 라이센스 허가

docgen 에 구현된 소스는 BSD 라이센스를 준수하며, 문서 생성에 사용된 사용자의 개별 스크립트나 이미지등의 2차 저작물은 온전히 사용자의 소유입니다.

1.4.3 선행 과정

Microsoft Word 의 설정 자동화 제한으로 수식 표현에 대하여, 초기에 MS 표준 수식 표현인 유니코드로 고정되어 있습니다. 이를 "LaTeX"로 표현하고자 한다면, Figure 1-1와 같이 최초 한번 리본 메뉴의 '수식/변환'(먼저 '삽입/수식'으로 수식이 생성한 후 선택되어야 아래 메뉴가 보임.) 탭에서 '**/유니코드**' 대신 '**{} LaTeX**'를 선택하는 과정이 필요합니다.

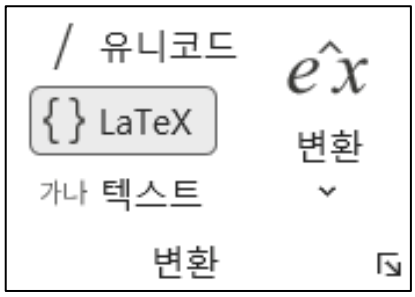

Figure 1-1. LaTeX 설정

NOTE: Microsoft 에서는 현재 이 부분에서 자동 변경 방법을 제공하지 않고 있으며, 왜 변경 기능을 막아놨는지 모르겠음.

1.5 알려진 문제들

1.5.1 Excel table의 지역화 문제

엑셀의 숫자 데이터의 표현에 대해서, 원래는 기본 표현형식 날짜의 'number format'이 국가 설정에 따라 다릅니다.

예) 2024년 3월 8일의 경우

```
미국 : number format("M/D/YYYY")
       -> '3/8/2024'
한국 : number format("YYYY-M-D")
       -> 2024-3-8
```

위와 같이 국가설정에 따라 연도/월/일 표시 순서가 다르지만, 여기서는 미국 표현형식만을 사용합니다. 만약 다른 형식을 사용하고자 한다면, excel에서 기본 형식이 아닌, 직접 사용자 정의 'number format'을 입력해야 합니다.

2. Lua 표현

Lua 스크립트는 매우 가벼고 속도가 빠르며 단순하여, 비-프로그래머를 위해 주로 사용되는 언어입니다. '워크래프트 WoW', '앵그리버드' 등 각종 게임에서 비 프로그래머인 게임 디자이너가 사용한 예시가 있습니다.

docgen 역시 '1.3. 실행 방법'에서 나열한 것과 같이 docgen.lua 파일을 실행하여 동작합니다.

세부적으로는 범용적인 Lua 의 기본 기능과 codegen에 추가된 기능을 사용하여 docgen을 구현하고, 다시 이 구현을 통해 문서 자동화가 이루어 집니다.

표현은 범용적인 Lua 문법과 CodeGen의 추가 문법 그리고 아래에서 나열되는 기능들이 추가됩니다.

- 외부 Lua 관련 링크
  - 쉽게 배우는 프로그래밍 입문/Lua
  - property[] 속성 객체
  - AddRevision() 함수
  - AddTerm() 함수
  - AddParagraph() 함수

여기서는 docgen 에 추가된 기능만을 설명하고 있으며, Lua 에 대한 학습 없이 최소 template 구현만으로 문서를 생성할 수 있습니다.

2.1 속성 객체

```
property["Document_Name"]           -- 문서 이름
property["IP_Version"]              -- IP 버전 (예:"1.00")
property["Main_Title"]              -- 문서 첫장의 메인 타이틀 이름
property["Sub_Title"]               -- 문서 첫장의 서브 타이틀 이름 (생략 가능)
property["IP_Name_First_Page"]      -- 문서 첫장의 이름
property["IP_Name_Header"]          -- 헤더 및 파일 상의 이름 (예 : "doc_guide")
property["Ownership"]               -- 오너쉽 명
property["Document_Name_Header"]    -- 헤더 이름 (...)
property["Water_Mark"]              -- 워터마킹 문구 (사용하지 않을 경우 비워 둔다.)
```

워드에는 메뉴 "파일/정보" 에 Figure 2-1와 같이 '속성' 탭이 존재합니다.

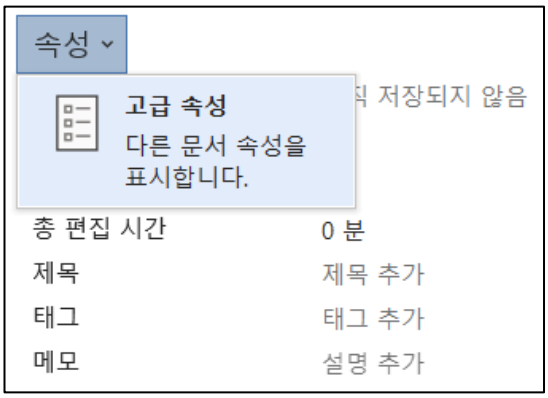

Figure 2-1. WORD 속성 탭

이 메뉴에서 사용자가 원하는 Field 속성을 추가하거나, 기존 Property를 변경할 수 있습니다. 선언된 Property 는 문서 전체의 Field 정보에 반영됩니다.

이걸 lua 코드 내에서는 아래와 같은 방법으로 템플릿 문서에 이미 존재하는 필드 값을 변경 가능합니다.

또는 속성(Property) 참조에서 처럼 markdown 문장 내에서 직접적인 참조가 가능합니다.

예) 이 문서의 Field 지정 예시 ("--" 로 시작하는 문장은 Lua에서 '주석' 표기를 의미합니다.)

```
property["Document_Name"]           = "UserGuide"
property["IP_Version"]              = "1.00"
property["Main_Title"]              = "Document Generator"
property["IP_Name_First_Page"]      = "User Guide"
property["IP_Name_Header"]          = "document_generator"
property["Ownership"]               = "TestDrive"
property["Document_Name_Header"]    = "userguide"
property["Water_Mark"]              = "TESTDRIVE CONFIDENTIAL"
```

2.2 AddRevision() 함수

함수 원형 : AddRevisoion(버전, 연도, 월, 일, 설명)

문서의 버전 정보를 관리하는 함수입니다. 아래 예시와 같이 버전 순서로 나열할 수 있으며, 설명은 "**3. Paragraph 표현**"을 사용할 수 있습니다.

AddRevision() 함수를 한번도 사용하지 않을 경우는 "Document Revision History" 절은 자동으로 제거됩니다.

예)

```
AddRevision("1.0",  2022, 1,  14,   "Initial Draft")
AddRevision("1.1",  2022, 2,  15,   "Second release")
AddRevision("1.2",  2022, 3,  16,   "Third release")
```

결과)

Document Revision History

Doc Revision Number	Date	Description
1.2	March 16, 2022	Third release
1.1	February 15, 2022	Second release
1.0	January 14, 2022	Initial Draft

Figure 2-2. AddRevision() 함수 사용 예시

2.3 AddTerm() 함수

함수 원형 : AddTerm(단어, 설명)

단어에 대한 설명을 삽입하는 함수입니다. 아래 예시와 같이 사용될 수 있으며, 설명은 "3. **Paragraph 표현**"을 사용할 수 있습니다.

AddTerm() 함수를 한번도 사용하지 않을 경우는 "List of Terms" 절은 자동으로 제거됩니다.

```
AddTerm("TestDrive", "TestDrive Profiling Master (@<link:https://testdrive-profiling-master.github.io/>)")
AddTerm("Lua", "Lua script language (@<link:https://ko.wikipedia.org/wiki/루아_(프로그래밍_언어);Wiki>, @<link:http://www.lua.org/;Homepage>)")
```

결과)

List of Terms

List	Description
TestDrive	TestDrive Profiling Master (https://testdrive-profiling-master.github.io/)
Lua	Lua script language (Wiki, Homepage)

Figure 2-3. AddTerm() 함수 사용 예시

2.4 AddParagraph() 함수

함수 원형 : AddParagraph(단어, 문장)

실제 문서 내용을 채울 수 있으며, 문장은 "3. **Paragraph 표현**"을 사용할 수 있습니다.

문장의 내용을 직접 문자열로 기술할 수 있지만, 문장을 "**[[파일명]]**"으로 기술하면 외부 text 파일로 참조하게 되며, 직접 문장 표기는 POSIX 이스케이프 문자 표기법에 영향을 받으므로 소프트 프로그래밍에 익숙하지 않은 사용자는 "예 #2)"와 같이 외부 text 파일로 기술하는 것을 권장합니다.

예 #1) 직접 구현

```
AddParagraph("#제목\
문단 내용 1\
문단 내용 2")
```

예 #2) 외부 markdown 표현 text 파일 구현

```
AddParagraph("[[some.txt]]")
```

예 #3) 파일 읽기 함수를 활용하여 구현

```
do
    local   txt_contents    = String(nil)
    txt_contents:ReadFile("some.txt", false)
    AddParagraph(txt_contents.s)
end
```

예 #4) 'IsInsert' 변수를 확인하여, 여러 파일 읽은 뒤,

"%ABC%" 표현을 "good" 으로 모두 변경 후 문장 삽입하는 Lua 응용

```
if IsInsert == true then    -- directive 확인
    local   txt_contents    = String(nil)

    txt_contents:Append(ReadText("some_1.txt")) -- some_1.txt 추가
    txt_contents:Append(ReadText("some_2.txt")) -- some_2.txt 추가
    txt_contents:Append(ReadText("some_3.txt")) -- some_3.txt 추가
    txt_contents:Replace("%ABC%", "good", true) -- "%ABC%"를 "good"로 모두 변경

    AddParagraph(txt_contents.s)    -- 문서 적용
end
```

3. Paragraph 표현

Paragraph 표현은 Lua 함수 AddRevision, AddTerm, AddParagraph 의 인자 설명 및 문장을 기술하는 법을 의미합니다. 표기 방법은 markdown 과 몇몇 html 표기법을 혼합하여 취하고 있으며, 표현의 다향성을 위해 추가적으로 자체 문법도 가지고 있습니다. 또한 각 수식은 문단, 표등 문장이 쓰이는 어디서든 동등하게 사용할 수 있습니다.

NOTE: Paragraph 표현에 대한 건의 사항이 개선 요청은 언제든지 저에게 연락해 주세요. (1. 소개 참조.)

3.1 제목 표현

표현식 : # 대제목

부제목1

부제목2

부제목3

부제목4

부제목5

문서 제목 표기는 대제목과 부제목n까지 '#' 문자를 라인 선두에 두어 제목을 표시합니다. 이는 markdown 과 같습니다. '#' 문자가 라인 선두에 없으면 제목으로 인식하지 않으며, 워드의 템플릿 문서의 "텍스트 스타일" 에 영향을 받습니다. 최대 6단 까지 사용 가능.

3.2 목록화

표현식 : * 요소1

**** 요소2**

***** 요소3**

**** 요소4

***** 요소5

****** 요소6

목록화는 markdown 과 비슷하지만 더 다양한 표현기능을 위해 약간의 차이가 있습니다. '*' 문자가 줄 선두에 존재할 경우 목록화로 표시되며, '*' 개수만큼 들여쓰기가 결정됩니다.

또한 번호나 특수한 표현을 위해 나열된 '*' 문자 다음에 '>' 를 사용하여, 기존 표식을 없애고 다른 형태의 표현식을 쓸 수 있습니다. 최대 6단 까지 사용 가능.

예)

```
* 첫번째 요소
** 두번째 요소
*** 세번째 요소
**** 네번째 #1 요소
**** 네번째 #2 요소
**** >네번째 #2 요소 확장
**** @<b>@<color:FF0000>목록@</color>(目錄, List)@</b>은 사람의 이름이나 물품의 이름, 책 제목, 목차, 점검 해야할 항목 따위를 일정한 기준과 순서로 적어 알아보기 쉽도록 만든 것을 이르는 말이다.
* >1). 첫번째 번호 요소
** >1-1). 두번째 번호 요소
* >2). 첫번째 번호 요소
** >2-1). 첫번째 번호 요소
*** >표식 없는 요소 연장 단락
```

결과)

- 첫번째 요소
  - 두번째 요소
    - 세번째 요소
      - 네번째 #1 요소
      - 네번째 #2 요소

        네번째 #2 요소 확장
      - **목록(目錄, List)**은 사람의 이름이나 물품의 이름, 책 제목, 목차, 점검 해야할 항목 따위를 일정한 기준과 순서로 적어 알아보기 쉽도록 만든 것을 이르는 말이다.

1). 첫번째 번호 요소

1-1). 두번째 번호 요소

2). 첫번째 번호 요소

2-1). 첫번째 번호 요소

표식 없는 요소 연장 단락

3.3 문자 수식

문자 수식은 글자의 색상, 굵기, 이텔릭, 밑줄, 크기, 위첨자/아래첨자 등을 지정할 수 있으며, 적용 범위는 한 라인 내에서만 제한되며, 여러 수식 표기법이 중복 표현 될 수 있습니다.

3.3.1 굵은 글자

표현식 : @표현식@

굵은 글씨는 표현식 을 HTML 방식과 유사하게 'b' 태그로 둘러쌓아 표현합니다.

예)

```
@<b>굵은 글씨입니다.@</b> 굵은 글씨가 아닙니다.
```

결과 : **굵은 글씨입니다.** 굵은 글씨가 아닙니다.

3.3.2 이텔릭체

표현식 : @표현식@

이텔릭체 문장은 표현식 을 HTML 방식과 유사하게 'i' 태그로 둘러쌓아 표현합니다.

예)

```
@<i>이텔릭체@</i> Non 이텔릭체
```

결과 : *이텔릭체* Non 이텔릭체

3.3.3 밑줄

표현식 : @<u>표현식@</u>

밑줄 글씨 표현은 표현식 을 HTML 방식과 유사하게 'u' 태그로 둘러쌓아 표현합니다.

예)

```
@<u>밑줄 글자@</u>
```

결과 : 밑줄 글자

3.3.4 취소선

표현식 : @<s>표현식@</s>

취소선 표현은 표현식 을 HTML 방식과 유사하게 's' 태그로 둘러쌓아 표현합니다.

예)

```
@<s>취소선 글자@</s>
```

결과 : ~~취소선 글자~~

3.3.5 위첨자

표현식 : @^{표현식@}

위첨자 표현은 표현식 을 HTML 방식과 유사하게 'sup' 태그로 둘러쌓아 표현합니다.

예)

```
글씨의@<sup>위첨자@</sup>
```

결과 : 글씨의위첨자

3.3.6 아래첨자

표현식 : @_{표현식@}

아래첨자 표현은 표현식 을 HTML 방식과 유사하게 'sub' 태그로 둘러쌓아 표현합니다.

예)

```
글씨의@<sub>아래첨자@</sub>
```

결과 : 글씨의아래첨자

3.3.7 글자 색상

표현식 : @<color:색상값>표현식@</color>

글자 색상 변경을 위해서는 표현은 표현식 을 HTML 방식과 유사하게 'color' 태그로 둘러쌓아 표현하며, 색상 지정은 색상값에 24bit RGB 16진수로 표현합니다.

예)

```
@<color:FF0000>붉은 글씨@</color> 표현
```

결과 : 붉은 글씨 표현

3.3.8 글자 크기

표현식 : @<size:크기값>표현식@</size>

글자 크기 변경을 위해서는 표현은 표현식 을 HTML 방식과 유사하게 'size' 태그로 둘러쌓아 표현하며, 색상 지정은 크기값에 point 단위로 지정합니다.

예)

```
@<size:30>큰 글씨@</size> @<size:10>작은 글씨@</size>
```

결과 : 큰 글씨 작은 글씨

3.3.9 문단 스타일

표현식 : :::스타일_이름

다음 줄의 문단 스타일을 '스타일_이름'로 변경합니다. 한 줄에 대해서만 스타일이 변경되며, 다음 라인에 대해서는 원래의 스타일 서식으로 다시 돌아갑니다.

스타일 서식은 초기 템플릿 문서에 지정된 스타일이 참조되어 구현되며, WORD 상에 '텍스트 스타일'로 검색하여 변경 및 추가할 수 있습니다.

예) 노트 서식을 적용

```
:::NoteHeading
민간인과 군인 사망자를 모두 합하여 약 6,000 만~7,000 만 명에 달하는 사람들이 제 2 차 세계
대전으로 인해 사망했다. 이 전쟁의 여파로 서구권에서는 그동안 사회 주류였던 집단주의 사상이
쇠퇴하고 개인주의 사상이 대두되어 오늘날까지 이어지게 된다.
```

결과)

NOTE: 민간인과 군인 사망자를 모두 합하여 약 6,000 만~7,000 만 명에 달하는 사람들이 제 2 차 세계 대전으로 인해 사망했다. 이 전쟁의 여파로 서구권에서는 그동안 사회 주류였던 집단주의 사상이 쇠퇴하고 개인주의 사상이 대두되어 오늘날까지 이어지게 된다.

3.4 상호 참조

표현식 : @<bookmark:대상>

제목, 그림, 표등의 상호 참조를 구현합니다.

대상에 그림, 표의 캡션 내용이나 제목의 내용을 적으면 됩니다.

3.4.1 그림 또는 표의 상호 참조

예)

```
@<bookmark:LaTeX setting>
```

결과 : Figure 1-1

3.4.2 제목의 상호 참조

제목의 경우 아래와 같이 Prefix 유무에 따라 상호 참조의 표현은 크게 4가지로 구현 가능합니다.

1. 일반 문장 참조 (별도의 prefix 없이 표현)
   : **대상**
2. 페이지 번호 참조
   : **&대상**
3. 장/절 번호 참조
   : **#대상**
4. 장/절 번호 및 문장 전체 참조
   : **@대상**

아래 예제는 실제 사용 예시를 보이며, 문장 클릭시 현 문서의 링크로 이동 됨을 확인 할 수 있습니다.

예) 일반 문장 참조

```
@<bookmark:사용상 제한 및 라이센스 허가와 선행 과정>
```

결과 : 사용상 제한 및 라이센스 허가와 선행 과정

예) 페이지 번호 참조

```
@<bookmark:&사용상 제한 및 라이센스 허가와 선행 과정>
```

결과 : 11

예) 장/절 번호 참조

```
@<bookmark:#사용상 제한 및 라이센스 허가와 선행 과정>
```

결과 : 1.4

예) 장/절 번호 및 문장 전체 참조

```
@<bookmark:@사용상 제한 및 라이센스 허가와 선행 과정>
```

결과 : 1.4. 사용상 제한 및 라이센스 허가와 선행 과정

3.5 속성(Property) 참조

표현식 : @<property:속성이름>

'Lua 표현' 중 '속성 객체'로 다양한 이름의 속성을 만들 수 있으며, 이를 본문에서 사용하는 방법을 제공한다.

예)

```
@<property:Main_Title>, @<property:Ownership>
```

결과 : Docgen, clonextop@gmail.com

3.6 하이퍼 링크(Hyperlink) 표현

표현식 : @<link:URL_대상;표시_문구>

하이퍼 링크는 위와 같이 표시하며, 표시_문구는 생략 가능합니다.

예) "https://testdrive-profiling-master.github.io/" 링크 표현

```
@<link:https://testdrive-profiling-master.github.io/;TestDrive Profiling Master>
```

결과 : TestDrive Profiling Master

3.7 수학 수식 표현

표현식 : $$수학수식$$

수학 수식의 경우 markdown 과 같게 수학 수식에 "$$" 로 둘러싸서 표시합니다.

단! WORD 의 수학 수식은 기본이 "**유니코드**" 표현식으로 초기 지정되어 있습니다. 때문에 "**LaTeX**" 수식을 활용하고자 한다면, "1.4.3**. 선행 과정**"을 참고하여 설정을 변경해야 함을 유의합니다.

수학 수식을 단일로 표현할 경우, 가운데 정렬로 표기 되지만, 문장 중간에 표현할 경우 문장과 어울리도록 표현이 자동 변경되며, 아래의 예제는 LaTeX 표현으로 구현된 내용이므로 만약 결과 표기가 잘못 되어 있다면, 위 "선행 과정"을 시도하지 않은 결과입니다.

예) 수학 수식 표현 2가지

```
$$f\left(x\right)=a_0+\sum_{n=1}^{\infty}\left(a_n\cos{\frac{n\pi
x}{L}}+b_n\sin{\frac{n\pi x}{L}}\right)$$
```

```
문장 안의 표현 식은 이렇게
$$f\left(x\right)=a_0+\sum_{n=1}^{\infty}\left(a_n\cos{\frac{n\pi x}{L}}+b_n\sin{\frac{n\pi x}{L}}\right)$$ 변합니다.
```

결과 :

$$f(x) = a_0 + \sum_{n=1}^{\infty} \left(a_n \cos\frac{n\pi x}{L} + b_n \sin\frac{n\pi x}{L} \right)$$

문장 안의 표현 식은 이렇게 $f(x) = a_0 + \sum_{n=1}^{\infty} \left(a_n \cos\frac{n\pi x}{L} + b_n \sin\frac{n\pi x}{L} \right)$ 변합니다.

WORD의 수학 수식은 Figure 3-1처럼 같이 수식을 먼저 만든 후, 메뉴에서 '1차원 형식' 버튼을 눌러 변경 가능합니다. 이 문자열을 양끝단에 "$$" 문자를 덧붙여 수학 수식으로 작성할 수 있습니다.

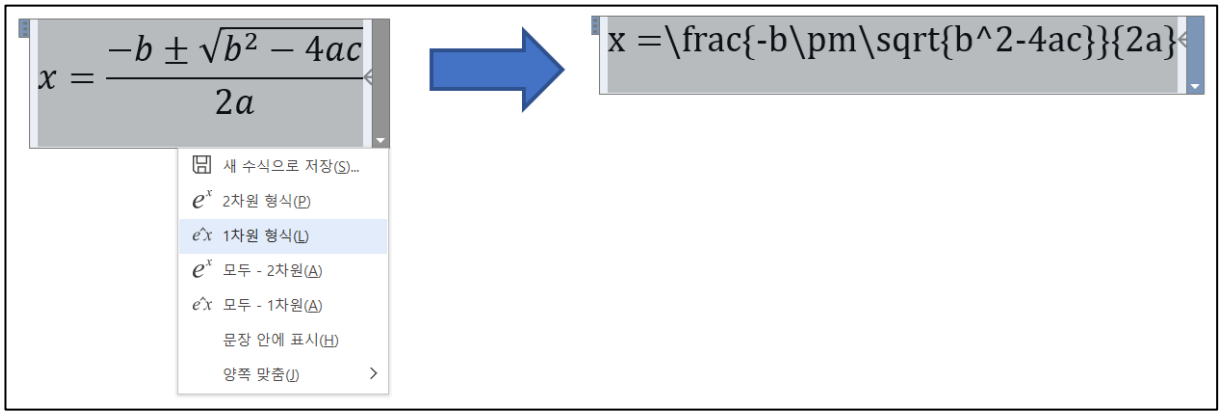

Figure 3-1. WORD 수학 수식 LaTeX 변환

3.8 코드 인용

표현식 : ```코드_형식

코드_내용

```

코드_형식 의 경우 아래와 같이 "code_highlighter -l" 명령어 입력을 통해 아래와 같이 지원하는 코드 형식을 확인 할 수 있습니다.

```
> code_highlighter -l
*** Available current input code format(C:/Project/Profiles/Common/bin/source-highlight-lang/lang.map) list

        C, F77, F90, H, ac, ada, adb, am, applescript, asm
        autoconf, awk, bash, bat, batch, bib, bison, c, caml, cbl
        cc, changelog, clipper, cls, cobol, coffee, coffeescript, conf, cpp, cs
        csh, csharp, css, ctp, cxx, d, desktop, diff, dmd, docbook
        dtx, el, eps, erl, erlang, errors, f, f77, f90, feature
```
```

```
        fixed-fortran, flex, fortran, free-fortran, glsl, go, groovy, h, haskell,
haxe
        hh, hpp, hs, htm, html, hx, hxx, in, ini, ipxe
        islisp, java, javalog, javascript, js, json, kcfg, kdevelop, kidl, ksh
        l, lang, langdef, latex, ldap, ldif, lex, lgt, lhs, lilypond
        lisp, ll, log, logtalk, lsm, lua, ly, m4, makefile, manifest
        mf, ml, mli, moc, opa, outlang, oz, pas, pascal, patch
        pc, perl, php, php3, php4, php5, pkgconfig, pl, pm, po
        postscript, pot, prg, prolog, properties, proto, protobuf, ps, py, python
        r, rb, rc, rs, ruby, s, scala, scheme, scm, scpt
        sh, shell, sig, sl, slang, slsh, sml, spec, sql, sty
        style, syslog, systemverilog, tcl, tcsh, tex, texi, texinfo, tk, tml
        txt, ui, upc, v, vala, vbs, vbscript, verilog, vim, xhtml
        xml, xorg, y, yacc, yaml, yml, yy, zsh
```

아래는 verilog 코드를 인용하는 예를 나타낸다. 인용시 "```" 문자열을 사용하려면, "@```" 로 표기하여 인용할 수 있으며, 코드에 라인 번호를 삽입하려면 '#' 를 붙여 "#코드_형식"으로 표기할 수 있습니다.

예) 라인 번호가 있는 verilog 코드 인용

```
```#verilog
`ifndef __TESTDRIVE_SRAM_SINGLE_V__
`define __TESTDRIVE_SRAM_SINGLE_V__
`timescale 1ns/1ns

module SRAM_Single #(
    parameter   ADDR_WIDTH   = 4,
    parameter   DATA_WIDTH   = 4
) (
    // System
    input   CLK,   // clock
    // SRAM interface
    input   nCE,   // Chip enable (active low)
    input   nWE,   // write enable (active low)
    input   [(ADDR_WIDTH-1):0]   ADDR,   // address
    input   [(DATA_WIDTH-1):0]   DIN,   // input data
    output reg [(DATA_WIDTH-1):0] DOUT   // output data
);
// synopsys template

// definition & assignment ---
reg   [(DATA_WIDTH-1):0]   mem[(2**ADDR_WIDTH)-1:0];

// implementation --
always@(posedge CLK) begin
    if(!nCE) begin
        if(!nWE)
            mem[ADDR]   <= DIN;
        DOUT   <= mem[ADDR];
    end
end

endmodule

`endif//__TESTDRIVE_SRAM_SINGLE_V__
```
```

결과)

```
1: `ifndef __TESTDRIVE_SRAM_SINGLE_V__
2: `define __TESTDRIVE_SRAM_SINGLE_V__
3: `timescale 1ns/1ns
4:
5: module SRAM_Single #(
6:     parameter   ADDR_WIDTH          = 4,
7:     parameter   DATA_WIDTH          = 4
8: ) (
9:     // System
10:    input                           CLK,            // clock
11:    // SRAM interface
12:    input                           nCE,            // Chip enable (active low)
13:    input                           nWE,            // write enable (active low)
14:    input   [(ADDR_WIDTH-1):0]      ADDR,           // address
15:    input   [(DATA_WIDTH-1):0]      DIN,            // input data
16:    output  reg [(DATA_WIDTH-1):0]  DOUT            // output data
17: );
18: // synopsys template
19:
20: // definition & assignment ---------------------------------------------------
21: reg     [(DATA_WIDTH-1):0]      mem[(2**ADDR_WIDTH)-1:0];
22:
23: // implementation -----------------------------------------------------------
24: always@(posedge CLK) begin
25:     if(!nCE) begin
26:         if(!nWE)
27:             mem[ADDR]   <= DIN;
28:         DOUT    <= mem[ADDR];
29:     end
30: end
31:
32: endmodule
33:
34: `endif//__TESTDRIVE_SRAM_SINGLE_V__
```

3.9 코드 실행

표현식 : ```[코드_형식]

코드_내용

```

'코드 인용'과 비슷한 형식이지만, "코드 형식"이 "[]" 로 둘러 싸여 있을 경우, 코드 내용을 실행한다.

현재 사용 가능한 코드 형식은 "lua" 만 지원한다.

예)
```

```
```[lua]
property["Document_Name"] = "UserGuide"
property["IP_Version"] = "1.00"
```
```

3.10 페이지 나눔

표현식 : ;;;

라인 선두에 ';' 문자를 최소 3개 이상 입력하면, 다음 페이지로 넘깁니다.

3.11 수평선

표현식 : ---

라인 선두에 '-' 문자를 최소 3개 이상 입력하면, markdown 과 동일하게 수평선이 입력됩니다.

예)

```
---
```

결과)

3.12 문서 활성화

표현식 #1 : %%% 언어_코드명

표현식 #2 : %%% (Lua_코드)

문서 활성화는 다음 문장들이 실제 문서에 적용할지에 대한 여부를 결정합니다. 문서 활성화에 조건은 "언어_코드명" 또는 소괄호 "()" 로 감싸진 "Lua_코드" 의 반환 값으로 지정할 수 있습니다.

"언어_코드명" 의 경우, '1.3. 실행 방법'에서 지정한 '-l' 옵션에서 지정한 '언어 코드명'과 비교합니다. 만약 같지 않으면, 다음 문서 활성화 지정까지 문서 내용을 무시합니다. 이로써 원하는 언어로 작성된 문장만 문서에 사용할 수 있습니다.

"Lua_코드" 의 경우, 반드시 소괄호 "()" 로 감싸져야 합니다. Lua 코드 실행의 반환 값이 'true'(boolean) 일 경우, 다음 문서 내용을 사용하며, 그외의 값일 경우, 다음 문서 내용은 무시됩니다.

만약 언어 코드나 Lua 코드 없이 '%%%' 로 끝났다면, 다음 줄부터 문서 내용이 활성화됩니다.

NOTE: 언어코드 이름은 대소문자를 구별하며, 언어 코드 이름 뒤에 '-', '=' 또는 '%' 문자들을 나열하여 가독성을 높일 수 있습니다. 기본값은 'en' 을 가집니다. 언어코드는 Lua 에서 'docgen_language' 변수로 사용할 수 있습니다.

예1)

```
%%% ko ------------------------
한글 출력입니다.
%%% en ------------------------
It's an english output.
%%% --------------------------
모든 출력입니다.
```

결과) '-l ko' 옵션 사용시.

```
한글 출력입니다.
모든 출력입니다.
```

예2)

[Lua code]

```
test_mode = true
```

[Markdown code]

```
%%% (test_mode == true)
test_mode 가 활성화 되었습니다.
%%% (test_mode ~= false)
test_mode 가 비활성화 되었습니다.
%%%
```

결과)

```
test_mode 가 활성화 되었습니다.
```

3.13 Lua 함수 호출

표현식 : @<lua:lua_함수>

Lua 함수를 호출합니다. 반환된 값이 문자열 형식일 경우 다음 문서 내용에 적용합니다.

예)

```
Lua 변수 'docgen_language'는 @<lua:docgen_language> 입니다.
```

결과)

Lua 변수 'docgen_language'는 ko 입니다.

3.14 그림 삽입

표현식 : @<img:파일명;스케일;캡션>

그림은 jpg, png, bmp, gif, tif, svg, wmf, vsd/vsdx(Visio 설치 필요) 포멧을 지원하고 있습니다. 파일명은 표시할 파일명을 지정하게 되며, 그림에 외각선 경계가 필요할 경우 "#파일명"으로 표시합니다. Visio 파일일 경우는 "파일명[페이지명]"으로 별도의 페이지 이름을 지정할 수 있으며, 페이지명이 지정되지 않을 때는 첫 페이지로 간주합니다.

또한 스케일의 경우, 종이의 폭 기준 최대값 1.0으로 0 초과 1 이하의 값을 넣어 크기를 지정할 수 있습니다. 지정되지 않을 경우 기본 1.0으로 간주 합니다.

캡션은 말 그대로 캡션 내용을 지정하며, '3.4. 상호 참조'에서 사용될 수 있습니다. 지정되지 않을 경우 캡션은 삽입되지 않습니다.

예1) 1/4 크기의 경계선 있는 도넛 그림

```
@<img:#media/donut.jpg;0.25;도넛 이미지 예시>
```

결과)

Figure 3-2. 도넛 이미지 예시

예2) 경계선 없는 벡터 방식 그림

```
@<img:media/awesome_svg.svg;0.7;SVG 벡터 그림 예시>
```

결과)

Figure 3-3. SVG 벡터 그림 예시

예3) Visio 특정 page를 벡터 이미지로 삽입

```
@<img:media/test.vsdx[test_sample];0.8;Visio 벡터 그림 예시>
```

결과)

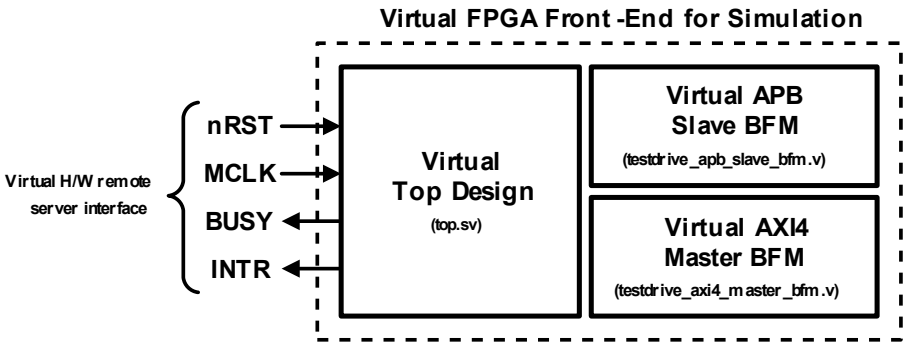

Figure 3-4. Visio 벡터 그림 예시

예4) 그림 없이 캡션만 삽입

```
@<img:nil;0;그림 내용 없이 캡션만 삽입>
```

결과)

Figure 3-5. 그림 내용 없이 캡션만 삽입

3.15 표 삽입

표는 엑셀에 의한 표현 방법과 Lua 스크립트에 의한 표현 방법 2가지를 지원합니다.

표가 매우 길게 기술되어 다음 페이지로 넘어갈 경우, 해더가 자동으로 반복 표기됩니다.

3.15.1 엑셀 파일로부터 표 삽입

표현식 : @<tbl:파일명;시트명;캡션>

표 삽입의 경우 엑셀에서 작상된 내용을 옮겨 넣을 수 있으며, 엑셀상 Merge 된 셀들과 폭은 최대한 동일하게 적용됩니다.

표의 첫 줄은 상단 제목이며, 하위 데이터 열을 가운데 정렬을 하고자 할 때에는 제목 이름 앞에 '*'문자를 삽입합니다.

예1) 입력 엑셀 파일 : "media/table_sample.xlsx"

```
@<tbl:media/table_sample.xlsx;Sheet1;엑셀 표 예시>
```

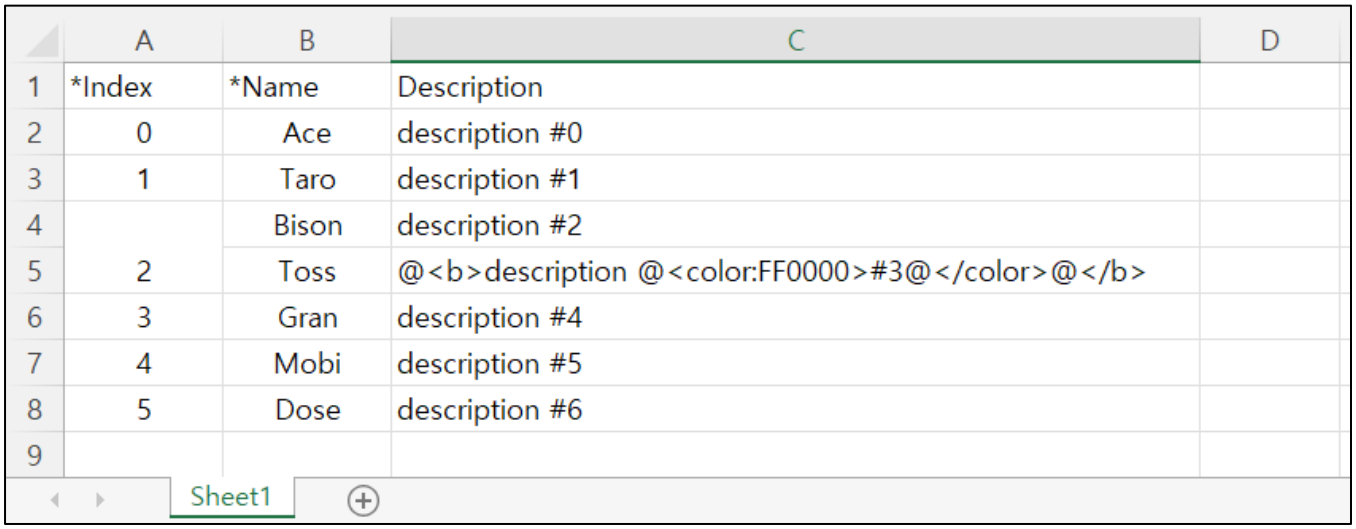

Figure 3-6. "media/table_sample.xlsx" 실제 엑셀 파일

결과)

Table 3-1. 엑셀 표 예시

Index	Name	Description
0	Ace	description #0
1	Taro	description #1
2	Bison	description #2

Index	Name	Description
	Toss	**description #3**
3	Gran	description #4
4	Mobi	description #5
5	Dose	description #6

예2) 테이블 내용 없이 캡션만 삽입

```
@<tbl:nil;nil;표 내용 없이 캡션만 삽입>
```

결과)

Table 3-2. 표 내용 없이 캡션만 삽입

3.15.2 Lua 표현으로부터 표 삽입

표현식 : @<tbl:lua;테이블_변수명;캡션>

기본적으로 docgen 이 Lua 스크립트 기반으로 작성되기 때문에, Lua 변수로 테이블을 만들어 이를 활용하는 방법을 취할 수 있습니다.

본문에서는 이미 아래와 같이 lua 테이블이 정의되어 있는 것을 활용한 예입니다.

예) 선언된 lua 표현

```
lua_table_example = {
    HeaderCount = 2,
    {{"Head A", merge={1,2}}, {"Head B", merge={3,1}}, {width=300,
style="TableTextCenter"}, ""},
    {"", "Sub A", "Sub B", "Sub C"},
    {"index1", "a", "1", "aa"},
    {"index2", "b", "2", "bb"},
    {"index3", {"c", merge={2,1}}, "", "@<color:ff0000>cc@</color>"}
}
```

예) paragraph 내 표현식

```
@<tbl:lua;lua_table_example;Lua 표 예시>
```

결과)

Table 3-3. Lua 표 예시

Head A	Head B		
	Sub A	Sub B	Sub C
index1	a	1	aa

Head A	Head B		
	Sub A	Sub B	Sub C
index2	b	2	bb
index3	c		cc

Lua table 은 "{ { { ... }, ... }, ... }" 와 같이 다중의 중괄호로 표기한다. 맨 외곽의 중괄호는 Table을 기술함을 말하며, 다음 중괄호는 Row, 그 다음 중괄호는 Column 을 표기한다.

Column 과 Row 를 표기시 아래와 같은 속성을 지정할 수 있으며, Column 의 경우 문자열로만 기술할 경우 중괄호를 생략할 수 있다.

- Lua table Row 속성
  - HeaderCount : 해더 줄 수를 지정한다. (기본 : 1)
- Lua table Column 속성
  - data : 셀의 문자열 데이터를 지정한다. "data=" 를 생략하고, 문자열만으로 기술하는 것이 가능하다. 단 생략시 셀 기술의 첫 번째 속성이어야 한다. (기본 : " ")
  - width : 하나의 컬럼 가로 길이를 지정한다. 모든 컬럼의 길이 총합에서 나뉘어 최종 상대적 길이가 지정된다. 이 속성은 첫 Row 에 기술된 Column 에 대해서만 유효하다. (기본 : 100)
  - style : 컬럼에 적용할 텍스트 스타일 이름을 지정한다. 지정된 스타일은 하위 동일 컬럼에도 동일하게 적용된다. (기본 : "TableTextLeft", 스타일을 다시 기본으로 지정할 경우 nil 값이 아닌 "" 을 지정한다.)
  - merge : 현재 셀 위치부터 일정 범위의 다중 셀을 병합하는 기능을 한다. 속성에서 범위 지정은 **{width,height}**로 지정할 수 있다.