

Projektstudium im Sommersemester 2019

# Automatisierte Generierung eines Flashtables und Shell-Scriptes anhand von Daten aus einer FlashMap

Vorgelegt von: Jannik Szwajczyk  
Hans-König-Weg 3  
35688 Dillenburg

Matrikelnummer: 5234101

Eingereicht bei  
Hochschulbetreuer: Prof. Dr. Carsten Lucke  
Fachbetreuer: Guy Sagnes

Unternehmen: Continental Automotive GmbH

Eingereicht am: 28.01.2018

## **Sperrvermerk**

Der vorliegende Praxisphasenbericht beinhaltet interne vertrauliche Informationen der Firma Continental Automotive GmbH.

Die Weitergabe des Inhaltes der Arbeit und eventuell beiliegender Zeichnungen und Daten im Gesamten oder in Teilen ist grundsätzlich untersagt. Es dürfen keinerlei Kopien oder Abschriften - auch in digitaler Form - gefertigt werden. Ausnahmen bedürfen der schriftlichen Genehmigung der Firma Continental Automotive GmbH.

## Inhaltsverzeichnis

<b>Abkürzungsverzeichnis</b>	<b>III</b>
<b>Abbildungsverzeichnis</b>	<b>IV</b>
<b>Tabellenverzeichnis</b>	<b>V</b>
<b>1 Einleitung</b>	<b>1</b>
1.1 Continental . . . . .	1
1.2 Abteilung System Software . . . . .	1
1.3 Problemstellung . . . . .	2
1.4 Motivation . . . . .	2
1.5 Ziel . . . . .	2
<b>2 Theoretische Grundlagen</b>	<b>3</b>
2.1 Aufgabenstellung . . . . .	3
2.2 Ausgangslage . . . . .	3
2.3 Definition der Requirements . . . . .	3
2.4 Programmiersprache Python . . . . .	4
2.5 Planung der Arbeitspakete . . . . .	5
2.6 Dokumentation des Projekts . . . . .	6
<b>3 Realisierung der Arbeitspakete</b>	<b>7</b>
3.1 Ergänzung der projektspezifischen Daten in der FlashMap . . . . .	7
3.2 Installation der Python Bibliothek pandas . . . . .	8
3.3 Implementierung der reader-Funktion für die FlashMap . . . . .	9
3.4 Implementierung der reader-Funktion für die Vorlagen . . . . .	9
3.5 Implementierung der handler-Funktion zum Austauschen der Platzhalter . .	9
<b>4 Verifizierung des Codes</b>	<b>10</b>
4.1 Abdeckung der Projektanforderungen . . . . .	10
4.2 Funktionalität des Codes . . . . .	10
4.3 Code Review via GIT . . . . .	11
<b>5 Zusammenfassung und Ausblick</b>	<b>12</b>
<b>Literaturverzeichnis</b>	<b>VI</b>
<b>A Python Programmcode</b>	<b>VII</b>



## **Abkürzungsverzeichnis**

**SSW**     System Software

**TCU**     Telematic control unit

**BU**       Business Unit

**IDE**     Integrated development environment

## Abbildungsverzeichnis

1	5 Divisionen von Continental . . . . .	1
2	Requirements in Excel . . . . .	4
3	Arbeitspakete in Excel . . . . .	5

## **Tabellenverzeichnis**

## 1 Einleitung

### 1.1 Continental

Die Continental AG ist ein weltweit tätiger deutscher Automobilzulieferer mit Hauptsitz in Hannover. Zur Zeit werden circa 240.000 Mitarbeiter weltweit beschäftigt, welche an über 400 Standorten in insgesamt 61 Ländern tätig sind.

Continental ist in die Automotive und Rubber Group aufgeteilt und besteht aus insgesamt fünf verschiedene Divisionen. Die Automotive Group umfasst die Divisionen Chassis & Safety, Powertrain sowie Interior und die Rubber Group Reifen und ContiTech.

Wetzlar ist einer der zahlreichen Automotive Standorte und gehört zu der Divisionen Interior, sowie zu der untergeordneten Business Unit Infotainment & Connectivity.

Der Standort wurde 1955 von Philips gegründet und 2007 von Continental übernommen. Zur Zeit fokussiert sich das Produkt Portfolio von Wetzlar auf die Entwicklung von digitalen Kombiinstrumenten, sowie Telematic control units.

Chassis & Safety	Powertrain	Interior	Tires	ContiTech
Vehicle Dynamics	Engine Systems	Instrumentation & Driver HMI	PLT, Original Equipment	Air Spring Systems
Hydraulic Brake Systems	Transmission	Infotainment & Connectivity	PLT, Repl. Business, EMEA	Benecke-Kaliko Group
Passive Safety & Sensorics	Hybrid Electric Vehicle	Intelligent Transportation Systems	PLT, Repl. Business, The Americas	Compounding Technology
Advanced Driver Assistance Systems (ADAS)	Sensors & Actuators	Body & Security	PLT, Repl. Business, Asia Pacific	Conveyor Belt Group
	Fuel & Exhaust Management	Commercial Vehicles & Aftermarket	Commercial Vehicle Tires	Elastomer Coatings
			Two Wheel Tires	Fluid Technology
				Power Transmission Group
				Vibration Control

PLT – Passenger and Light Truck Tires

Abbildung 1: 5 Divisionen von Continental

### 1.2 Abteilung System Software

Nicht alle Abteilungen am Standort gehören zur Business Unit Infotainment & Connectivity, so zum Beispiel auch das Persistence Team der Abteilung System Software. Diese gehört



zusammen mit anderen Abteilungen zur Business Unit Systems & Technology und arbeitet mit an der Entwicklung eines Cross Domain Hubs.

Das Persistence Team beschäftigt sich in diesem Projekt mit der Sicherung der Daten, sowie der Basissoftware auf der Hardware.

### **1.3 Problemstellung**

Dieser Cross Domain Hub wird in verschiedenen Varianten entwickelt, was die Möglichkeit bieten soll individueller auf Kundenwünsche eingehen zu können.

So unterscheiden sich diese Varianten zum Beispiel in dem zugrunde liegenden Betriebssystem auf der Hardware, sowie in der Anwendungssoftware, welche später auf dem Endprodukt laufen soll.

Dabei werden pro Variante verschiedene Partitionen mit unterschiedlicher Größe im Speicher des Systems reserviert. Diese Daten liegen in einer FlashMap (Excel-Dokument) vor. Um die Informationen nutzen zu können, müssen diese in einen flashtable (c-file), sowie ein shell-script übertragen werden.

Zur Zeit geschieht dieser Prozess händisch und muss für jede Produktvariante und neue FlashMap-Version durchgeführt werden. Jedoch ist diese Verfahren sehr zeitaufwendig und fehleranfällig.

### **1.4 Motivation**

Seit längerer Zeit unterscheiden sich von Kunde zu Kunde die Anforderungen der für das Projekt genutzten Software. Dabei fordern die Kunden oft unterschiedliche Betriebssysteme bei ihrem Produkt. Daran zeichnet sich ab, dass der Cross Domain Hub eine Zukunft besitzt und mit der Automatisierung des Prozesses, auch auf längere Dauer, Zeit eingespart und Fehler vermieden werden können.

### **1.5 Ziel**

Das Ziel des Projektes ist es die Generierung des Flashtables, sowie des Shell-Scripts für die einzelnen Projektvarianten mit einer zentralen Lösung zu automatisieren.

Dabei ist es optimal die Lösung so einfach und übersichtlich wie möglich zu halten, da diese später von einem Kollegen übernommen und weitergeführt werden soll.

## 2 Theoretische Grundlagen

### 2.1 Aufgabenstellung

Es soll eine Möglichkeit gefunden werden aus vorhandenen projektbezogenen Daten, welche in Form einer FlashMap (Excel-Dokument) vorliegen, automatisiert Projektdaten zu generieren. Diese Projektdaten sollen in Form eines FlashTables (C-File), sowie eines Shell-Scripts vorliegen.

Dabei soll die Automatisierung dieses Prozesses die Möglichkeit bieten mehrere Basisvarianten des Produktes abdecken zu können. Das heißt mit verschiedenen Tabellenstrukturen in der FlashMap umgehen zu können, um die vorliegenden Daten zu verarbeiten.

Außerdem soll es möglich sein die vorliegenden Daten in der FlashMap ergänzen zu können, ohne das dabei die Lauffähigkeit des Programmes beeinträchtigt wird.

Dementsprechend sollte das Programm sehr dynamisch geschrieben sein, sodass derartige Änderungen keinen negativen Effekt erzielen. Nach dem Abschluss des Projektes soll die Wartung und Erweiterung des Codes von einem Kollegen übernommen werden.

Dieser besitzt gegebenenfalls keinerlei Erfahrungen in der gewählten Programmiersprache, was auch berücksichtigt werden muss.

Des Weiteren sollen nach Abschluss des Projektes möglichst wenig Rückfragen entstehen.

### 2.2 Ausgangslage

Zur Zeit liegt nur ein Teil der für die Generierung der Dateien benötigten Informationen in der FlashMap vor. Die restlichen Informationen sind lediglich in den zu generierenden Dateien vorhanden.

Außerdem ist die Struktur der FlashMap noch nicht ausreichend strukturiert um die benötigten Daten richtig auslesen und verarbeiten zu können.

### 2.3 Definition der Requirements

Nachdem die Ausgangslage des Projektes gesichtet wurde, konnten die requirements zusammen mit dem Auftraggeber definiert werden. Dabei ist es wichtig darauf zu achten, dass sowohl Auftraggeber, wie auch Entwickler bei jedem requirement die gleiche Auffassung haben.

Oft verstehen Entwickler und Auftraggeber unter dem gleichen requirement unterschiedliche

Ansätze wie dieses verwirklicht werden können. Dies führt dann dazu, dass requirements im Nachgang angepasst oder ganze Codeteile abgeändert werden müssen. Auf Grund der knappen Zeit für das Projekt soll dies durch eine gründliche Absprache vermieden werden.

number	requirement	severity	status
1.	All project data has to be added inside FlashMap so one central file with all infromation exists	S	done
2.	Output of python script shall be the flashtable	S	done
3.	Code shall be dynamic so further additions do not impact executability of the script	S	done
4.	Code shall be kept simple to ensure maintainability	A	done
5.	Output of python script shall be added by shell-script	S	done

Abbildung 2: Requirements in Excel

Hier zu sehen die vorab besprochenen requirements des Projektes. Diese wurden dabei jeweils mit einer Gewichtung versehen.

Die Punkte eins bis vier wurden dabei zu Beginn des Projektes definiert und der Punkt fünf wurde nachträglich ergänzt.

## 2.4 Programmiersprache Python

Als Programmiersprache für das Projekt wurde Python gewählt. Ein Grund hierfür ist die umfangreiche Anbindung der Programmiersprache im Bezug zur Datenverarbeitung von Dateien.

So gibt es bereits diverse Bibliotheken, welche es dem Entwickler ermöglichen Excel-Dateien einzulesen und die enthaltenen Daten relativ simpel zur Verarbeitung zu nutzen.

Ein weiterer Grund für die Wahl von Python sind Vorkenntnisse über die Sprache, welche aus diversen vergangenen Projekten erlangt worden sind. Dementsprechend muss keine umfangreiche Einarbeitung in die Programmiersprache stattfinden, was den zeitlichen Rahmen des Projektes entlastet.

Dieser Punkt hat auch maßgeblich zur Wahl der Bibliothek, welche die Anbindung zur Excel-Datei bereitstellt, beeinflusst. Die open source Bibliothek wurde bereits in vorherigen Projekten zur Realisierung der Datenverarbeitung von Excel-Dateien genutzt und soll deshalb auch hier seine Anwendung finden.

## 2.5 Planung der Arbeitspakete

Task	Time		Status
	estimated	taken	
Write a function that reads the contents of the FlashMap and returns it as a dictionary	7h	14h	done
Write a function that reads the contents of a template file and returns it	1h	0.5h	done
Write a handler that traverses through content of the template and replaces placeholders with actual content	21h	28h	done
Feature- /Change request			
Write new handler that traverses through shell-template and replaces placeholders	2h	1.5h	done
Change handler to call individual handler for either C-File or Shell-Script	0.5h	0.5h	done
Move replacement of placeholders to individual write functions	7h	7h	done

Abbildung 3: Arbeitspakete in Excel

Nachdem nun alle Bedingungen für den Start des Projektes geklärt waren, konnten die Arbeitspakete geplant werden.

Die Arbeitspakete geben die Möglichkeit vorab das Projekt zu planen und dabei direkt benötigte Bearbeitungszeiten mit einzutragen. So besteht ein besserer zeitlicher Überblick über das Projekt, außerdem gibt es immer ein direkt geplantes Ziel, auf welches hingearbeitet wird.

Da bereits bei den requirements Änderungen während des Projektes vorgenommen wurden, haben sich auch dementsprechend die Arbeitspakete erweitert.

Außerdem gab es nach einem Code Review Änderungswünsche bezüglich eines Code Abschnittes.

So musste zum Beispiel noch eine neue handler-Funktion ergänzt werden, da der Code um das requirement des Shell-Scripts erweitert worden ist.

## 2.6 Dokumentation des Projekts

Die Dokumentation des Projektes soll, wie für die meisten internen Projekte über die Software GIT erfolgen. GIT ist ein Versionsverwaltungssystem für Software.

Die Dokumentation des Projektes mit GIT bringt einige Vorteile mit sich. Zum Einen ist es möglich den neu geschriebenen Code direkt hoch zu laden, sodass immer ein aktueller Stand der Software vor liegt. So kann Jeder zu jedem Zeitpunkt auf den aktuellsten Stand der Software zugreifen.

Ein weiter Grund ist die Möglichkeit seinen Code von einem Kollegen überprüfen zu lassen. Dabei wird ein Pull request von dem Entwickler-Branch auf den Master gestellt und der prüfende Kollege als Reviewer eingetragen. Dieser erhält somit eine Benachrichtigung, dass er die Änderungen überprüfen soll. So erhält man schnell Feedback zu seinem Code und Fehler oder unschöne Codeabschnitte können direkt geändert werden.

Außerdem kann das GIT-Repository übersichtlich strukturiert werden, somit können alle benötigten Dateien mit abgelegt werden. Wenn nun irgendwelche Änderungen an den Dateien erfolgen, können diese zusammen mit einem Kommentar hochgeladen werden. Dies ermöglicht eine erhöhte Transparenz während des Projektes.

### 3 Realisierung der Arbeitspakete

Nachdem alle theoretischen Grundlagen definiert und besprochen waren, konnte mit der eigentlichen Umsetzung des Projektes gestartet werden. Für die Durchführung des Projektes wurden die Arbeitspakete der Reihe nach unter Berücksichtigung der requirements, abgearbeitet.

#### 3.1 Ergänzung der projektspezifischen Daten in der FlashMap

Zuerst mussten alle relevanten Informationen aus den vorliegenden Dateien (C-Datei und Shell-Script) identifiziert werden, um diese im Excel-Dokument ergänzen zu können. Dazu wurden die Inhalte der Dateien verglichen um existierende Unterschiede aufzudecken. Allerdings war zu beachten, dass noch nicht für alle Informationen Spalten oder Tabellen im Excel-Dokument vorgesehen waren, da die Daten bisher immer nur in den anderen beiden Dateien vorhanden waren.

Dementsprechend war es kompliziert zu erkennen, welche der Informationen rein spezifische für das Projekt gelten und welche allgemein gültig sind. Um eine bessere Übersicht zu erhalten wurde eine Liste der zu ergänzenden Informationen erstellt.

Nachdem eine erste gründliche Sichtung der Dokumente vorgenommen wurde, fand eine Absprache mit dem Betreuer statt. Dies war notwendig, um sicherstellen zu können, dass sämtliche projektbezogene Informationen identifiziert wurden. Durch einen Hinweis auf fehlende Informationen konnten auch diese der Liste hinzugefügt werden.

Dadurch war es möglich die noch fehlenden Informationen im Excel-Dokument zu ergänzen. Bei diesem Prozess war es notwendig die bereits vorhandenen Tabellenstrukturen abzuändern, zu ergänzen oder sogar neue Tabellen zu erstellen, da die vorhandenen Strukturen nicht die notwendigen Möglichkeiten geboten haben. Wichtig bei dieser Abänderung des Dokuments war die Übersichtlichkeit weiterhin zu gewährleisten, da auch in Zukunft Informationen einfach ergänzbar oder zu finden sein sollen.

Nachdem alle Informationen in das Excel-Dokument integriert wurden, gab es eine erneute Absprache mit dem Auftraggeber, um zu prüfen ob auch dessen Vorstellungen erfüllt worden sind.

### 3.2 Installation der Python Bibliothek pandas

Da nun alle notwendigen Informationen in die FlashMap integriert wurden, konnte mit der Erstellung des eigentlichen Programmes begonnen werden.

Hierzu musste jedoch zuerst noch die für das Projekt ausgewählte Bibliothek pandas für Python 3 installiert werden.

Wie auch viele andere Firmen besitzt Continental einen Proxy Server, welcher dafür sorgt das nicht alle Programme einfach aus dem Internet herunter geladen werden können. Dies sorgt auch beim Installieren von Bibliotheken und Paketen in Python, über das interne Paketverwaltungsprogramm Pip, für Schwierigkeiten. Denn Pip kann keine Verbindung zu externen repositories herstellen um sich die Ressourcen von dort zu installieren.

Um genau diese Problematik umgehen zu können wurde ein interner JFrog Artifactory Server aufgesetzt, welcher unter anderem auch immer die aktuellsten Bibliotheken und Pakete für Python 2 und 3 zur Verfügung stellt. Damit Pip beim Installieren auch den Artifactory Server nutzt, muss dieser in der Initialisierungsdatei von Pip hinterlegt werden.

Auf Windows Systemen ist diese Datei unter %APPDATA%/pip zu finden.

```
[global]

index-url = https://artifactory.geo.conti.de/artifactory/api/pypi/i-st_pd_pypi_v/simple
disable-pip-version-check = true


[install]

trusted-host = artifactory.geo.conti.de
```

Listing 1: Inhalt der Pip Initialisierungsdatei

Nachdem diese Konfiguration in der Initialisierungsdatei vorgenommen wurde, konnte die Bibliothek über die Konsole installiert werden.

```
python -m pip install pandas
```

Listing 2: Installation von pandas über die Konsole

Da zur Entwicklung des Projektes die Integrated development environment (IDE) PyCharm genutzt wurde, hätte man auch hier den Artifactory Server hinterlegen können um Pakete und Bibliotheken für Python zu installieren.

### 3.3 Implementierung der reader-Funktion für die FlashMap

Die Funktion welche als erstes implementiert werden musste, war die reader-Funktion zum Einlesen der Daten aus der FlashMap. Um einen besseren Überblick über die vorhandenen Daten im eingelesenen Format zu erhalten, wurden diese zuerst in der IDE mit Hilfe des Debuggers analysiert. Nachdem ein Grundverständnis für die eingelesene Struktur vorhaben war, konnte die Daten aus dem Excel-Dokument ausgelesen werden. Hierbei stellte sich heraus, dass zum Abspeichern der Informationen aus der Tabelle, Objekte anbieten.

Zum Einen können diese beim Erstellen direkt mit allen benötigten Attributen initialisiert werden, zum Anderen erfolgt der Zugriff auf die Attribute einfach über eine Punktoperation und den Namen des Attributs, was die Lesbarkeit des Codes deutlich steigert.

```
class ExcelEntry:
    """
    This class provides objects for the table entries inside of the FlashMap
    """
    def __init__(self, address, bank, collection, container, flags, img_id, img_size, img_type):
        self.address = address
        self.bank = bank
        self.collection = collection
        self.container = container
        self.flags = flags
        self.img_id = img_id
        self.img_size = img_size
        self.img_type = img_type
```

Listing 3: Beispielhaft der Aufbau einer der Klassen für Excel-Elemente

Die Informationen sind innerhalb des Excel-Dokuments auf verschiedene Blätter verteilt. Um alle Informationen lesen zu können, muss pro Blatt ein Datenframe in Python erstellt werden, worüber auf die Daten zugegriffen werden kann.

```
df_overview = pd.read_excel(filename, 'Overview')
```

Listing 4: Erstellung eines Datenframes

### 3.4 Implementierung der reader-Funktion für die Vorlagen

### 3.5 Implementierung der handler-Funktion zum Austauschen der Platzhalter



## 4 Verifizierung des Codes

Zum Abschluss des Projektes musste noch überprüft werden, ob die vorgegebenen Anforderungen (requirements) an das Programm erfüllt worden sind.

Dies geschieht normalerweise, bei größeren Projekten, über automatisierte Tests. Zum einen über static code check mit Hilfe von Klocwork (einem static code analyzer), welcher beim automatischen Kompilieren der Software via Jenkins mit eingebunden ist. Zum Anderen über Tests, welche direkt gegen die vorgegebenen requirements prüfen.

Allerdings waren diese Maßnahmen beim vergleichsweise geringen Umfang des Programmes nicht notwendig. Somit wurden beim Projekt drei Aspekte berücksichtigt, die Abdeckung der Anforderungen, die Funktionalität des Codes sowie die Qualität des selbigen.

### 4.1 Abdeckung der Projektanforderungen

Das Einzige requirement, welches ohne weitere Tests überprüft werden konnte, war die Ergänzung der notwendigen Projektdaten in der FlasMap. Diese Anforderung wurde jedoch bereits zum Beginn des Projektes validiert (siehe Abschnitt Ergänzung der projektspezifischen Daten in der FlashMap). Somit konnte diese Anforderung als erfüllt gesetzt werden.

### 4.2 Funktionalität des Codes

Die nächsten Anforderungen konnten mit der Kontrolle des Codes auf Funktionalität abgedeckt werden. Dafür wurden die vom Programm erstellten Dateien mit den händisch erzeugten Dateien auf inhaltliche Unterschiede verglichen.

Beim Vergleichen der Dateien sind allerdings Unterschiede aufgefallen. Diese sind jedoch auf Fehler bei den händisch erstellten Dateien zurück zu führen. Bereits hier zeigte sich, dass die automatisiert erstellten Dateien weniger anfällig für Fehler sind.

Somit konnten sowohl die requirements 2. und 5 der der Anforderungsliste auf abgeschlossen gesetzt werden.

Um zu überprüfen ob der Code Anforderung 3. der Liste (Dynamischer Code, welcher nicht durch Ergänzungen in der FlashMap beeinflusst wird) auch erfüllt, mussten Daten in der FlasMap ergänzt werden. Nachdem dies geschehen war wurde das Programm neu ausgeführt, um validieren zu können ob der Code immer noch korrekte Ergebnisse liefert.

Auch dieser Test verlief ohne Probleme und lieferte ein positives Resultat, wodurch auch dieses requirement erfüllt wurde.

### 4.3 Code Review via GIT

Die letzte abzudeckende Anforderung ist die Qualität des geschriebenen Codes. Um diese zu sichern wurde ein Code Review durchgeführt. Dafür wurde das Projekt auf den Entwickler Branch eines selbst erstellten Repository eingchecked.

Das einchecken des Projekts in das Repository erfolgte über die Konsole.

```
git add -A

git commit -m "Initial repository commit"

git push
```

Listing 5: Git Kommandos

Nachdem die Änderungen auf den Server gepusht wurden, konnte ein Pull Request erstellt werden. Dieser sollte die Änderungen vom Entwickler Branch auf den Master mergen. Dabei wurde ein verantwortlicher Kollege als Reviewer hinterlegt und musste erst sein Okay geben, bevor der Code auf den Master Branch gemerged werden konnte.

Somit fand auch das notwendige Review der Code Qualität statt. Nachdem das Review durchgeführt wurde, gab es eine kurze Besprechung bezüglich des Codes in der jeweils gute und schlechte Codeabschnitte besprochen worden sind.

Der Kollege gab sein okay für den Code und der Pull Request konnte geschlossen werden. Somit war auch die letzte Anforderung des Projekts erfüllt.

## **5 Zusammenfassung und Ausblick**

## Literaturverzeichnis

- [1] **Konzernstruktur:** *Willkommen bei der Continental AG*, <https://www.continental-corporation.com/de/unternehmen/konzernstruktur> [abgerufen 09.04.2017]
- [2] **Parthier, Rainer:** *Messtechnik - Grundlagen und Anwendungen der elektrischen Messtechnik*, Springer Vieweg, Wiesbaden, 2016, ISBN 978-3-658-13597-3

## A Python Programmcode

```
try :
    from . import pandas as pd
except:
    import pandas as pd
import re
import sys

class ExcelEntry:
    """
    This class provides objects for the table entries inside of the
    """
    def __init__(self , address , bank , collection , container , flags ,
        self.address = address
        self.bank = bank
        self.collection = collection
        self.container = container
        self.flags = flags
        self.img_id = img_id
        self.img_size = img_size
        self.img_type = img_type

class ExcelHistory:
    """
    This class provides objects for the history entries inside the
    """
    def __init__(self , author , date , description , version):
        self.author = author
        self.date = date
        self.description = description
        self.version = version
```

```
class ExcelContainer:
    """
    This class provides objects for the entries in Container-Sheet
    """
    def __init__(self, id, storage, number, hexid, flags):
        self.id = id
        self.storage = storage
        self.number = number
        self.hexid = hexid
        self.flags = flags
        self.size = ''

    def set_size(self, x):
        self.size = x


class ExcelPartition:
    """
    This class provides objects for the entries in Partition-Sheet
    """
    def __init__(self, name, nr, blocks, range, type):
        self.name = name
        self.nr = nr
        self.blocks = blocks
        self.range = range
        self.type = type


def reader_excel(filename):
    """
    Reads the content from Excel-File into a Dictionary
    :param filename: Path to Excel-File
    :return: Dictionary
    """
    try:
        dict_excel = {}
```

```
# <editor-fold desc="Overview">
# Reads the Overview-Sheet
df_overview = pd.read_excel(filename, 'Overview')
bool_date = False

for i in range(0, df_overview.shape[0] - 1):

    entry = str(df_overview.iloc[i][0])

    if entry == 'Variant':
        dict_excel['Variant'] = df_overview.iloc[i][1]

    elif entry == 'Date':
        bool_date = True
        dict_excel['History'] = []

    elif bool_date and entry != 'nan':
        dict_excel['History'].append(ExcelHistory(str(df_ov
                                                    str(df_ov

    elif bool_date and entry == 'nan:':
        break

del df_overview
del bool_date
# </editor-fold>

# <editor-fold desc="Collections">
# Reads the Collections-Sheet
df_collection = pd.read_excel(filename, 'Collection')
bool_collection = False

for i in range(0, df_collection.shape[0]):

    entry = str(df_collection.iloc[i][0])
```

```
        if entry == 'Collection':
            bool_collection = True
            dict_excel['Collections'] = []

        elif bool_collection and entry != 'nan':
            dict_excel['Collections'].append('COLLECTION_' + str(entry))

        elif bool_collection and entry == 'nan':
            break

    del df_collection
    del bool_collection
# </editor-fold>

# <editor-fold desc="Container">
# Reads the Container-Sheet
df_container = pd.read_excel(filename, 'Container')
bool_container = False

list_containers = []

for i in range(0, df_container.shape[0]):

    entry = str(df_container.iloc[i][1])

    if entry == 'ContainerID':

        bool_container = True

    elif bool_container and entry != 'nan':

        flags = df_container.iloc[i][5].split('\r\n')
        obj_container = ExcelContainer(df_container.iloc[i][2:5],
                                       df_container.iloc[i][6:10])
        list_containers.append(obj_container)
```



```
dict_excel['ContainerTable'] = list_containers

del df_container
del flags
del list_containers
del obj_container
del bool_container
# </editor-fold>

# <editor-fold desc="Partition">
df_partition = pd.read_excel(filename, 'Partition')
dict_excel['Partitions'] = []

for i in range(0, df_partition.shape[0]):

    entry = df_partition.iloc[i][0]

    if entry != '(Reserved for GPT Table)':
        partition = ExcelPartition(df_partition.iloc[i][0],
                                    df_partition.iloc[i][6],
                                    dict_excel['Partitions']).append(partition)

    elif entry == 'nan':
        break

del df_partition
del partition
del entry
# </editor-fold>

# <editor-fold desc="FlashMap">
# Reads the FlashMap-Sheet
df_flashmap = pd.read_excel(filename, 'FlashMap')
bool_container = False
bool_container_group = False
```

```
bool_size = False

dict_excel['Containers'] = []
list_container_areas = []

count_container = 0

for i in range(0, df_flashmap.shape[0] - 1):

    entry = str(df_flashmap.iloc[i][3])

    # Checks if the entry is a Container Group name
    if not bool_container_group and 'Container Group' in entry:

        string_container_group = entry
        bool_container_group = True

        dict_excel[string_container_group] = []

    # Checks if the entry is the Container header of table
    elif not bool_container and 'Container' in entry:

        bool_container = True

    # Checks if an entry under the Container column is read
    elif bool_container and bool_container_group and entry:

        if str(df_flashmap.iloc[i][5]) != 'FREE':
            list_flags = str(df_flashmap.iloc[i][14]).split

            dict_excel[string_container_group].append(Excel

        if entry not in dict_excel['Containers']:
```

```
dict_excel['Containers'].append(str(df_flashmap

if 'FLI' not in str(df_flashmap.iloc[i][9]) and 'FSI'
    bool_size = True

# Checks if a Container Group table has ended
elif bool_container and bool_container_group and entry :

    if not bool_size:

        container_size = 0
    else:

        container_size = str(df_flashmap.iloc[i][8])
    dict_excel['ContainerTable'][count_container].set_s

    count_container += 1
    bool_container_group = False
    bool_container = False
    bool_size = False
    list_container_areas.append(string_container_group)

dict_excel['Container Groups'] = list_container_areas
# </editor-fold>

return dict_excel
except BaseException as e:

    raise e

def reader_template(filename):
    """
    Reads a textfile as input and converts it into an dictionary
    :param filename: Path to file
    :return: Dictionary
```

```
"""
try:

    with open(filename , 'r') as f:
        data_store = f.readlines()

    dict_data = {}
    count = 0

    for entry in data_store:
        dict_data[count] = entry
        count += 1

    return dict_data
except BaseException as e:

    raise e

def string_builder(obj):
    """
    This method builds the string which is later written into the I
    :param obj: Information of corresponding Excel-Entry
    :return: String to write to file
    """
    try:
        # ContainerID
        container_id = '.m_uiContainerId = ' + obj.container + ',\n'

        # Address
        fill_up_digits = '0x'
        for i in range(0, 16 - len(obj.address)):

            fill_up_digits += '0'
        address = '.m_uiAddress = ' + fill_up_digits + obj.address + '\n'
    
```

```
# ImageID
image_id = '.m_uiImageId = ' + obj.img_id + ',\t'

# Collection
if obj.collection != '-':

    collection = '.m_uiCollection = COLLECTION_' + obj.coll
else:

    collection = '.m_uiCollection = IIO_FT_NO_COLLECTION,\t'

# Bank
if '-' not in str(obj.bank):

    bank = '.m_uiBank = ' + str(obj.bank) + ',\t'

else:

    bank = '.m_uiBank = IIO_FT_NO_BANK,\t'

# ImageType
if 'FSN' in obj.img_type:

    image_type = '.m_eImageType = IIO_IMGTYPE_NBX,\t'

else:

    image_type = '.m_eImageType = IIO_IMGTYPE_' + obj.img_t

# ImageSize
# Max length: 25 = 3x 4 digits , 3x UL, 5x Space , 2x *
if int(obj.img_size) > 1024:

    size = int(int(obj.img_size)/1024)

    whitespaces = ''
```

```
        for i in range(0, 4 - len(str(size))):

            whitespaces += ' '

        image_size = str(size) + 'UL * 1024UL * 1024UL,\t'

    else:

        whitespaces = ''

        for i in range(0, 13 - len(str(obj.img_size))):

            whitespaces += ' '

        image_size = str(obj.img_size) + 'UL * 1024UL,\t'

    image_size = '.m_uiImageSize = ' + whitespaces + image_size

    # Flags
    flags = '.m_uiFlags = IIO_FT_IMAGEFLAG_' + obj.flags[0]
    del obj.flags[0]

    for flag in obj.flags:

        flag = 'IIO_FT_IMAGEFLAG_' + flag
        flags += ' | ' + flag
    flags += '},\n'

    return '\t\t\t\t{ ' + container_id + address + image_id + col

except BaseException as e:

    raise e

def write_history(c_file , dict_excel):
```

[illegible]

---



```
def write_collections(c_file , dict_excel):
    """
    Replaces the "-- [COLLECTIONS] --" line in template
    :param c_file:
    :param dict_excel:
    :return:
    """
    for string in dict_excel['Collections']:
        c_file.write('#define ' + string + '\n')

def write_containers(c_file , dict_excel , dict_template , key):
    """
    Replaces the "-- [CONTAINERS] --" line in template
    :param c_file:
    :param dict_excel:
    :param dict_template:
    :param key:
    :return:
    """
    for container in dict_excel['ContainerTable']:

        c_file.write('#define CONTAINER_' + container.id + ' ' + co

def write_variantversion(c_file , dict_excel , value):
    """
    Replaces the "-- [VARIANTVERSION] --" part in line with variant
    :param c_file:
    :param dict_excel:
    :return:
    """
    version = dict_excel['Variant'] + ' v'
    c_file.write(value.replace('-- [VARIANTVERSION] --', version))
```

---

```
:param dict_excel:
:return:
"""
string_table_entries = '\t\t.m_uiNofDefaultImagetableEntries =

for container in dict_excel['Container Groups']:
    string_table_entries += str(len(dict_excel[container])) + '

string_table_entries = string_table_entries.strip(' + ')
string_table_entries += ',\n'
c_file.write(string_table_entries)

def write_imagetable(c_file, dict_excel):
    """
    Replaces the "-- [IMAGETABLE] --" line in template
    :param c_file:
    :param dict_excel:
    :return:
    """
    for container in dict_excel['Container Groups']:

        c_file.write('\t\t\t/* Images on ' + container + ' */\n')

        if len(dict_excel[container]) != 0:

            for obj in dict_excel[container]:
                c_file.write(string_builder(obj))

            c_file.write('\n')

        else:

            c_file.write('\t\t\t/* Free area */\n\n')

    c_file.write('\t\t}\n\t},\n')
```

```
def handler(excel_filename , template_filename , template_shell):
    """
    Handles the writing to the C-File and replacement of placeholders
    :param excel_filename:
    :param template_filename:
    :param template_shell:
    :return:
    """
    try:
        dict_excel = reader_excel(excel_filename)
        dict_template = reader_template(template_filename)
        del excel_filename
        del template_filename

        string_version = dict_excel['History'][len(dict_excel['History'])-1]

        string_c_filename = 'files\\iio_cfg_iip_kilimanjaro_' + string_version
        c_file = open(string_c_filename , 'w')
        del string_c_filename

        for key in dict_template:
            value = dict_template[key]

            if '-- [HISTORY] --' in value:
                write_history(c_file , dict_excel)

            elif '-- [VERSION] --' in value:
                write_version(c_file , string_version)

            elif '-- [MINORVERSION] --' in value:
                write_version_minor(c_file , string_version)

            elif '-- [COLLECTIONS] --' in value:
                write_collections(c_file , dict_excel)
```

```
elif '-- [CONTAINERS] --' in value:
    write_containers(c_file , dict_excel , dict_template ,

elif '-- [VARIANTVERSION] --' in value:
    write_variantversion(c_file , dict_excel , value)

elif '-- [CONTAINERTABLE] --' in value:
    write_containertable(c_file , dict_excel)

elif '-- [TABLEENTRIES] --' in value:
    write_tableentries(c_file , dict_excel)

elif '-- [IMAGETABLE] --' in value:
    write_imagetable(c_file , dict_excel)

else:
    c_file.write(value)

    handler_shell(template_shell , dict_excel , string_version)

except BaseException as e:

    raise e

def handler_shell(shell_filename , dict_excel , version):
    """
    Reads the shell template and creates the new shell-script
    :param shell_filename: Path to shell-template
    :param dict_excel: Dictionary containing information from excel
    :param version: String of current build version
    :return: -
    """
    dict_shell_template = reader_template(shell_filename)
    shell_file = open('files\\iio-mmcc-part-layout.sh' , 'w')
```

```
for key in dict_shell_template:
    value = dict_shell_template[key]

    if '-- [VERSION] --' in value:

        replacement = value.replace('-- [VERSION] --', dict_exc
        shell_file.write(replacement)

    elif '-- [PARTITIONS] --' in value:

        for partition in dict_excel['Partitions']:

            part = 'part' + str(partition.nr) + '_'

            shell_file.write('# Size in 512B blocks: ' + str(pa
            shell_file.write(part + 'name="' + partition.name +
            shell_file.write(part + 'sectors="' + partition.ran
            shell_file.write(part + 'fs="' + partition.type + '

        shell_file.write('nof_partitions=' + str(dict_excel['Pa

    else:
        shell_file.write(value)

def main():
    handler(sys.argv[1], sys.argv[2], sys.argv[3])

if __name__ == "__main__":
    main()

[frame=single, language=Python, basicstyle=]
```

## **Versicherung**

Ich versichere, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Die den benutzten Hilfsmitteln wörtlich oder inhaltlich entnommenen Stellen habe ich unter Quellenangabe kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen.

Wetzlar, 28.01.2018