pymaketool

Release 2.0.6

Apr 17, 2022

Contents

Release v2.0.6. (*Installation*)

pymaketool is an elegant and simple tool to generate a C project with GNU Make files.

Behold, the power of pymaketool

```
# app_mk.py
from pymakelib import module

def getSrcs(m: module.ModuleHandle):
   return m.getAllSrcsC()

def getIncs(m: module.ModuleHandle):
   return m.getAllIncsC()
```

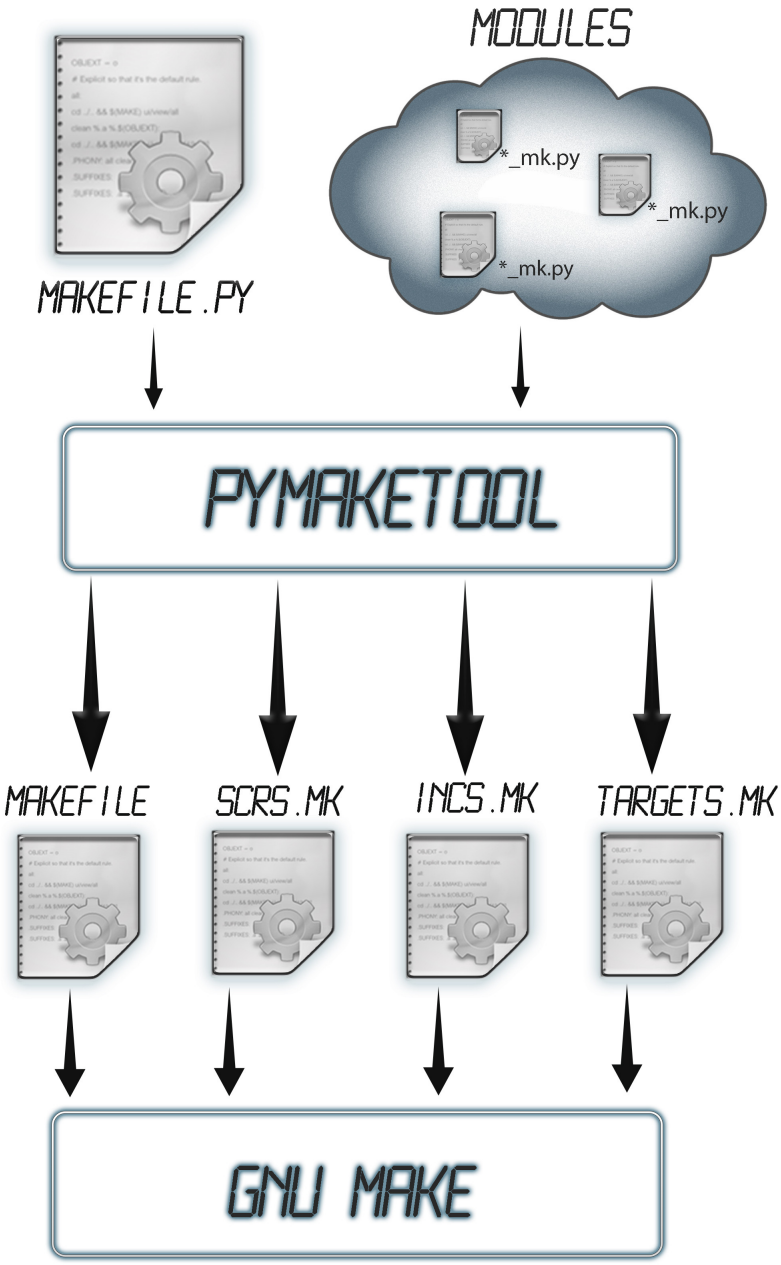
Or in class mode:

```
# app_mk.py
from pymakelib import module

@module.ModuleClass
class mod(module.BasicCModule):
   pass
```

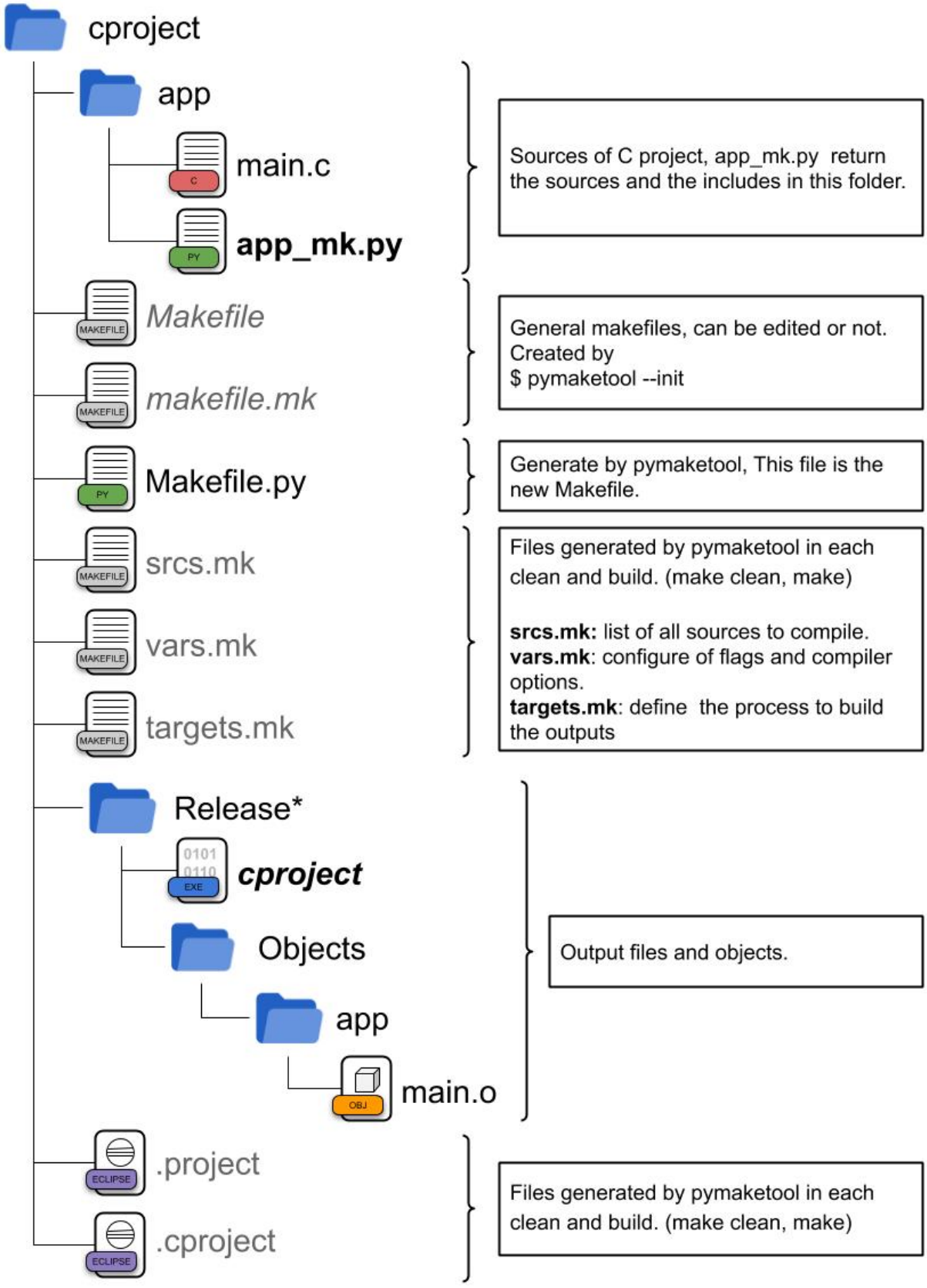
Load remote module:

```
# extlib_mk.py
from pymakelib import module

@module.ModuleClass
class ExtLib(module.ExternalModule):

   def getModulePath(self)->str:
      # Location of module
      return '/LIBS/module_lib/module_lib_mk.py'
```

pymaketool allow to you create C projects with anything structure extremely easily. Use Eclipse IDE for open and edit your project, pymaketool generates the necessary files for this.



Structure of un pymaketool project


cproject
app
main.c
app_mk.py
Sources of C project, app_mk.py return the sources and the includes in this folder.
Makefile
makefile.mk
General makefiles, can be edited or not. Created by
$ pymaketool --init
Makefile.py
Generate by pymaketool, This file is the new Makefile.
srcs.mk
vars.mk
targets.mk
Files generated by pymaketool in each clean and build. (make clean, make)
srcs.mk: list of all sources to compile.
vars.mk: configure of flags and compiler options.
targets.mk: define the process to build the outputs
Release*
cproject
Objects
app
main.o
Output files and objects.
.project
.cproject
Files generated by pymaketool in each clean and build. (make clean, make)

CHAPTER 1

Installation of pymaketool

This part of the documentation covers the installation of pymaketool. The first step to using any software package is getting it properly installed.

1.1 Ubuntu/debian

```
$ sudo apt install -y python3 python3-pip python3-gi python3-gi-cairo gir1.2-gtk-3.0␣
↪git time zip
$ pip3 install pymaketool
```

1.2 Fedora

```
$ sudo dnf install python3-gobject gtk3
$ sudo dnf install python3-pip
$ pip3 install pymaketool
```

1.3 Arch Linux

```
$ sudo pacman -S python-gobject gtk3
$ sudo pacman -S python-pip
$ pip install pymaketool
```

1.4 macOS

```
$ brew install pygobject3 gtk+3
$ brew install python3
$ pip3 install pymaketool
```

1.5 Get the Source Code

pymaketool is actively developed on GitHub, where the code is always available.

You can either clone the public repository

```
$ git clone https://github.com/ericsonj/pymaketool.git
```

Or, download the tarball

Once you have a copy of the source, you can embed it in your own Python package, or install it into your site-packages easily

CHAPTER 2

Makefile.py

Makefile.py is used to build **vars.mk** and **targets.mk**.

Example of Makefile.py for build a linux application:

```
from os.path import basename
from pymakelib import MKVARS
from pymakelib import toolchain as tool

def getProjectSettings():
    """
    Return the project settings.

    Returns:
        dict: with keys PROJECT_NAME and FODLER_OUT
    """
    return {
        'PROJECT_NAME': basename(os.getcwd()),
        'FOLDER_OUT':   'Release/Objects/'
    }

def getTargetsScript():
    """
    Return the of targets
    """
    PROJECT_NAME = basename(os.getcwd())
    FOLDER_OUT = 'Release/'
    TARGET = FOLDER_OUT + PROJECT_NAME

    TARGETS = {
        # target
        'TARGET': {
            # key of target
            'LOGKEY':  'OUT',
            # Name of output file
            'FILE':    TARGET,
```

(continues on next page)

(continued from previous page)

```
            # Script to generate de output file
            'SCRIPT':  [MKVARS.LD, '-o', '$@', MKVARS.OBJECTS, MKVARS.LDFLAGS]
        },
        'TARGET_ZIP': {
            # key of target
            'LOGKEY':   'ZIP',
            # Name of output file
            'FILE':     TARGET + '.zip',
            # Script to generate de output file
            'SCRIPT':   ['zip', TARGET + '.zip', MKVARS.TARGET]
        }
    }

    return TARGETS


def getCompilerSet():
    """
    Return the compilet set.

    Returns:
        dict with path of executables:
        'CC', 'CXX', 'LD', 'AR', 'AS', 'OBJCOPY', 'SIZE', 'OBJDUMP'.
    """
    return tool.confLinuxGCC()


LIBRARIES = [ '-lpthread']

def getCompilerOpts():
    """
    Return all compiler options.

    Returns:
        dict with:
        KEY: name of group of options
        VALUE: list of options
    """
    PROJECT_DEF = {
        'HAVE_CONFIG_H':  None
    }

    return {
        'MACROS': PROJECT_DEF,
        'MACHINE-OPTS': [
        ],
        'OPTIMIZE-OPTS': [
        ],
        'OPTIONS': [
        ],
        'DEBUGGING-OPTS': [
            '-g3'
        ],
        'PREPROCESSOR-OPTS': [
            '-MP',
            '-MMD'
        ],
```

(continues on next page)

(continued from previous page)

```
        'WARNINGS-OPTS': [
        ],
        'CONTROL-C-OPTS': [
            '-std=gnu11'
        ],
        'GENERAL-OPTS': [
        ],
        'LIBRARIES': LIBRARIES
    }


def getLinkerOpts():
    """
    Return all linker options.

    Returns:
        dict with:
        KEY: name of group of options
        VALUE: list of options
    """
    return {
        'LINKER-SCRIPT': [
        ],
        'MACHINE-OPTS': [
        ],
        'GENERAL-OPTS': [
        ],
        'LINKER-OPTS': [
        ],
        'LIBRARIES': LIBRARIES
    }
```

Example of Makefile to build firmware for STM32F4 microcontroller:

```
import os
from os.path import basename
from pybuild import MKVARS

PROJECT_NAME = basename(os.getcwd())
FOLDER_OUT = 'Release/stm32f4-sandbox/'

TARGET_ELF = FOLDER_OUT + PROJECT_NAME + '.elf'
TARGET_HEX = FOLDER_OUT + PROJECT_NAME + '.hex'
TARGET_MAP = FOLDER_OUT + PROJECT_NAME + '.map'
TARGET_BIN = FOLDER_OUT + PROJECT_NAME + '.bin'


def getProjectSettings():
    return {
        'PROJECT_NAME': PROJECT_NAME,
        'FOLDER_OUT':   FOLDER_OUT,
    }


def getTargetsScript():

    TARGETS = {
```

(continues on next page)

(continued from previous page)

```
        'TARGET': {
            'LOGKEY':  'LD',
            'FILE':    TARGET_ELF,
            'SCRIPT':  [MKVARS.LD, '-o', '$@', MKVARS.OBJECTS, MKVARS.LDFLAGS]
        },
        'TARGET_HEX': {
            'LOGKEY':   'HEX',
            'FILE':     TARGET_HEX,
            'SCRIPT':   [MKVARS.OBJCOPY, '-O', 'ihex', MKVARS.TARGET, TARGET_HEX]
        },
        'TARGET_BIN': {
            'LOGKEY':   'BIN',
            'FILE':     TARGET_BIN,
            'SCRIPT':   [MKVARS.OBJCOPY, '-O', 'binary', MKVARS.TARGET, TARGET_BIN]
        }
    }

    return TARGETS


def getCompilerSet():
    pfx = 'arm-none-eabi-'
    return {
        'CC':       pfx + 'gcc',
        'CXX':      pfx + 'g++',
        'LD':       pfx + 'gcc',
        'AR':       pfx + 'ar',
        'AS':       pfx + 'as',
        'OBJCOPY':  pfx + 'objcopy',
        'SIZE':     pfx + 'size',
        'OBJDUMP':  pfx + 'objdump',
        'INCLUDES': [
            toolchain + 'arm-none-eabi/include',
            toolchain + 'arm-none-eabi/include/c++/8.2.1',
            toolchain + 'arm-none-eabi/include/c++/8.2.1/arm-none-eabi',
            toolchain + 'arm-none-eabi/include/c++/8.2.1/backward',
            toolchain + 'lib/gcc/arm-none-eabi/8.2.1/include',
            toolchain + 'lib/gcc/arm-none-eabi/8.2.1/include-fixed'
        ]
    }


def getCompilerOpts():

    PROJECT_DEF = {
        'USE_HAL_DRIVE':            None,
        'CORE_CM4':                 None,
        'STM32F407xx':              None,
        'DEBUG':                    None,
        'VERSION':                  "0.0.1",
        'STM32F4xx':                None,
    }

    return {
        'MACROS': PROJECT_DEF,
        'MACHINE-OPTS': [
            '-mcpu=cortex-m4',
```

(continues on next page)

(continued from previous page)

```
            '-mfpu=fpv4-sp-d16',
            '-mfloat-abi=hard',
            '-mthumb'
        ],
        'OPTIMIZE-OPTS': [
            '-O0'
        ],
        'OPTIONS': [
            '-ffunction-sections',
            '-fstack-usage',
            '-fdata-sections '
        ],
        'DEBUGGING-OPTS': [
            '-g3'
        ],
        'PREPROCESSOR-OPTS': [
            '-MP',
            '-MMD'
        ],
        'WARNINGS-OPTS': [
            '-Wall'
        ],
        'CONTROL-C-OPTS': [
            '-std=gnu11'
        ],
        'GENERAL-OPTS': [
            '--specs=nano.specs'
        ]
    }


def getLinkerOpts():
    return {
        'LINKER-SCRIPT': [
            '-TSTM32F407VETX_FLASH.ld'
        ],
        'MACHINE-OPTS': [
            '-mcpu=cortex-m4',
            '-mfpu=fpv4-sp-d16',
            '-mfloat-abi=hard',
            '-mthumb'
        ],
        'GENERAL-OPTS': [
            '--specs=nosys.specs'
        ],
        'LINKER-OPTS': [
            '-Wl,-Map='+TARGET_MAP,
            '-Wl,--gc-sections',
            '-static',
            '-Wl,--start-group',
            '-lc',
            '-lm',
            '-Wl,--end-group',
            '-u_printf_float'
        ]
    }
```

2.1 Makeclass

Makefile.py in class mode:

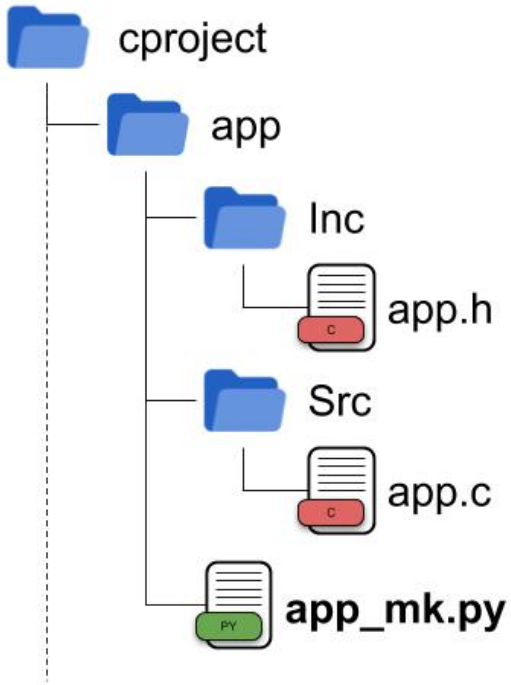
```
from pymakelib import AbstractMake, Makeclass

@Makeclass
class Project(AbstractMake):

    def getProjectSettings(self, **kwargs):
        ...

    def getTargetsScript(self, **kwargs):
        ...

    def getCompilerSet(self, **kwargs):
        ...

    def getCompilerOpts(self, **kwargs):
        ...

    def getLinkerOpts(self, **kwargs):
        ...
```

CHAPTER 3

<name>_mk.py

Python file that find and return the sources to compile in the current path.

The file name must end with **_mk.py**.



app_mk.py sample:

```
from pymakelib.module import ModuleHandle
```

(continues on next page)

(continued from previous page)

```
def init(mh: ModuleHandle):
    """
    Optional function, function that is always executed
    at the beginning of the module.
    """

def getSrcs(mh: ModuleHandle):
    """
    Return the path of all sources in 'cproject/app'

    Returns:
        list of string or Path
    """
    return mh.getAllSrcsC()


def getIncs(mh: ModuleHandle):
    """
    Return the folder of all includes in 'cproject/app'

    Returns:
        list of string or Path
    """
    return mh.getAllIncsC()


def getCompilerOpts(mh: ModuleHandle):
    """
    Optional function, change in the options
    of how these sources are compiled.
    """
    opt = mh.getWorkspace()['compilerOpts']
    opt['CONTROL-C-OPTS'] = ['-std=c89']
    return opt
```

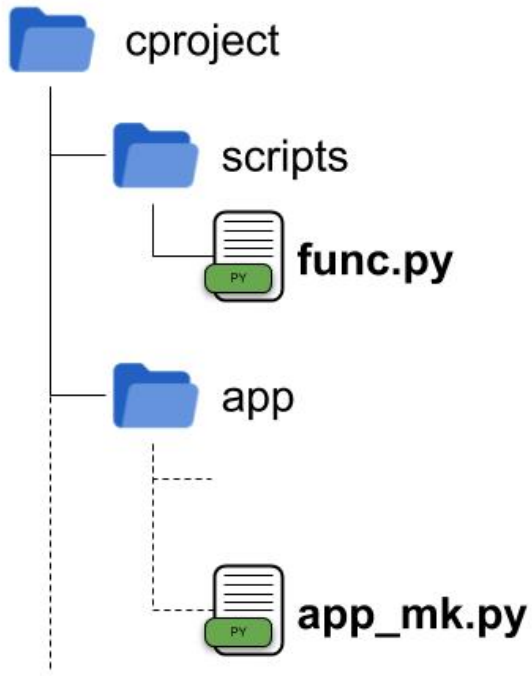
app_mk.py in hardcode mode:

```
from pymakelib.module import ModuleHandle

def getSrcs(mh: ModuleHandle):
    return [
        'app/app.c'
    ]


def getIncs(mh: ModuleHandle):
    return [
        'app'
    ]
```

CHAPTER 4

User scripts

The developer can add more python scripts and import into _mk.py files.



For example in **func.py**:

```
# File func.py

def log(msg):
    print(msg)
```

The **func.py** can import in **app_mk.py**:

```
from pymakelib.module import ModuleHandle
import scripts.func as f


def init(mh: ModuleHandle):
    f.log('Init module app')


def getSrcs(mh: ModuleHandle):
    return [
        'app/app.c'
    ]


def getIncs(mh: ModuleHandle):
    return [
        'app'
    ]
```

CHAPTER 5

pybuildanalyzer2

Util for ARM embedded systems. This utility summarizes memory usage and symbols size.

```
$ pybuildanalyzer2 -h
usage: pybuildanalyzer2 [-h] [-g] [-v] elf

Builder Analyzer for ARM firmware

positional arguments:
elf            ELF file

optional arguments:
-h, --help     show this help message and exit
-g, --gtk      Show in gtk window
-v, --version  show program's version number and exit
```

5.1 Output in console

```
$ ./pybuildanalyzer2 Release/app/app.elf
| Region      | Start          | End            |         Size|         Free|
↪Used             Usage(%) |
| RamLoc32    | 0x10000000     | 0x10008000     |     32.00 KB|     31.77 KB|
↪232 B |          |   0.71% |
| RamLoc40    | 0x10080000     | 0x1008a000     |     40.00 KB|     40.00 KB|
↪ 0 B |          |   0.00% |
| MFlashA512  | 0x1a000000     | 0x1a080000     |    512.00 KB|    502.54 KB|      9.
↪46 KB |          |   1.85% |
| MFlashB512  | 0x1b000000     | 0x1b080000     |    512.00 KB|    512.00 KB|
↪ 0 B |          |   0.00% |
| RamAHB32    | 0x20000000     | 0x20008000     |     32.00 KB|     32.00 KB|
↪ 0 B |          |   0.00% |
| RamAHB16    | 0x20008000     | 0x2000c000     |     16.00 KB|     16.00 KB|
↪ 0 B |          |   0.00% |
```

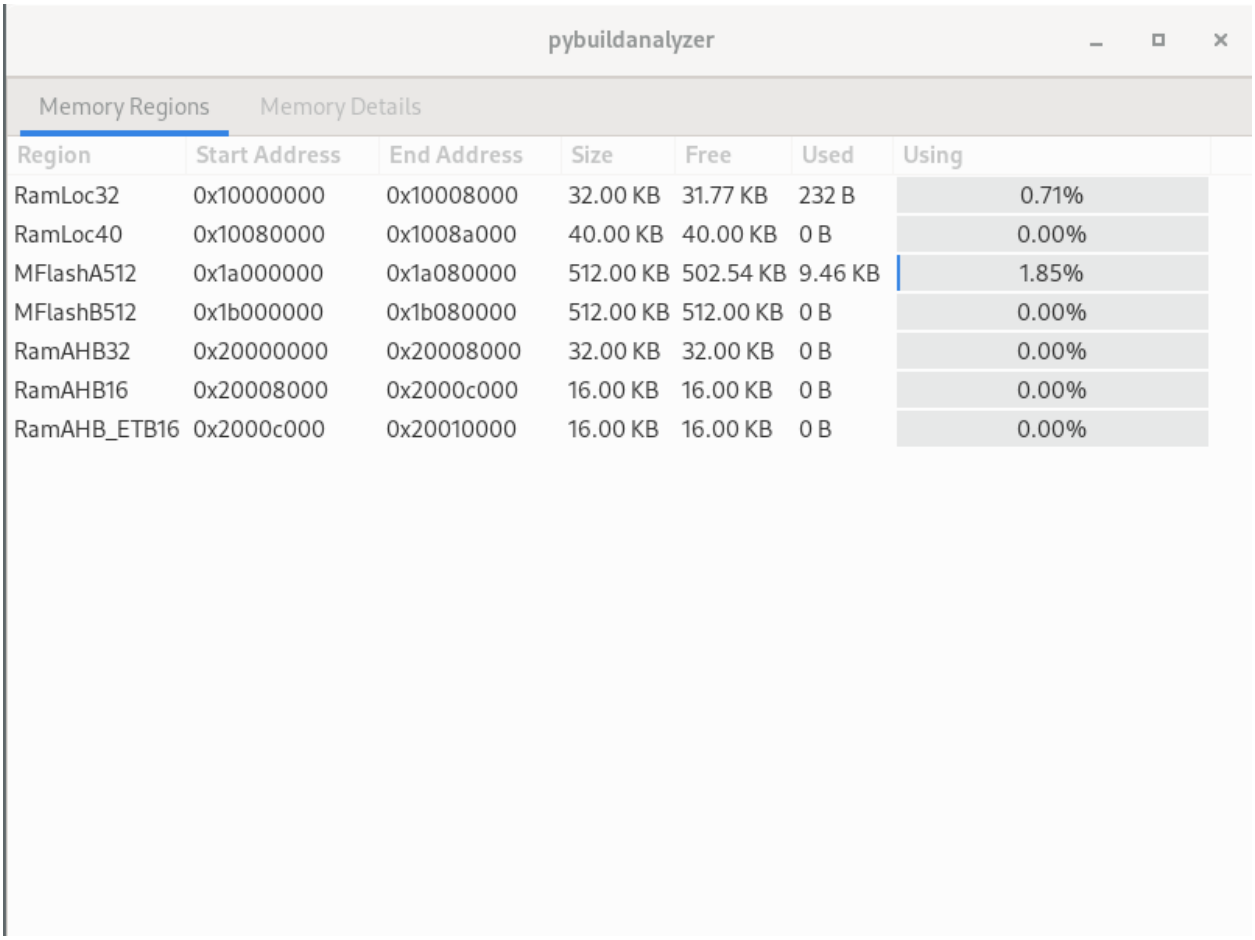
(continues on next page)

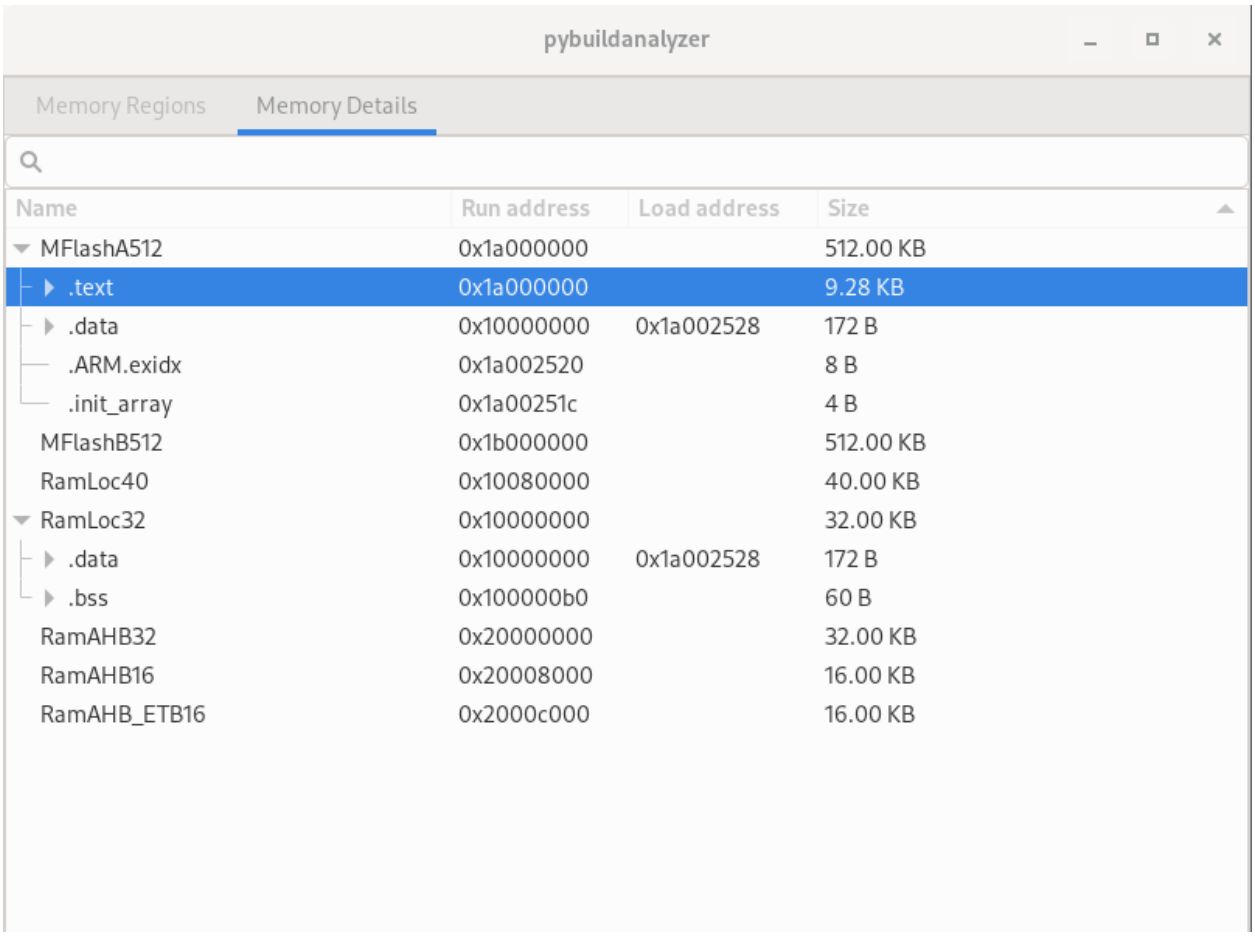
(continued from previous page)

```
| RamAHB_ETB16| 0x2000c000      | 0x20010000      |      16.00 KB|      16.00 KB|
↪ 0 B |           |    0.00% |
```

5.2 Output in GTK

```
$ ./pybuildanalyzer2 -g Release/app/app.elf
```



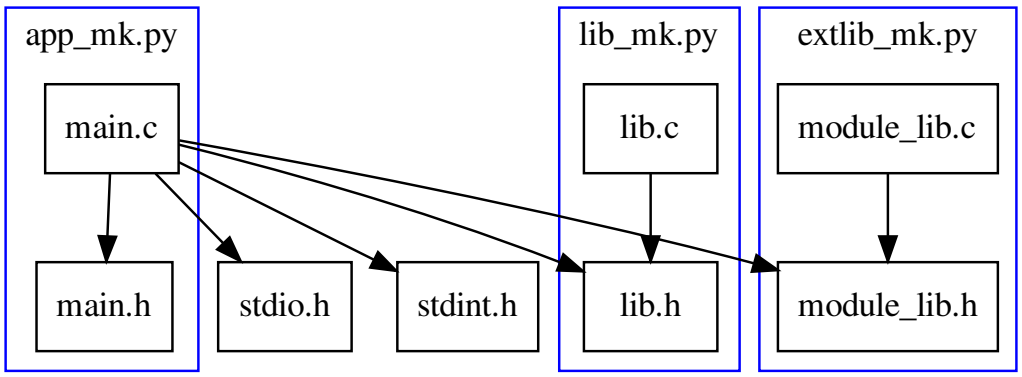

pybuildanalyzer
Memory Regions
Memory Details
Name
Run address
Load address
Size
MFlashA512 0x1a000000 512.00 KB
.text 0x1a000000 9.28 KB
.data 0x10000000 0x1a002528 172 B
.ARM.exidx 0x1a002520 8 B
.init_array 0x1a00251c 4 B
MFlashB512 0x1b000000 512.00 KB
RamLoc40 0x10080000 40.00 KB
RamLoc32 0x10000000 32.00 KB
.data 0x10000000 0x1a002528 172 B
.bss 0x100000b0 60 B
RamAHB32 0x20000000 32.00 KB
RamAHB16 0x20008000 16.00 KB
RamAHB_ETB16 0x2000c000 16.00 KB

CHAPTER 6

pymakedot

This utility create a simple dot file (a.out) of pymaketool modules. e.g.:

```
$ pymakedot app/application/app_mk.py lib/lib_mk.py extlib/extlib_mk.py
$ xdot a.out
```



CHAPTER 7

Addons

pymaketool support addons for extenden functionalities. The addons will be executed before the compilation of the project. Add the next lines en **Makefile.py**

```
from pymakelib import addon

addon.add(yourAddonFunction)
# or
addon.add(yourAddonClass)
```

7.1 Addon function

Simple addon, her entry point is a function with two arguments, for example:

```
def myAddon(projectSettins, compilerSettings):
    """
    Print project and compiler settings
    """
    print(projectSettins)
    print(compilerSettings)
```

7.2 VSCODE Addon

vscode_addon.py is a simple addon that generate c_cpp_properties.json and launch.json

```
import json
import os

def vscodeGen_c_cpp_properties(projSett, compSett):
    """
```

(continues on next page)

(continued from previous page)

```
    Generate file .vscode/c_cpp_properties.json
    """
    defines = []
    for d, v in projSett['C_SYMBOLS'].items():
        if not v is None:
            defines.append(str(d) + "=" + str(v))
        else:
            defines.append(str(d))

    # Change here
    c_cpp_properties = {
        "configurations": [
            {
                'name': 'gcc',
                'defines': defines,
                "compilerPath": compSett['CC'],
                "intelliSenseMode": "linux-gcc-x86",
                "cStandard": "gnu11",
                "cppStandard": "c++17",
                "includePath": projSett['C_INCLUDES'],
                "browse": {
                    "path": projSett['C_INCLUDES'],
                    "limitSymbolsToIncludedHeaders": True,
                    "databaseFilename": "${workspaceFolder}/.vscode/browse.vc.db"
                }
            }
        ],
        "version": 4
    }

    output = json.dumps(c_cpp_properties, indent=4)
    if not os.path.exists('.vscode'):
        os.makedirs('.vscode')
    print("Generate .vscode/c_cpp_properties.json")
    fileout = open(".vscode/c_cpp_properties.json", "w")
    fileout.write("// pymaketool: File autogenerate, see vscode_plugin.py\n")
    fileout.write(output)
    fileout.close()


def vscodeGen_launch(projSett, compSett):
    """
    Generate file .vscode/launch.json
    """
    outputFile = projSett['C_TARGETS']['TARGET']['FILE']
    launch = {
        "version": "0.2.0",
        "configurations": [
            {
                "name": "(gdb) Launch",
                "type": "cppdbg",
                "request": "launch",
                "program": "${workspaceFolder}/" + str(outputFile),
                "args": [],
                "stopAtEntry": False,
                "cwd": "${workspaceFolder}",
                "environment": [],
```

(continues on next page)

(continued from previous page)

```
                "console": "externalTerminal",
                "MIMode": "gdb",
                "setupCommands": [
                    {
                        "description": "Enable pretty-printing for gdb",
                        "text": "-enable-pretty-printing",
                        "ignoreFailures": True
                    }
                ]
            }
        ]
    }

    output = json.dumps(launch, indent=4)
    if not os.path.exists('.vscode'):
        os.makedirs('.vscode')
    print("Generate .vscode/launch.json")
    fileout = open(".vscode/launch.json", "w")
    fileout.write("// pymaketool: File autogenerate, see vscode_plugin.py\n")
    fileout.write(output)
    fileout.close()

def vscode_init(projSett, compSett):
    """
    Entry point of vscode_addon
    """
    # print(projSett)
    # print(compSett)
    vscodeGen_c_cpp_properties(projSett, compSett)
    vscodeGen_launch(projSett, compSett)
```

Add in Makefile.py the line:

```
addon.add(vscode_init)
```

7.3 Addon class

Scalable version for development and extenden addons, for example:

```
from pymakelib import addon

class MyAddon(addon.AddonAbstract):
    """
    Print project and compiler settings
    """
    def init(self):
        print(self.projectSettins)
        print(self.compilerSettings)
```

Add in Makefile.py the line:

```
addon.add(MyAddon)
```

CHAPTER 8

Logger

To enable logger of **pymaketool**, set environment variable LOGLEVEL. Default value is NOTSET.

```
$ export LOGLEVEL=DEBUG
```

Or add paramter in make command.

```
$ make LOGLEVEL=DEBUG
```

CHAPTER 9

pymakelib

9.1 pymakelib package

9.1.1 Submodules

9.1.2 pymakelib.addon module

class pymakelib.addon.**AddonAbstract**(*projectSettings*, *compilerSettings*)
Bases: object

init()

pymakelib.addon.**add**(*inst*)

pymakelib.addon.**init**(*func*)

9.1.3 pymakelib.armsize module

class pymakelib.armsize.**ItemSizeStat**(*name*, *size*, *addr*)
Bases: object

getAddr()

getSize()

string()

pymakelib.armsize.**getSizeAddr**(*line*)

pymakelib.armsize.**main**(*argv*)

pymakelib.armsize.**printKB**(*value*, *decimals=1*)

pymakelib.armsize.**printPrtg**(*value*, *decimals=1*)

9.1.4 pymakelib.ceedling module

pymakelib.ceedling.**getCeedlingHeaderFiles**()

9.1.5 pymakelib.eclipse_addon module

class pymakelib.eclipse_addon.**EclipseAddon**(*projectSettings*, *compilerSettings*)
Bases: *pymakelib.addon.AddonAbstract*

Generate Eclipse cproject files.

generateCProject()

generateLanguageSettings()

init()

9.1.6 pymakelib.eclipse_cproject module

pymakelib.eclipse_cproject.**generate_cproject**(*listconf: dict*)

pymakelib.eclipse_cproject.**generate_languageSettings**(*compilerSettings: dict*)

pymakelib.eclipse_cproject.**writeXmlExcluding**(*excList*)

pymakelib.eclipse_cproject.**writeXmlIncludes**(*incList*)

pymakelib.eclipse_cproject.**writeXmlSymbols**(*symList*)

9.1.7 pymakelib.eclipse_files module

9.1.8 pymakelib.exclude module

9.1.9 pymakelib.git module

pymakelib.git.**getBranchName**()

pymakelib.git.**getCommitHash**(*abbreviated=True*)

pymakelib.git.**getDescribe**(*options='–long'*)

pymakelib.git.**printRelativePath**(*filemacro*)

9.1.10 pymakelib.make_files module

9.1.11 pymakelib.module module

class pymakelib.module.**AbstractModule**
Bases: abc.ABC

Abstract class of pymaketool module

Parameters **path** (*str*) – path to module, _mk.py file.

path
path of module

Type str

findIncs(*inc_type: pymakelib.module.IncType*) → list
Util method for find includes in module path

Parameters **inc_type** (IncType) – Type of includes C or CPP

Returns list of includes paths realtive to project

Return type list

findSrcs(*src_type: pymakelib.module.SrcType*) → list
Util method for find sources in module path

Parameters **src_type** (SrcType) – Type of sources C, CPP or ASM

Returns list of sources paths realtive to project

Return type list

getAllIncsC() → list
Útil method for get all includes in module, type C

Returns list of includes paths realtive to project

Return type list

getAllSrcsC() → list
Util method for get all sources in module, type C

Returns list of sources paths realtive to project

Return type list

getCompilerOpts()
Get special compiler options for module

getIncs() → list
Abstract method to get the includes paths of module

Returns list of includes paths realtive to project

Return type list

getSrcs() → list
Abstract method to get the sources paths of module

Returns list of sources paths realtive to project

Return type list

get_module_name() → str
Module name

Returns name of module (default: class name)

Return type str

get_path()
Get path of module in filesystem

Returns path (default: self.__module__.module_name)

Return type str

init()
Initialization of module

class pymakelib.module.**BasicCModule**
Bases: *pymakelib.module.AbstractModule*

Basic C module, find all sources and includes in module path

Parameters **path** (*str*) – path to module, _mk.py file.

getIncs() → list
return list with all includes in module path

Returns includes path

Return type list

getSrcs() → list
Return list with all sources in module path

Returns sources paths

Return type list

class pymakelib.module.**CompilerOptions**(*opts: dict*)
Bases: object

addOption(*key*, *value*)

setOption(*key*, *value*)

class pymakelib.module.**ExternalModule**
Bases: *pymakelib.module.AbstractModule*

The ExternalModule object that inherits from AbstractModule for include external pymaketool module

Parameters **path** (*str*) – path to module, _mk.py file.

remoteModule
remote module object.

Type *AbstractModule*

Raises AttributeError – path is not valid

getCompilerOpts()
Call and return getCompilerOpts from remote module

Returns compiler options

Return type disct

getIncs()
Call and return getIncs from remote module

Returns list of includes

Return type list

getModulePath() → str
Abstract methos to get string path of external module

Returns path of external module

Return type str

getSrcs()
Call and return getSrcs from remote module

Returns list of sources

Return type list

init()
Call and return init from remote module

Returns may be StaticLibrary object or None

Return type object

class pymakelib.module.**GCC_CompilerOpts**(*copts*)
Bases: *pymakelib.module.CompilerOptions*

addGeneralOpt(*opts: list*)

addMacroOpts(*macro*, *value=None*)

getMacroValue(*macro*)

isDefine(*macro*)

isMacroValue(*macro*, *value*)

setControlCOpts(*opts: list*)

setDebuggingOpts(*opts: list*)

setOptimizationOpts(*opts: list*)

setWarningdOpts(*opts: list*)

class pymakelib.module.**IncType**
Bases: object

C = ['.h']

CPP = ['.h', '.hpp', '.h++', '.hh']

class pymakelib.module.**Module**(*srcs*, *incs*, *flags*, *filename*, *staticLib: pymakelib.module.StaticLibrary = None*)
Bases: object

getDirs()

isEmpty()

pymakelib.module.**ModuleClass**(*clazz*)
Add class to modules of pymaketool

Parameters **clazz** (*class*) – Class inheritance of Module.AbstractModule

class pymakelib.module.**ModuleHandle**(*modDir*, *gCompOpts*, *goal=None*)
Bases: object

getAllIncs(*incType: pymakelib.module.IncType*)

getAllIncsC()

getAllSrcs(*srcType: pymakelib.module.SrcType*)

getAllSrcsC()

getFileByNames(*names*)

getFilesByRegex(*regexs*, *relativePath=None*)

getGeneralCompilerOpts()

getGoal()

getRelaptivePath()

getSrcsByPath(*srcs*)

getWorkspace()

initGitModule = DeprecationWarning(<function ModuleHandle.initGitModule>)

class pymakelib.module.**POJOModule**(*path*)
Bases: *pymakelib.module.AbstractModule*

getCompilerOpts()
Get special compiler options for module

getIncs() → list
Abstract method to get the includes paths of module

Returns list of includes paths realtive to project

Return type list

getSrcs() → list
Abstract method to get the sources paths of module

Returns list of sources paths realtive to project

Return type list

init()
Initialization of module

class pymakelib.module.**SrcType**
Bases: object

ASM = ['.s', '.S', '.asm']

C = ['.c']

CPP = ['.C', '.cc', '.cpp', '.CPP', '.c++', '.cp', '.cxx']

class pymakelib.module.**StaticLibrary**(*name: str*, *outputDir: str*, *rebuild=False*, *lib_linked_opts=None*, *orden=1*)
Bases: object

rebuildByCheckStr(*checkStr: str*)

setRebuild(*rebuild: bool*)

class pymakelib.module.**StaticLibraryModule**
Bases: object

decorate_module()

get_command(*key*) → str

get_lib_name() → str

get_lib_outputdir() → str

get_linker(*key*) → str

get_linker_opts() → str

get_objects(*key*) → str

get_order()

get_rebuild()

get_rule(*key*) → str

pymakelib.module.**cleanModuleInstance**()

pymakelib.module.**getModuleInstance**() → pymakelib.module.AbstractModule

9.1.12 pymakelib.moduleignore module

pymakelib.moduleignore.**readIgnoreFile**(*file=PosixPath('.moduleignore')*)

pymakelib.moduleignore.**writeIgnoreFile**(*ignoreList: list*, *file=PosixPath('.moduleignore')*)

9.1.13 pymakelib.preconts module

9.1.14 pymakelib.prelib module

pymakelib.prelib.**add_value2list**(*dstList: list*, *values*)

pymakelib.prelib.**compilerOptsByModuleToLine**(*compOpts*)

pymakelib.prelib.**list2str**(*l*)

pymakelib.prelib.**macrosDictToString**(*macros*)

pymakelib.prelib.**overrideFile**(*outfile*)

pymakelib.prelib.**readGenHeader**(*headerpath*)

pymakelib.prelib.**readModule**(*modPath*, *compilerOpts*, *goals=None*)

pymakelib.prelib.**read_Makefilepy**(*workpath=''*)

pymakelib.prelib.**read_Makefilepy_obj**(*workpath=''*) → pymakelib.AbstractMake

pymakelib.prelib.**read_module**(*module_path: pathlib.Path*, *compiler_opts*, *goals=None*) → List[pymakelib.module.AbstractModule]

pymakelib.prelib.**tmp_file_name**(*file_path: str*)

pymakelib.prelib.**wprGetCompilerOpts**(*mod*, *modHandle*, *moduleInstance=None*)

pymakelib.prelib.**wprGetIncs**(*mod*, *modHandle*, *moduleInstance=None*)

pymakelib.prelib.**wprGetSrcs**(*mod*, *modHandle*, *moduleInstance=None*)

pymakelib.prelib.**wprInit**(*mod*, *modHandle*, *moduleInstance=None*)

9.1.15 pymakelib.preutil module

pymakelib.preutil.**copyFile**(*file_path_from*, *file_path_to*)

pymakelib.preutil.**getAllIncs**(*wkmh*, *incType: pymakelib.module.IncType*)

pymakelib.preutil.**getAllIncs_C**(*wkmh*)

pymakelib.preutil.**getAllSrcs**(*wkmh*, *srcType: pymakelib.module.SrcType*)

pymakelib.preutil.**getAllSrcs_C**(*wkmh*)

pymakelib.preutil.**getFileHash**(*file_path: str*)

pymakelib.preutil.**getSrcsByRgx**(*wkmh*, **regexs*)

9.1.16 pymakelib.printsrc module

class pymakelib.printsrc.**bcolors**
Bases: object

BOLD = '\x1b[1m'

CBEIGE = '\x1b[36m'

CBEIGE2 = '\x1b[96m'

CBEIGEBG = '\x1b[46m'

CBEIGEBG2 = '\x1b[106m'

CBLACK = '\x1b[30m'

CBLACKBG = '\x1b[40m'

CBLINK = '\x1b[5m'

CBLINK2 = '\x1b[6m'

CBLUE = '\x1b[34m'

CBLUE2 = '\x1b[94m'

CBLUEBG = '\x1b[44m'

CBLUEBG2 = '\x1b[104m'

CBOLD = '\x1b[1m'

CEND = '\x1b[0m'

CGREEN = '\x1b[32m'

CGREEN2 = '\x1b[92m'

CGREENBG = '\x1b[42m'

CGREENBG2 = '\x1b[102m'

CGREY = '\x1b[90m'

CGREYBG = '\x1b[100m'

CITALIC = '\x1b[3m'

CRED = '\x1b[31m'

CRED2 = '\x1b[91m'

CREDBG = '\x1b[41m'

CREDBG2 = '\x1b[101m'

CSELECTED = '\x1b[7m'

CURL = '\x1b[4m'

CVIOLET = '\x1b[35m'

CVIOLET2 = '\x1b[95m'

CVIOLETBG = '\x1b[45m'

CVIOLETBG2 = '\x1b[105m'

CWHITE = '\x1b[37m'

CWHITE2 = '\x1b[97m'

CWHITEBG = '\x1b[47m'

CWHITEBG2 = '\x1b[107m'

CYELLOW = '\x1b[33m'

CYELLOW2 = '\x1b[93m'

CYELLOWBG = '\x1b[43m'

CYELLOWBG2 = '\x1b[103m'

ENDC = '\x1b[0m'

FAIL = '\x1b[91m'

HEADER = '\x1b[95m'

OKBLUE = '\x1b[94m'

OKGREEN = '\x1b[92m'

UNDERLINE = '\x1b[4m'

WARNING = '\x1b[93m'

pymakelib.printsrc.**colorSrc**(*pathsrc*)

9.1.17 pymakelib.project module

pymakelib.project.**define**(*key*) → str
Get value of define if exist.

Parameters **key** (*str*) – name of define (macro)

Returns value of define in string, if define value is None return '',if key is not defined return None

Return type str

pymakelib.project.**getCompilerOpts**() → dict
Get the project compiler options

Returns General project compiler options

Return type dict

pymakelib.project.**getSettings**()

pymakelib.project.**get_base_build**() → str
Get base build folder

Returns base build folder

Return type str

pymakelib.project.**isdefined**(*key*) → bool
Check if project have define.

Parameters **key** (*str or D*) – name of define or macro

Returns True if key is defined

Return type bool

pymakelib.project.**setSettings**(*settings*)

9.1.18 pymakelib.pycodegen module

pymakelib.pycodegen.**HEADER_FILE**(**args*, ***kwargs*)

pymakelib.pycodegen.**comment**(*value: str*)

pymakelib.pycodegen.**enum**(*names, values=[0]*)

pymakelib.pycodegen.**enum_sf**(*strformat*, *range*, *init=0*)

pymakelib.pycodegen.**enum_str_map**(*name*, *strdict: dict*)

pymakelib.pycodegen.**out**(*value*)

9.1.19 pymakelib.toolchain module

pymakelib.toolchain.**confARMeabiGCC**(*binLocation=''*, *prefix='arm-none-eabi-'*, *extIncludes=[]*)

pymakelib.toolchain.**confGCC**(*binLocation=''*, *prefix=''*, *extIncludes=[]*, *iscpp=False*)

pymakelib.toolchain.**confLinuxGCC**(*binLocation=''*, *extIncludes=[]*)

pymakelib.toolchain.**confToolchain**(*cmd_gcc*, *cmd_gxx*, *cmd_ld*, *cmd_ar*, *cmd_as*, *cmd_objcopy*, *cmd_size*, *cmd_objdump*, *cmd_nm*, *cmd_ranlib*, *cmd_strings*, *cmd_strip*, *cmd_cxxfilt*, *cmd_addr2line*, *cmd_readelf*, *cmd_elfedit*, *includes*)

pymakelib.toolchain.**getGCCHeaderFiles**(*cmd_gcc*)

pymakelib.toolchain.**get_c_linux**(*bin_location=''*, *ext_incs=[]*) → dict

pymakelib.toolchain.**get_cpp_linux**(*bin_location=''*, *ext_incs=[]*) → dict

pymakelib.toolchain.**get_gcc_arm_none_eabi**(*binLocation=''*, *prefix='arm-none-eabi-'*, *extIncludes=[]*) → dict

Get dictionary with gcc compiler set of arm-none-eabi-

Parameters

- **binLocation** (*str, optional*) – Location of toolchain binary. Defaults to ‘’.
- **prefix** (*str, optional*) – prefix of ARM toolchain. Defaults to ‘arm-none-eabi-‘.
- **extIncludes** (*list, optional*) – list of external includes. Defaults to [].

Returns set of gcc compiler e.g. {‘CC’: ‘arm-none-eabi-gcc’ . . . }

Return type dict

pymakelib.toolchain.**get_gcc_linux**(*bin_location=''*, *ext_incs=[]*) → dict

Get dictionary with gcc compiler set for linux

Parameters

- **bin_location** (*str, optional*) – location of toolchain. Defaults to ‘’.
- **ext_incs** (*list, optional*) – list of external includes. Defaults to [].

Returns set of gcc compiler e.g. {‘CC’: ‘gcc’ . . . }

Return type dict

pymakelib.toolchain.**get_gpp_linux**(*bin_location=''*, *ext_incs=[]*) → dict

Get dictionary with g++ compiler set for linux

Parameters

- **bin_location** (*str, optional*) – location of toolchain. Defaults to ‘’.
- **ext_incs** (*list, optional*) – list of external includes. Defaults to [].

Returns set of gcc compiler e.g. {‘CC’: ‘g++’ ... }

Return type dict

9.1.20 Module contents

class pymakelib.**AbstractMake**

Bases: abc.ABC

getCompilerOpts(***kwargs*) → dict

getCompilerSet(***kwargs*) → dict

getLinkerOpts(***kwargs*) → dict

getProjectSettings(***kwargs*) → dict

getTargetsScript(***kwargs*) → dict

class pymakelib.**Define**(*value*)

Bases: object

Direct define: { ‘__USE_FILE__’: D(‘file.h’) } => -D__USE_FILE__=file.h

getDefine()

class pymakelib.**Logger**

Bases: object

static getInstance()

Static access method.

static getLogger() → logging.Logger

class pymakelib.**MKVARS**

Bases: object

ADDR2LINE = '$(ADD2LINE)'

CELFEDIT = '$(ELFEDIT)'

CXXFILT = '$(CXXFILT)'

LD = '$(LD)'

LDFLAGS = '$(LDFLAGS)'

NM = '$(NM)'

OBJCOPY = '$(OBJCOPY)'

OBJECTS = '$(OBJECTS)'

PROJECT = '$(PROJECT)'

RANLIB = '$(RANLIB)'

READELF = '$(READELF)'

SIZE = '$(SIZE)'

STATIC_LIBS = '$(SLIBS_NAMES)'

STRINGS = '$(STRINGS)'

STRIP = '$(STRIP)'

TARGET = '$(TARGET)'

pymakelib.**MOD_PATH**(*wk*)

pymakelib.**Makeclass**(*clazz*)

class pymakelib.**Pymaketool**(*workpath='./'*)

Bases: object

getModulesPaths() → list

readModules(*modulesPaths*) → list

read_modules(*modulesPaths*) → List[pymakelib.module.AbstractModule]

pymakelib.**getProjectInstance**() → pymakelib.AbstractMake

Python Module Index

p

Index

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U

W