

PDF形式のハードウェアデータシートの自動処理方法の提案

細合 晋太郎^{1,a)}

概要：組込みシステムにおけるデバイスドライバ開発では、対象とするハードウェアデバイスに関する情報が必要となる。多くのハードウェアデバイスはPDF形式のハードウェアデータシートが提供されており、ソフトウェア開発者はこのデータシートを元にしてデバイスドライバを開発する。PDFはどのような環境でも同一の表示を得ることができ、人が読むには適した形式であるが、文書構造に関する情報を持たないため、プログラムにより情報を抽出し自動処理等に用いることが難しい。本稿では、PDFの解析を行い、構造情報を再構成することにより、PDFから必要な情報を抽出する方法の提案と、利用方法の一例について述べる。

1. はじめに

多くの電子情報の多くが、PDF (Portable Document Format) によって提供されている。PDFは印刷機向けのPostScript[1]から派生したもので、Acrobat社が仕様ならびに実装を提供している。PDF形式は公開されており[2]、サードパーティ製のPDFリーダーも存在する。PDFはリーダーさえあれば、OSやソフトウェアが異なっても、同一の表示が得られ、印刷においてもプリンタが異なっても同様の印刷結果を得ることができる。

このように表示に関しては、有益なデータフォーマットであるが、文章構造を持たないため表現されている内容を自動処理することは難しい。文書構造のメタデータを持ったPDF/X形式もあるが、一般に多く提供されているデータシートのPDFはメタデータを持たないデータフォーマットである。

このようなPDFから構造をもったデータを抽出するには、文字のフォントや位置を元に章や節、表や図といった文章の構造を復元する必要がある。加えて、文書構造から特定の情報を得るためにはその章がどのような情報を表しているかといった、ドメイン知識とのマッピングが必要となる。

本稿ではハードウェアデータシートのレジスタマップの章を対象に、PDFの文書構造解析を行い、レジスタマッ

プの抽出および、ヘッダファイルにおける#define定義のコード生成を行う。

2. ハードウェアデバイスのデータシート形式

ハードウェアデバイスのデータシートには多くの観点の情報が含まれる。多くのデータシートでは、デバイスの提供する機能概要、機能ブロック図、ピンの配置および定義、インタフェース、電気特性および物理特性、操作方法などが含まれる。

このうちソフトウェア開発に必要な情報として、通信方法・操作方法、レジスタ、ピン配置などである。ソフトウェアが実行されるMCUとどのように接続され(ピン配置)、どのような通信規格(インタフェース)で、内部の情報(レジスタ)にどのようにアクセスするか、を知る必要がある。

デバイスは多種多様に渡りデバイスの種類によってデータシートの形式も異なるが、今回は主にI²CやSPI等のシリアルで接続し内部のレジスタにアクセスして制御を行うデバイスを対象とする。

3. PDF解析

PDFのファイルフォーマットは、テキストとバイナリの混在した独自フォーマットとなっている。今回はPDFファイルのアクセスにiText[3]を利用した。iTextはJavaおよびC#向けのPDFライブラリで、PDFの分割・結合といった高レベル操作から、内部オブジェクトへのアクセスといった低レベルの操作まで幅広い機能を提供している。今回は、JavaおよびXtend言語により、抽出プログ

¹ 九州大学 大学院システム情報科学研究所 システム情報科学府高度 ICT 人材教育センター

QUTE:Kyushu University Research Center for Advanced Information and Communication Technology Education

^{a)} hosoai@qito.kyushu-u.ac.jp

ラムを作成した。

3.1 テキスト抽出

iText でテキスト抽出を行う場合、SimpleTextExtractionStrategy といったクラスを用いることで一括変換したテキストを得ることができるが、フォントや位置の情報が欠落してしまう。今回は、テキストの位置とフォント情報をもとに文章構造の再構成を行うため、仮想のレンダラーを実装することで、フォント情報、テキスト位置情報を抽出する。

具体的にはページ毎のバイト列を PdfContentStreamProcessor を介して、RenderListener に渡すことで実現する。RenderListener の renderText が呼ばれる際に、TextRenderInfo というクラスにテキストのオブジェクトが渡され、このオブジェクトがフォントやテキストの位置情報を保持している。フォント情報には、サイズの情報は含まれておらず、テキストの BaseLine (フォントが描画される基準線) および AscentLine (フォントの上端) といった情報をもとに再計算する必要がある。

PDF のテキストは、テキストの位置の情報を持つため、必ずしもページ上のテキストの並びと、ファイル上のテキストの並びは一致しない。このため、テキストの位置情報によって一度ソートを行い、実際のテキストの並びを再現した上で抽出を行う必要がある。また、空白の情報も保持していない箇所があり、テキストオブジェクト間の位置がシングルスペースより大きい場合は空白に変換する必要がある。

3.2 章構成の再構築

章の構成には、フォントの種類およびサイズをもとに、テキストの出現数を計数し、本文と章タイトルを分類する。一番出現率の多いものを本文テキストとし、二番目に多いフォントをタイトルフォントとして、章番号を含むテキストを抽出し、章、項、節として再構築する。

3.3 表の抽出

PDF は表形式のデータ構造を持たないため、罫線とテキストの位置情報から表を再構成する必要がある。罫線の情報には、テキスト抽出よりもさらに低レベルのアクセスが必要となり、ページ毎のバイト列から、PDF の描画オペレータを抽出し、解析する必要がある。今回は、罫線の描画である 'l' オペレータおよび、四角形の描画である 're' オペレータを抽出し、表を再構成した。多くの場合、表の罫線は 're' オペレータで細い四角として描画されていた。

表の再構成では、一旦すべての罫線を抽出し、縦線・横線に分類し、罫線の開始位置、終了位置を元に、表ごとに罫線をグループ化した。その後、罫線の位置情報をもとに、罫線内に含まれるテキストオブジェクトを表のセル要素と

表 1 フォントの出現数 (上位 10 個のみ抽出)

出現数	フォント名	フォントサイズ
10916	LHCMPE+Helvetica	8.021423
2406	Helvetica-Bold	8.051697
1696	LHCMPE+Helvetica	8.794556
663	LHCMPE+Helvetica	7.248268
649	LHCMPE+Helvetica	6.378479
502	Helvetica-BoldOblique	8.051697
271	LHCNGC+Symbol	7.720337
265	Helvetica-Bold	9.700867
225	LHCMPE+Helvetica	8.794571
191	Helvetica-Bold	9.700836

して振り分けることで、表情情報を再構成した。

3.4 抽出情報の利用

抽出した文章構造および表情情報をもとに、コードの自動生成を行う。I²C や SPI で接続するデバイスでは、内部にレジスタを持っており、I²C や SPI でレジスタにアクセスすることで、設定の書き換えやセンサ値の取得を行うものが多い。このようなレジスタにはレジスタの番地を指定しアクセスするが、プログラム内で番地情報を直接記述すると可読性が著しく低下するため、#define によりレジスタ名とレジスタ番地のマッピングを行うことが多い。

今回は抽出したデータシートのレジスタマップをもとに、C のインクルードファイルの #define 句の生成を行った。章タイトルに register mapping 含むページを抽出し、そのページに含まれる表をレジスタマップの表とし、Name 行をレジスタ名、register address 行をアドレスとして抽出し、Xtend のテキストテンプレート機能により、C のヘッダファイルのコードを生成する。

今回の生成では、不具合なくコード生成を行えたが、必ずしもすべてのデータシートで同様の章名、列名であるとは限らない。デバイスの種類や形式、バンダーのフォーマットを考慮した上でドメイン情報とマッピングを行う必要がある。

4. 評価実験

今回はジャイロセンサ L3GD20 のデータシート [4] をサンプルに章構成の抽出およびレジスタマップの抽出とその利用を行う。

4.1 章構成の再構築

L3GD20 のデータシートから得られたフォントの出現数を表 1 に示す。このデータシートでは、LHCMPE+Helvetica の 8pt のフォントが最多となり、このフォントが本文フォントであると推測できる。

また二番目に多い Helvetica-Bold フォントであるが、これを暫定的にタイトルフォントとみなし、同種のフォント

で章番号で始まるテキストをすべて抽出する。

List.1 抽出した章構成（7 章までの抜粋）

1Block diagram and pin description	6
2Mechanical and electrical specifications	8
3Application hints	14
4Digital main blocks	15
5Digital interfaces	21
6Output register mapping	28
7Register description	30
8Package information	41
9Revision history	42
1 Block diagram and pin description	
1.1 Pin description	
2 Mechanical and electrical specifications	
2.1 Mechanical characteristics	
2.2 Electrical characteristics	
2.3 Temperature sensor characteristics	
2.4 Communication interface characteristics	
2.4.1 SPI - serial peripheral interface	
2.4.2 I	
2	
2	
2	
2.5 Absolute maximum ratings	
2.6 Terminology	
2.6.1 Sensitivity	
2.6.2 Zero-rate level	
2.7 Soldering information	
5 Digital interfaces	
5.1 I	
2	
2	
5.1.1 I	
2	
5.2 SPI bus interface	
5.2.1 SPI read	
5.2.2 SPI write	
5.2.3 SPI read in 3-wire mode	
6 Output register mapping	
7 Register description	

今回の例では Table of Contents の目次まで拾ってしまい、章構成だけの抽出ができていない。また 2.4. 2 および、5.1 で I で始まり、その後 2 の行が続いているが、これは I²C の上付き文字のフォントが異なることにより、一つの単語とみなせなかったため、テキストの抽出に失敗している。

4.2 表の抽出

4.1 で得られた章構成より、“register mapping”を含む章を検索し、そのページを得る。今回は、章構成に目次も含まれてしまっていたため、目次部分をスキップしてページ

表 2 抽出したレジスタマップ

Name	Type	Register address		Default
		Hex	Binary	
Reserved	-	00-0E	-	-
WHO_AM_I	r	0F	000 1111	11010100
Reserved	-	10-1F	-	-
CTRL_REG1	rw	20	010 0000	00000111
CTRL_REG2	rw	21	010 0001	00000000
CTRL_REG3	rw	22	010 0010	00000000
CTRL_REG4	rw	23	010 0011	00000000
CTRL_REG5	rw	24	010 0100	00000000
REFERENCE	rw	25	010 0101	00000000
OUT_TEMP	r	26	010 0110	output
STATUS_REG	r	27	010 0111	output
OUT_X_L	r	28	010 1000	output
OUT_X_H	r	29	010 1001	output
OUT_Y_L	r	2A	010 1010	output
OUT_Y_H	r	2B	010 1011	output
OUT_Z_L	r	2C	010 1100	output
OUT_Z_H	r	2D	010 1101	output
FIFO_CTRL_REG	rw	2E	010 1110	00000000
FIFO_SRC_REG	r	2F	010 1111	output
INT1_CFG	rw	30	011 0000	00000000
INT1_SRC	r	31	011 0001	output
INT1_TSH_XH	rw	32	011 0010	00000000
INT1_TSH_XL	rw	33	011 0011	00000000
INT1_TSH_YH	rw	34	011 0100	00000000
INT1_TSH_YL	rw	35	011 0101	00000000
INT1_TSH_ZH	rw	36	011 0110	00000000
INT1_TSH_ZL	rw	37	011 0111	00000000
INT1_DURATION	rw	38	011 1000	00000000

番号を得ている。表 2 に対象ページより抽出したレジスタマップの表を示す。

4.3 抽出情報の利用

4.2 で得られたレジスタマップより、C 言語のヘッダファイルの生成を行う。テンプレートとして用いたコードを List.2 に示す。

[t] List.2 テンプレート（Xtend 言語）

```
def generateHeader(String[] [] table)'''
    #ifndef __L3GD20_H
    #define __L3GD20_H

    <<FOR row : table.drop(2)>>
    <<IF !row.get(0).contains("Reserved")>>
    #define <<row.get(0)>> 0x<<row.get(2)>>
    <<ENDIF>>
    <<ENDFOR>>

    #endif
'''
```

簡易化のためいくつかのパラメータを固定で渡しており、汎用的に用いることはできない。また、4 行目のヘッダ行

のドロップや、5行目の Reserved の除去はテンプレートではなく、予めデータを成型しておいてテンプレートに渡す方が可読性が高い。List.2 によって生成されたコードを List.3 に示す。

List.3 生成コード

```
#ifndef __L3GD20_H
#define __L3GD20_H
#define WHO_AM_I      0x0F
#define CTRL_REG1      0x20
#define CTRL_REG2      0x21
#define CTRL_REG3      0x22
#define CTRL_REG4      0x23
#define CTRL_REG5      0x24
#define REFERENCE      0x25
#define OUT_TEMP       0x26
#define STATUS_REG     0x27
#define OUT_X_L        0x28
#define OUT_X_H        0x29
#define OUT_Y_L        0x2A
#define OUT_Y_H        0x2B
#define OUT_Z_L        0x2C
#define OUT_Z_H        0x2D
#define FIFO_CTRL_REG  0x2E
#define FIFO_SRC_REG   0x2F
#define INT1_CFG       0x30
#define INT1_SRC       0x31
#define INT1_TSH_XH    0x32
#define INT1_TSH_XL    0x33
#define INT1_TSH_YH    0x34
#define INT1_TSH_YL    0x35
#define INT1_TSH_ZH    0x36
#define INT1_TSH_ZL    0x37
#define INT1_DURATION  0x38
#endif
```

今回のような生成した範囲では直接利用することはできないが、本手法を用いて生成したコードを元にしたデバイスドライバで、実機での動作を確認した。

5. 考察

現在の問題点として、図の抽出が難しいことがあげられる。ビットマップデータとして挿入されている図であれば、容易に抽出することが可能であるが、PDF や Postscript 形式の図の場合、3.3 で述べたように描画オペレータ列として図要素が定義されているため、図の境界の抽出や図の再構成が難しい。ただし、図の内部表現も利用することを考えた場合、ビットマップデータを解析するよりもこのようなベクターデータを利用の方が容易である。

テキストの抽出を行う際、ページのヘッダー、フッターの情報が本文に含まれてしまうと、正しいテキスト列として得られないことがある。このため、予めヘッダー、フッターの領域情報を指定し、その部分の情報は除去する処

理を行っている。しかし、ベンダー毎にデータシートのフォーマットが異なるため、ヘッダー、フッターの領域情報はフォーマット毎に指定する必要がある。

章構成を得る方法として、目次のページをテキスト解析することで得ることも可能であるが、目次のフォーマットが必ずしも同じでないことと、目次で示されるページ番号と PDF のページ番号が一致しないこともあり、今回のような方式で章構成の抽出を行った。結果として目次も含む章構成が出力されることとなったが、目次情報の解析も組み合わせることで、より精度の高い抽出が行えるものと思われる。

また、フォントと文書構造のマッピングについても、現在はフォントサイズと出現率に基づいて対応付けているが、フォーマット毎に対応関係をマッピングする方が認識精度が向上すると思われる。PDF 全体の画像データを機械学習することにより、メーカーの判定やページ毎の情報の大まかな分類を行ったのち、本手法に適用することを検討している。

6. おわりに

本稿では、ハードウェアデータシートの PDF を解析し、ソフトウェア開発に必要な情報の抽出する方法の提案を行った。単純な例ではあるが、自動処理によって PDF ファイルからソースコードまで変換できることを確認することができた。

今回はレジスタマップという限られた情報の抽出であったが、他の項目に関しても PDF の解析方法と、データのマッピング方法を変えることで他の形式のデバイスについても応用することができる。また、データシートの解析によって、ハードウェア設計で用いる回路図 CAD のライブラデータの生成にも利用することも可能である。

データシートに限らず、論文や公文書など PDF で提供されている情報は多く、本手法により自動処理可能な領域は多くあると思われる。

参考文献

- [1] Adobe, Adobe PostScript language specifications (online), http://partners.adobe.com/public/developer/ps/index_specs.html (2016.10.12).
- [2] Adobe, PDF Reference and Adobe Extensions to the PDF Specification(online), http://www.adobe.com/jp/devnet/pdf/pdf_reference.html (2016.10.12).
- [3] iText Group NV, iText(online), <http://itextpdf.com/> (2016.10.12).
- [4] STMicroelectronics, L3DG20 datasheet(online), <http://www.st.com/content/ccc/resource/technical/document/datasheet/43/37/e3/06/b0/bf/48/bd/DM00036465.pdf/files/DM00036465.pdf/jcr:content/translations/en.DM00036465.pdf> (2016.10.12).