修士論文 2010 年度 (平成 22 年度)

# RF タグのための補償型トランザクション機構の実現

慶應義塾大学大学院 政策・メディア研究科
米澤 祐紀

*saburo@ht.sfc.keio.ac.jp*

# 修士論文要旨 2010 年度 (平成 22 年度)

# RF タグのための補償型トランザクション機構の実現

Passive 型 UHF 帯 RF(Radio Frequency) タグは，サプライチェーンマネジメント分野を中心に利用が期待されているものの，日本のサプライチェーンは，商社や卸売市場など複数のサプライヤによって構成されるため，サプライヤ間での情報共有に当たっては，サプライヤに依存しない情報共有システムの構築が必要になる．そこで本論文では，RF タグのユーザメモリにデータを蓄積することでサプライヤ間の情報共有を可能にするアーキテクチャについて検討する．

ユーザメモリにデータを蓄積する際には，特定のデータフォーマット (ISO/IEC 15962) に従い，エアインタフェースを用いて書き込みを行うが，ISO/IEC 18000-6TypeC に定義されているエアインタフェースでは，一度に全てのデータを追加することはできない．またデータ追加時には，既存データの読み込みを行う読み込みシーケンスと，書き込みを行う書き込みシーケンス間で，データフォーマットの一貫性を保証しなくてはならない．そのため既存データの上書きを行う必要があり，データ追加はデータ取得に比べ遅くなる．また，書き込み途中に RF タグと RFID リーダライタ間でアクセスができなくなった場合，ユーザメモリに不完全なデータが残存する．その結果，データフォーマットの解析ができなかったり，意図しないデータが取得されてしまったりという問題に繋がる．

この問題を解決するために，本論文では，ユーザメモリ内には完全なデータのみにすることを目的とした，RF タグへのデータ追加時における補償型トランザクション機能を提案した．補償型トランザクション機構はデータ追加時に，データパディング，Handle Manager，バックライトから構成され，解析できないデータフォーマットや意図しないデータの取得を防ぐ．

本論文では，補償型トランザクション機構とそれを組み込んだ RFID ミドルウェアの構築を行い，評価を行った．不完全データの無効化の有効性示すために，不完全データが発生する状況を作り，既存の RFID リーダライタドライバと比較を行った．またデータ追加の実行時間に対し，補償型トランザクション機構が提供するデータフォーマットの一貫性保証の影響を評価するためにパフォーマンス評価を行った．その結果，全ての試行において不完全データの無効化を確認するとともに，データ追加に要する時間を最大約 30 秒を約 0.5 秒に縮められたことを確認した．本論文により，RFID システムにおけるユーザメモリを活用したデータ共有に関する信頼性の確保を実現したことで，今後，より多様な分野における RFID システムの発展が期待できる．

キーワード：
RF タグ，トランザクション，データフォーマット，ミドルウェア，データフォーマットの一貫性

**慶應義塾大学大学院 政策・メディア研究科**
**米澤 祐紀**

# Abstract of Master's ThesisAcademic Year2010

# Compensatable Transaction for RFID

Radio Frequency Tag with UHF band is expected to be used in Supply Chain Management. For instance, in Japan, supply chain is operated by not only one but also many companies, there is a need for exchanging information of the products between them. In order to do that, each of the companies engaging in the process has the responsibility to report relevant information to other companies by adding essential data to the content written inside the RFTag with attached to products.

There are two requirements while writing data into RFTag. The first one is to follow the Data Format of the RFTag, and the second one is to use the air interface. However, two problems thence have arisen. The first problem happens during adding data. The process of adding data into the RFTag needs more than one access to the RFTag to write data into. The RFTag therefore may contain incomplete data fragment which makes the whole data unable to be analyzed. The second one ensures data consistency. It is caused by the requirement of reading the old data contained inside the RFTag and both the old data and the new one should be rewritten in order to add data into it. This fact makes the work of adding data into RFTag extremely time consuming.

To solve above problems, I assume that the incomplete data fragment will be cut off when such errors occur to the RFTag, and propose the method called Compensatable Transaction. Three modules of Compensatable Transaction, Data pad, Handle Manager, and Writing Backward will also be demonstrated. Besides the application of Compensatable Transaction enables the system to collect and terminate all the incomplete data so that only complete data of the RFTag is stored inside User Memory.

This thesis evaluates Compensatable Transaction and shows the validity for cutting off uncomplete data and ensuring data consistency.

Keyword:
RFTag, Data Format, Transaction, RFID-Middleware, Data Format Consistency

**Yuki Yonezawa**

**Graduate School of Media and Governance**
**Keio University**

# 目次

# 図目次

# 表目次

# ソースコード目次

# 第 1 章

# 序論

本章では，本論文の背景として，RFID システムのアーキテクチャについて述べる．そして，RF タグのユーザメモリへのデータ追加時の課題について述べる．

## 1.1 背景

Passive 型 UHF 帯 RF(Radio Frequency) タグは，サプライチェーンマネジメント分野を中心に，資産管理，製品作成工程の管理などでの利用が期待，検討されている [1][2][3][4].

サプライチェーンマネージメントにおける RFID システムの利用に関しては，EPCglobal に代表されるネットワーク型 RFID システムアーキテクチャが検討されてきた [5]．しかし，日本におけるサプライチェーンは，規模の異なるサプライヤによって構成されるため，全てのサプライヤが同じネットワークに接続し，一元管理をすることは難しい.

日本では，東日本旅客道が鉄道の乗車券として RF タグの一種である FeliCa を用いた，Suica システムを提供している．Suica システムでは，FeliCa 内のユーザメモリに改札通過などのデータを蓄積している．[6]．また，改札通過などのデータは，データセンターなどの中央サーバに送信され蓄積される．このように，日本で利用されている RFID システムでは，RF タグ自体にデータを残しかつ全てのデータをセンターのデータベースに蓄積する手法が用いられる [7].

エアバスやボーイングを中心とした ATA(Air Transport Association) では，飛行機でのメンテナンス効率を向上させるために，ユーザメモリの中にデータを格納することを SPEC 2000 で定義し検証している [8].

このように RFID システムは，複数のサービスサプライヤによって情報共有が求められる状況において，RF タグのユーザメモリを利用することにより，ネットワークが繋がらない場所でのシステム運用を可能にしている．また，複数サービスサプライヤ間で情報を共有するためにも，データフォーマットの共通化が必要となる．Q.E.D.Systems の Craig K.Harmon は，物に添付された RF タグは，人から人に渡され，その都度，新しい RFID システムで使われるため，特定の標準に準拠したデータフォーマットを用いる必要があると示唆されている [9].

### 1.1.1 ユーザメモリの利用

ユーザメモリへデータ追加時には，データフォーマットとエアインタフェースに注意する必要がある．本論文では，データフォーマットとして ISO/IEC 15962，エアインタフェースとして ISO/IEC 18000-6TypeC を対象とする.

#### データフォーマット

RF タグ内のデータフォーマットは，ISO/IEC 15962 により定義されている．それぞれのデータにデータの識別子を付与し，1 組にして追加する．追加時には，限られた資源であるユーザメモリを効率よく利用するためデータの圧縮などが行われる．結果として，1 データは最小でも 3 バイト以上でフォーマットティングが行われる.

**エアインタフェース**

ユーザメモリの読み書きを行うエアインタフェースとして，UHF 帯 RF タグでは ISO/IEC 18000-6TypeC が存在し，word 単位 (1word=16bit) でのデータの読み書きを行う．読み込みは，Read コマンドを用いて，複数 word を一度に読み込むことができる．書き込みは，1word 単位の書き込みコマンドの Write コマンドと複数 word 書き込みのコマンドである BlockWrite コマンドの 2 つが存在する．しかし，BlockWrite コマンドはオプション扱いであるため，実質，1word 単位での書き込みとなる．

**課題**

RF タグへのエアインタフェースとデータフォーマットがそれぞれ標準化されているため，以下に示す 2 つの課題が存在する．

1 つ目は，不完全データが残存する課題である．Write コマンドを用いた場合，1 度に書き込みができるデータ量は 2 バイトである．しかし，フォーマッティングされたデータは 3 バイト以上であるため，一度に全てのデータを書き込むことができない．そのため，書き込み途中で RF タグと RFID リーダライタ間でアクセスが不可能となった場合に，不完全なデータが残存してしまう．RF タグと RFID リーダライタ間のアクセスは，距離の制限だけでなく，外部環境により左右される．例えば，RF タグの周りにある金属が原因となって，100% 読み取りができない [10].

2 つ目は，データ追加実行時間の遅延である．データ追加時には，読み込みを行い，既存データの解析と追加するデータのアドレスの計算等を行ってから，書き込みを行う．しかし，データフォーマット上の一貫性を保証するために，既存データの上書きを行わなくてはならない．そのためデータ追加の際には，既存データと追加データの両方を書き込まなくてはならないというペナルティが発生する．

## 1.2 本論文の目的

第 1.1 節で述べたように，複数 RFID システム間での情報共有を行うために，ネットワークが繋がらない状況でも利用可能な RFID システム構築が必要となり，構築するためにユーザメモリが利用される．しかし，データ追加時には，不完全データの残存，データフォーマットの一貫性によるデータ追加時間の大幅な遅延問題が存在するため，複数 RFID システム間で正しいデータ交換が行えないという問題に繋がる．

そこで本論文では，データ追加時における課題を解決するために，ユーザメモリ内に完全なデータのみを残すことを目的とする．この目的を達成することにより，オフラインでも利用可能なモビリティの高い RFID システム・アプリケーションの作成が可能となる．

本論文は，RFID システムで根幹であるミドルウェア発展へ貢献することが予想される．対象としているデータフォーマットは，国際標準である ISO/IEC 15962 を用いており，今後サプライチェーンマネジメントや物品管理，工業製品の管理などの幅広い分野で利用が期待される．

## 1.3 本論文の構成

本論文の構成は以下の通りである．まず第 2 章にて RF タグへのデータ追加に必要な基本シーケンスと目的とする補償型データ追加手法と関連研究について述べる．第 3 章は，補償型トランザクション機構の提案を行う．第 4 章は，その設計を述べ，第 5 章は，その実装について述べる．第 6 章は，評価について述べ，第 7 章で本論文をまとめる．

# 第 2 章

# RF タグへのデータ追加手法

本章では，RF タグへのデータ追加に必要な技術とその課題について述べる．本論文が対象としているエアインタフェースとデータフォーマットの詳細について述べ，RFID ミドルウェアの基本設計について述べる．そして，RFID ミドルウェアのデータ追加処理の詳細シーケンスとその課題を述べ，課題を解決する関連研究を述べる．

## 2.1 エアインタフェース:ISO/IEC 18000-6TypeC

ISO/IEC 18000-6TypeC は，UHF 帯 RF タグへアクセスするためのエアインタフェースを定義している．RF タグへのデータアクセスは word 単位で行われる．ユーザメモリへのアクセスコマンドは，Read コマンド，Write コマンドとして定義されており，各アクションコマンドを実行するためには，図 2.1 に示すように RF タグのステータスを維持するために，Select コマンド，Inventory コマンドが推奨されている．Read コマンド，Write コマンドの実行には，Inventory コマンド実行後に取得できる handle 値が必要となる．

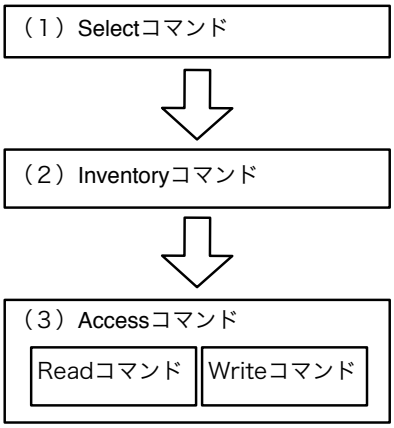


図 2.1　ISO/IEC 18000-6TypeC における基本アクセスコマンドシーケンス

### 2.1.1 Read コマンド

Read コマンドは，複数の word を一度に取得することができる．読み込みが可能な word 数は，RF タグや RFID リーダライタ依存してしまうという課題が存在する．この課題に対して慶應義塾大学の苧阪らは，読み込みの最適化手法を提案している [11].

Read コマンド実行には，メモリバンク，読み込みの開始アドレス，読み込み word 数，handle の値が必要である．

### 2.1.2 Write コマンド

RF タグへのデータ書き込みコマンドとして，1word 単位で書き込みを行う Write コマンドと複数 word 単位で書き込みを行う BlockWrite コマンドが用意されている．しかし BlockWrite コマンドはオプションコマンドであり，また RF タグへの対応も必要とるため利用できない可能性が高い．そこで本論文では，1word 単位の書き込みコマンドである Write コマンドを対象とする．

Write コマンド実行には，メモリバンク，書き込み開始アドレス，書き込みデータ (Write コマ

ンドは 1word, BlockWrite コマンドは複数 word 指定が可能)，handle の値が必要である．

### 2.1.3 Read コマンドと Write コマンドの実行時間の比較

Write コマンドは，メモリの書き換えを行うため，Read コマンドよりも遅いのが一般的である．そこで，Write コマンドと Read コマンドの速度差を明らかにするため，24，48，72，96word ごとに読み書きを行う実験を行った．表 2.1 に，実験で用いた機材を示す．

表 2.1　実装環境

| | |
|---|---|
| RF タグ | $\mu$-Chip Hibiki (ユーザメモリ領域 1536bit) |
| RFID リーダライタ | UHF 帯 RFID 対応 RW モジュール　評価キット [12] |
| アンテナ | 右円偏波型アンテナ |
| プログラミング言語 | Python2.7 |

図 2.2 に，実験結果を示す．x 軸は読み書きを行う word サイズを示している．y 軸は読み書きに要した実行時間である．96word のデータ読み込みには 0.17 秒しか要していないが，書き込みには，3.2 秒かかっており，Write コマンドは Read コマンドの約 19 倍遅いことを確認した．

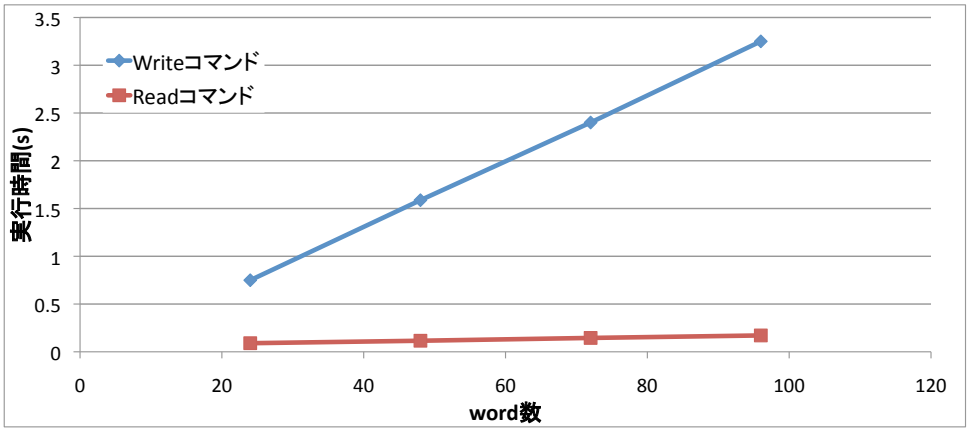


図 2.2　Read コマンドと Write コマンドの比較

## 2.2 データフォーマット:ISO/IEC 15962

本節では，UHF 帯 RF タグに保存する際の ISO/IEC 15962 のデータフォーマットの詳細について述べる．データフォーマットは，データフォーマットの種類を示すアクセスメソッドと実際のデータを保存するデータフォーマットの 2 つに分けられる．

### 2.2.1 アクセスメソッド

データフォーマットは, アクセスメソッドにより異なり, ISO/IEC 15962:2010 Final Committee Draft 版では以下の 4 つが定義されている.

- Non-Directory: 任意のデータの追加が可能なアクセスメソッドであり，以下に示す Directory，Tag-Data-Profile の元となるアクセスメソッド
- Directory: Non-Directory の拡張で，ユーザメモリの最後に各データが書き込まれている物理アドレスの書き込みを行い，読み込み時には，ダイレクトにデータの読み込みができるアクセスメソッド
- Tag-Data-Profile: Non-Directory の拡張で，事前に格納するデータのプロファイルを用意し，値が決まり次第，事前に決められた場所にデータを格納するアクセスメソッド
- Packed-Objects: 事前に登録したフォーマットにより，数字と文字にデータを分けることによりデータの圧縮効率をあげたアクセスメソッド

アクセスメソッドの中で，最もエンドユーザが利用可能なデータフォーマットとして Non-Directoery アクセスメソッドがある．理由として，プロファイルやフォーマットの事前登録が不要で，他のアクセスメソッドの基礎になっていることが挙げられる．そこで本論文では，Non-Directory アクセスメソッドを対象としたデータ追加手法を対象とする.

### 2.2.2 データフォーマット

図 2.3 に，Non-Directory アクセスメソッドを用いて，4 つのデータを追加した様子を示す．データフォーマットは，以下に示す要素により構成されており，図 2.3 に示すように，1 データは Precursor，Object Length，Object の 3 つからなり，先頭アドレスから順次配置される.

- DSFID(Data Storage Format Identifer):利用されているアクセスメソッドを示す.
- Precursor: データの識別子，Object の圧縮方式を示す.
- ObjectLength: データの圧縮後のデータ長を示す.
- Object: 圧縮後のデータを示す.
- END: データの終わりを示す．Non-Directory では”0x00” が終わりを示す.
- Skip: 読み込みを次のデータへスキップを行う．”0x80” で示されて，データの削除などに利用される.

## 2.3 RFID ミドルウェア

本節では，RF タグにデータを追加する RFID ミドルウェアについて述べる．以下に，RFID ミドルウェア上での各標準の位置づけを示す.

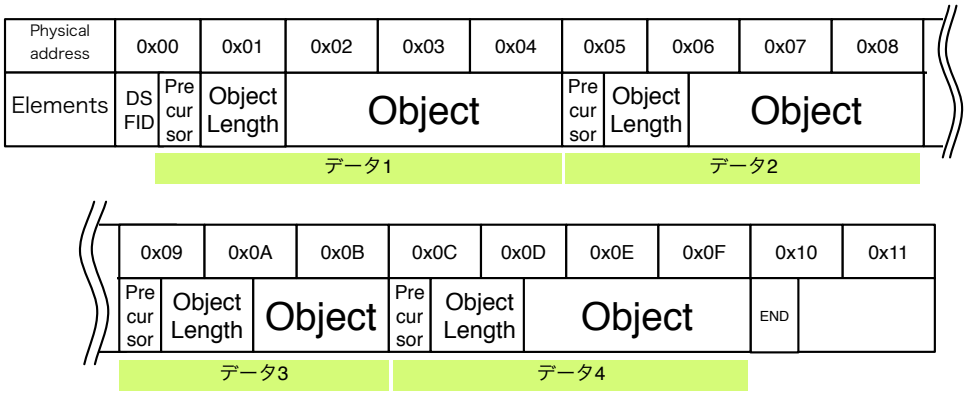


図 2.3　Non-Directory アクセスメソッド利用時のデータフォーマット

- ISO/IEC 15961:RFID システムが RF タグへデータの追加・取得をためのアプリケーションコマンドを定義している．
- ISO/IEC 15962:アプリケーションコマンドで取得したデータをユーザメモリに格納するためのデータフォーマットを定義している．
- ISO/IEC 18000-6TypeC:UHF 帯 RF タグを利用するためのエアインタフェースを定義している．

図 2.4 に，ISO/IEC 15961[13], 15962[14], 18000-6 Type C[15] を利用した際の RFID システム全体のレイヤ構成を示す．RFID システムは次の 3 つで構成される．1 つ目は，アプリケーション層でありアプリケーションロジックが組み込まれる．RF タグへのデータ追加・取得を行う際には，ISO/IEC 15961 で定義されているアプリケーションコマンドを用いる．2 つ目は，ミドルウェア層である．コマンド・レスポンスユニットは，アプリケーションコマンドに従って，データのエンコード・デコード，インタロゲータである RFID リーダライタの操作を行う．ISO/IEC 18000-6TypeC は，UHF 帯 RF タグを利用するためのエアインタフェースを定義しており，定義されたエアインタフェースである Read コマンド，Write コマンドによりユーザメモリのデータの読み書きを行う．

RFID ミドルウェアは，コマンド・レスポンスユニットが RFID リーダライタドライバを利用して，インタロゲータである RFID リーダライタを操作してデータの読み書きを行う．しかし，コマンド・レスポンスユニット，RFID リーダライタドライバに明確な設計は存在しない．そこで次節では，それぞれが設計すべき項目について述べる．

### 2.3.1　コマンド・レスポンスユニット

コマンド・レスポンスユニットは，アプリケーションコマンドに応じてユーザメモリへのデータの追加・削除・編集を行い，その結果をアプリケーションに通知する．コマンド・レスポンスユ

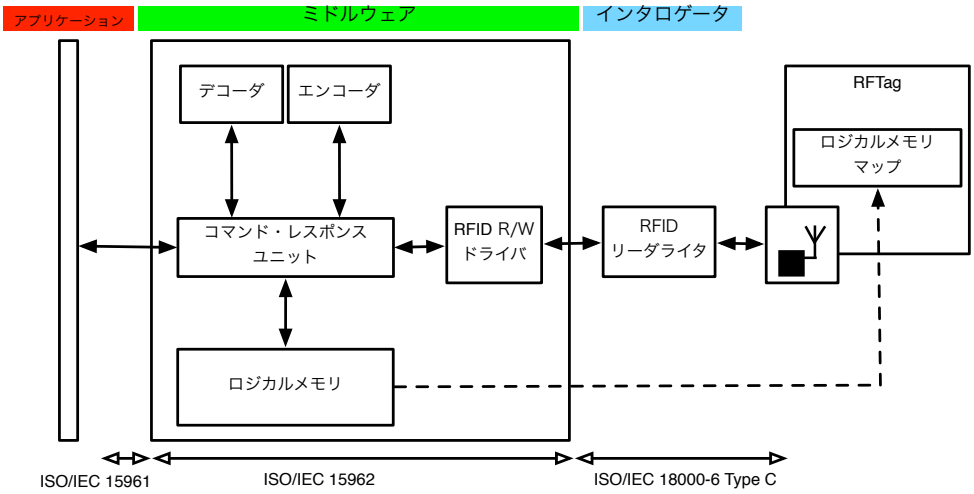


図 2.4　ISO/IEC 15961, 15962, 18000-6TypeC を利用した際のレイヤ構成

ニットの明確な設計は存在しないが，データ追加時には，次の 4 つの手順を実行することが考えられる．

1. DSFID の取得
2. 既存データの読み込みと解析
3. データのフォーマッティング
4. データの書き込み

既存データの読み込みと解析には，明確な仕様は存在しないが代表的な設計として次の 2 つが考えられる．1 つ目は逐次解析・読み取り型で，Object-Length の情報に従って，読み込みが必要な分だけ読み込みを行う．この手法は，最小限のデータのみの読み込みを行うことができ，無駄なデータを読み込む必要がない．2 つ目は一括読み取り・解析型で，一度すべてのデータの読み込みを行ってから解析を行う．この手法は，書き込みを行っていないデータまで読み込みを行ってしまうが，読み込みシーケンスで利用する Read コマンド数を少なくすることができる．

### 2.3.2　RFID リーダライタドライバ

RFID リーダライタドライバは，コマンド・レスポンスユニットとエアインタフェースを考慮したインタフェースが必要となる．

#### インタフェース

コマンド・レスポンスユニットを考慮した設計として，データフォーマットはバイト単位で構成されるため，提供するインタフェースの引数はすべてバイト単位で行う必要がある．EPCglobal は LLRP(Low Level ReaderProtocol)[16] に定義している．LLRP Version 1.1 は，書き込みイ

ンタフェースに C1G2Write を用意しており，スペック ID，メモリバンク，書き込み開始アドレス，書き込みデータ，アクセスパスワードの 4 つのパラメータを用いる．しかし，ISO/IEC 15962 のデータフォーマットの仕様と複数種類の RF タグへの対応を考慮した場合，以下に示すような RFID リーダライタドライバインタフェースが必要になる．

- writeDataToUserMemory: ユーザが利用可能なメモリ空間に書き込みを行う．
- readDataFromUserMemory: ユーザが利用可能なメモリ空間から読み込みを行う．
- setDSFID: DSFID を保存するアドレスへ書き込みを行う．
- getDSFID: DSFID が保存されているアドレスから DSFID を取得する．
- setAFI: AFI(Application Family Identifier) を保存するアドレスへの書き込みを行う．
- getAFI: AFI が保存されているアドレスから AFI を取得する．

**設計**

RFID リーダライタドライバは，利用する RF タグの種類 (例えば，UHF 帯 RF タグと HF 帯 RF タグ）により，DSFID を保存するメモリ空間が異なるため，共通のインタフェースで異なるアドレスへの書き込みも可能な設計が必要である．また，それぞれのインタフェースの利用タイミングは，コマンド・レスポンスユニットの設計に依存するため，完全に独立した設計にすることが一般的である．図 2.5 に，データ追加時のコマンド・レスポンスユニットの動作とエアインタフェースの呼び出し手順を示す．まず，DSFID の読み込みを行い，どのアクセスメソッドが利用されているのかを取得する (1)．DSFID 取得には，Read コマンドを用いるため，Select コマンド，Inventory コマンドを実行する．そして，書き込み位置を推測を行うために，既存データの読み込みとデータフォーマットの解析を行う (2)(3)．最後に，読み込みと書き込みの間のデータフォーマットの一貫性を保つために，既存データと追加データの書き込みを行う (4)．図 2.5 に示すように，読み書きを行うインタフェースが独立して動作するように，(1)，(2)，(4) のそれぞれで Select コマンド，Inventory コマンドが実行される．本研究では，この設計を既存 RFID リーダライタドライバとする．

### 2.3.3 データフォーマットの解析

図 2.6 に，Non-Directory アクセスメソッドのデータの解析手順を示す．はじめに DSFID を取得し，利用しているアクセスメソッドを取得する．そして Precursor を読み取り，次に Object-Length を読み取る．Object-Length で取得したデータ長分 Object の読み込みを行う．この操作の繰り返しを行い，Precursor 部分が END になった場合，解析処理を終了する．

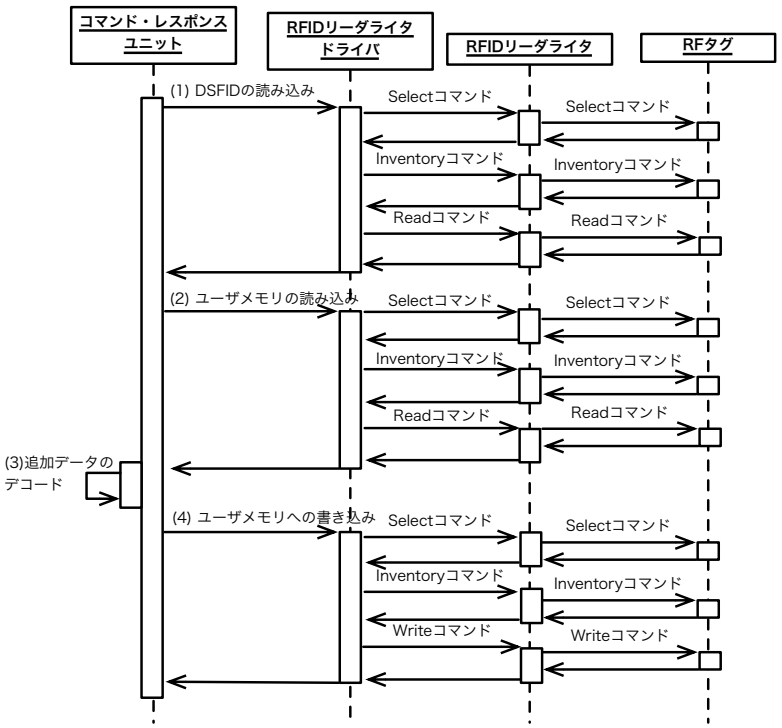


図 2.5　RF タグへのデータ追加シーケンス

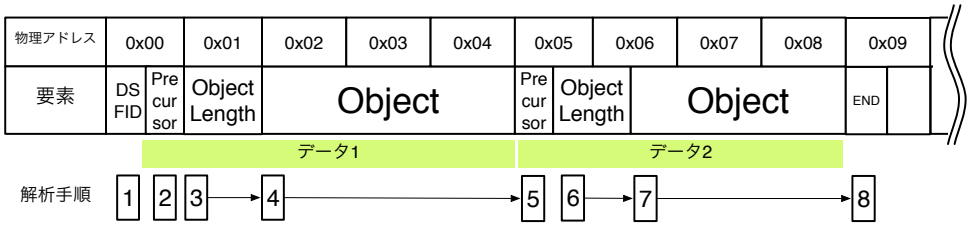


図 2.6　Non-Directory のデータ解析手順

### 2.3.4　データ追加の実験

コマンド・レスポンスユニットと既存 RFID リーダライタドライバのデータ追加における関係性を明らかにするために，データ追加の実験を行った．コマンド・レスポンスユニットは，逐次解析・読み込み型と一括読み取り・解析型の 2 つを用意し，RFID リーダライタは第 2.3.2 節で示した既存 RFID リーダライタドライバを用いた．一括読み取り・解析型では，データ長は既知であるものとし実験を行った．表 2.1 に，実験で用いた機材を示す．

実験で用いたデータは，サプライチェーンマネジメントを実現するためのアプリケーションで利用されている位置情報とタイムスタンプである [1][3]．位置情報として，グローバルロケーション

ナンバ (以下，GLN) と呼ばれる 14 桁の数字と，2011 年 01 月 12 日 12:15 を 1101121215 の 10 桁で表すタイムスタンプを用意し，2 つのデータを一組にして，一組ずつ追加した．

図 2.7 にデータ追加と実行時間の関係を示す．x 軸は追加した組数，y 軸は追加に要した実行時間を示している．実行時間では，エアインタフェースを出力し結果が得られるまでの時間を合計したものである．RFID リーダライタドライバは，読み書き間のデータ一貫性を保証できない．そのため，既存データの上書きする必要があり，実行時間は，書き込みデータ組数に比例している．逐次解析・読み込み型は，読み込みのたびに Select コマンド，Inventory コマンドがそれぞれ出力されるため 8 組目の追加で約 30 秒を超える時間が必要なっている．一括読み込み・解析型は，読み込みに必要な時間を一定に抑えることができるため，書き込み時間の増加のみで処理を終了することができているため 8 組目の追加で約 5 秒で処理を完了できる．このように，コマンド・レスポンスユニットの設計により実行時間に大きな差が生じるということがわかった．

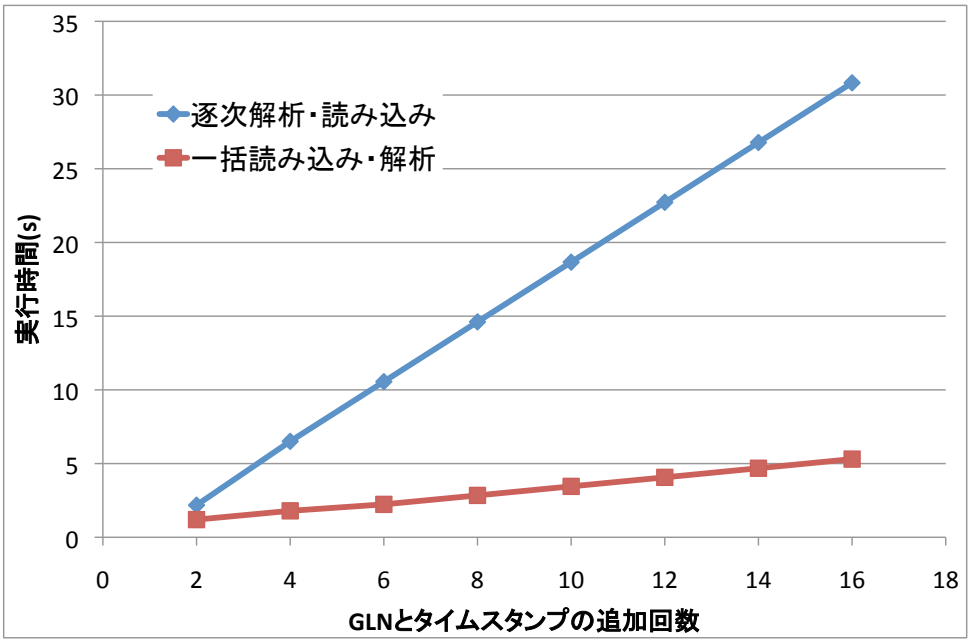


図 2.7　RF タグへの追加データと実行時間の関係

## 2.4　トランザクション機能の必要性

前節までに，データの追加に関する技術を述べた．本節ではまず，データ追加時におけるシナリオをあげ，発生する課題についてまとめる．そして，トランザクションの必要性について述べる．

### 2.4.1 対象とする環境

本論文は，GS1 データバーのように全てのオブジェクトに対して RF タグが添付され，さまざまな情報を保持することが可能となり，またユーザはそれらに情報を書き込むための RFID リーダライタが内蔵されたモバイル端末を保持していることを前提とする．そしてモバイル端末は，端末ごとに異なるベンダーによって作成されているとし，コマンド・レスポンスユニットの性能は異なっているものとする．

### 2.4.2 シナリオ

本論文では，特定のオブジェクトに，利用履歴 (返却時間) を書き込むアプリケーションを想定する．図 2.8 に，シナリオの図を示す．A 君が自身のモバイル端末 A を用いて，RF タグの付いたオブジェクト X に利用履歴を追加していた．しかし追加の際に，**A 君はデータ追加完了を待てず**，モバイル端末をオブジェクト X から離した．その後、B 君が同じオブジェクト X の利用履歴を見るために，オブジェクト X の情報を，自身の持つモバイル端末 B を用いて取得しようとした．モバイル端末 B は，A 君が**書き込んでいた途中のデータ**を読み込むことができた．しかし，読み取れたデータは，A 君が意図していたデータではなかったが，B 君はそれを知ることができず，自身のもつアプリケーションで利用してしまった．

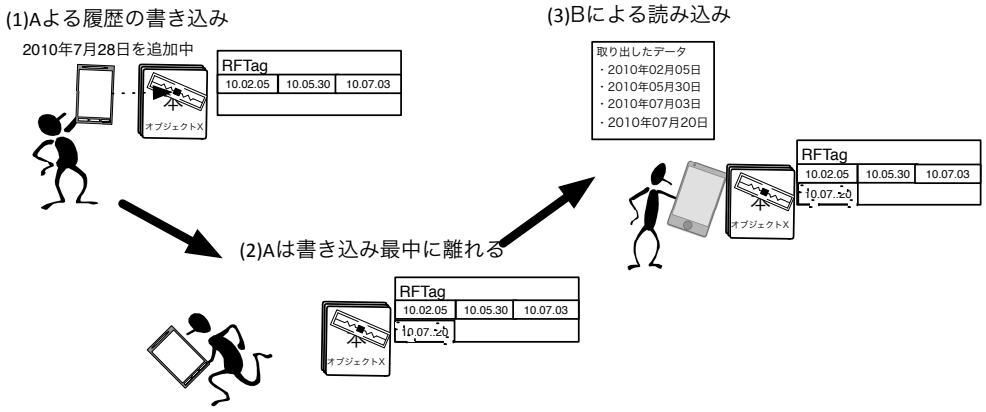


図 2.8　電波伝搬エラーにより一部のデータが RF タグの内部に残存する問題

### 2.4.3 トランザクション機能の要件

シナリオで示したとおり，不完全データが発生した場合，他の RFID システムでそれらが読み取られて利用されてしまう可能性がある．そこで，本論文では，データ追加時に不完全データが発生してしまった場合，それを無効化するトランザクション機能について検討する．

#### 不完全データの無効化と互換性

ISO/IEC 15962 の Non-Directory アクセスメソッドのデータフォーマットには，データの正当性を示す構造は持たない．そのため，読み込まれたデータは常に正当性の有無なしに利用される．つまり，RF タグに書き込まれているデータ自体の信頼性は低い．

物につけられた RF タグは，背景やシナリオで示したように，人から人にわたり，それぞれの RFID システムで利用されることが想定される．そのため不完全なデータは，他の RFID システムに対して無効化しておき，RF タグ内のデータは常に完成したデータのみにする必要がある．

#### データ追加の遅延

データ追加に要する実行時間は，図 2.7 で示したように，既存データと追加データの数に比例して，実行時間は上昇する．そのため，追加データは少ないのにも関わらず，実行時間を要するということが考えられる．また，RF タグへのアクセス中は，不意に動かすと磁界が不安定になるため，より短時間での書き込みが必要となる．

## 2.5 関連研究

本節では，ユーザメモリからデータの読み書きに関連する関連研究を紹介する．

### 2.5.1 Session Manager

慶應義塾大学の苧阪らは，RF タグ内の各データのスタートアドレスと長さをネットワークに繋がったデータベース上で管理し，それを元にデータベースに格納する手法である Session Manager を提案している [17].

図 2.9 に Session Manager の基本アーキテクチャと，図 2.10 に Session Manager を利用したデータ追加シーケンスを示す．まず Session Manager は，データマッピングルールを知るため DSFID の読み込みを行う (1)．次に Session Manager は，Session Resolver を通して，タグ内のデータ構造が登録されている Schema Registry にアクセスを行い，書き込みを行う場所を把握しデータの追加を行う (2)．そして，Session Manager は，データの追加処理を行う (4)．最後に，Session Manager はデータを追加したことを Schema Registry に登録を行うために Schema Resolver を通してアクセスを行う (5).

この手法により，完全なデータのみをネットワーク上のデータベースに保持することにより，不完全データを無効化でき，またデータフォーマットの一貫性もデータベースのトランザクション処理を利用することにより機能要件を満たすことが可能である．しかし Session Manager は，ネットワークを前提にした手法になっているため，本論文が対象としている環境には十分ではない．

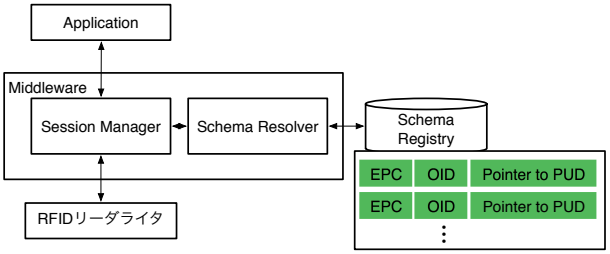


図 2.9　Session Manager の基本アーキテクチャ

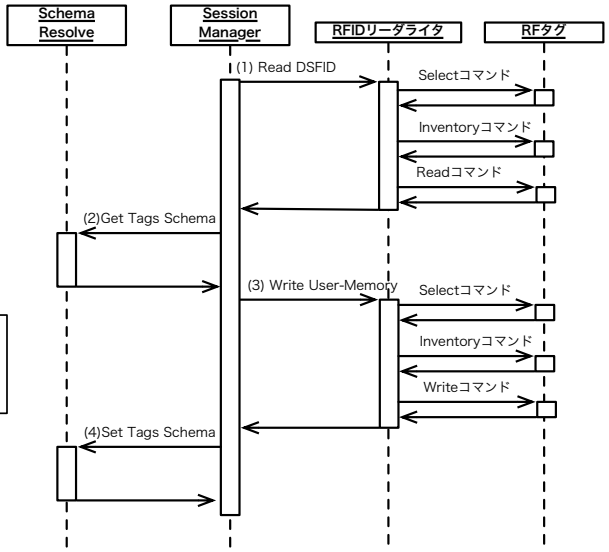


図 2.10　Session Manager を用いたデータ追加時のシーケンス

### 2.5.2　Validation Information

Pusan National University の Seok-Young らは，特定の RFID システムがデータ追加中であることを他の RFID システムに通知する Validation Information を提案している [18]．Validation Information は，データ追加後にユーザメモリのチェックデジットを別のメモリ空間に書き込みを行う．ミドルウェアが，データ追加処理を開始する際には，チェックデジットとユーザメモリの一致を確認をする必要がある．図 2.11 に Validation Information を利用したデータ追加シーケンスを示す．まず，チェックデジットの読み込みとユーザメモリの読み込みを行う (1)(2)．次に，チェックデジットとユーザメモリが一致することの確認を行う (3)．そして，チェックデジットとユーザメモリが一致すると書き込み処理を開始する (4)．最後にユーザメモリのチェックデジットの追加を行う (5)．もし，書き込み途中で処理が落ちた場合，チェックデジットとユーザメモリが一致しなくなるため，別の RFID システムの書き込みは発生しない．そのため，データフォーマットの一貫性を保つことができ，それにより不完全データの無効化が可能となり，機能要件を満たす．しかし，本論文が対象とする複数 RFID システム間での情報共有の点において十分ではない．

### 2.5.3　Virtual Tag Memory Service

ETH Zurich の Christian らは，RF タグに追加するデータをネットワーク上のデータベースと RF タグの両方に格納する Virtual Tag Memory Service を提案している [19]．RF タグには，メモリスペースやメモリ構成がメーカや規格に依存する．Virtual Tag Memory Service は，RF タ

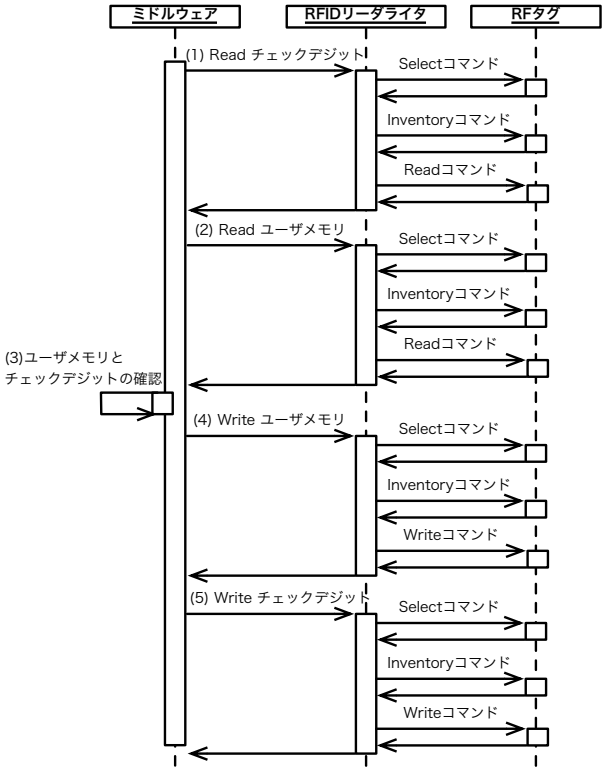


図 2.11　Vadliation Information を用いたデータ追加シーケンス

グに十分なメモリスペースが確保できなかったり，書き込みに失敗したりした場合，ネットワーク上のデータベースのみにデータを追加する．これにより，RF タグ依存の部分がなくすことが可能となる．この手法では，複数の場所にデータを残すため，本論文が対象としている複数 RFID システム間での情報共有には，利用が十分ではない．

### 2.5.4　Reprocessing Model

Pusan National University の Wooseok らは，データ追加処理が途中で終了してしまった場合，再度書き込み処理が開始できる Reprocessing Model の提案を行なっている [20]．表 2.2 に，Reprocessing Model で最書き込みに利用するデータの例を示す．再度書き込み処理を行うために，書き込みを行うデータの書き込み開始アドレス，データ長，データ，何 word 書きこまれたのか，書き込みのオペレーション ID を記録する．この提案手法を用いた RFID システムは，RF タグを検知後，まず書き込みが完了していないデータが存在しないかのチェックを行い，処理が完了していない場合は，その続きから書き込みを再開する．これにより，データフォーマットの一貫性を保つことが可能となるが，本論文が対象としている複数 RFID システム間での情報共有には，利用が十分ではない．

表 2.2　書き込み途中の処理を記録するテーブル

| ID | TID | アドレス | データ長 | データ | 書き込んだ Word 数 | オペレーション ID |
|---|---|---|---|---|---|---|
| 1 | 99887766 | @3.15 | 1 | L2 | 0 | 32 |
| 2 | 99887766 | @3.2 | 2 | 1234 | 1 | 627 |
| 3 | 99887766 | @3.0 | 2 | A902 | 0 | 184 |

### 2.5.5　Anti-tear Transaction

Suica システムで利用されている FeliCa では，データの追加時に Anti-tear Transaction というトランザクション機能がハードウェア実装として FeliCa 自身に埋め込まれている [21]．8 ブロック (1 ブロック=16 バイト) 単位で，トランザクションを有効にした書き込みがはじめから用意されている．しかし，物に添付されることが考えられている UHF 帯 RF タグは，ハードウェア上でのトランザクションが用意されていない．そのため，データフォーマット上でのトランザクション処理が必要となる．

## 2.6　既存研究のまとめ

Session Manager，Validation　Information，VTMS，Reprocessing Model についてそれぞれ述べた．Session Manager，VTMS，Reprocessing Model の共通点として，EPCglobal のように，ネットワークを前提となる提案手法となる．そのため，本論文が目的としている RFID システム間での情報共有の利用に十分ではない．また，Session Manager，Validation Information，Reprocessing Model は，同手法内のみでは，不完全データを無効にすることが可能であるが，他の RFID システムでは読み込まれて利用されてしまう危険性がある．

図 2.12 に既存手法として挙げた第 2.5 節を実装に落とし込んだ際の関係図を示す．各従来手法は，提供される RFID リーダライタドライバや LLRP を利用し，コマンド・レスポンスユニット内に手法依存の読み取り，書き込みを必要としていた．そのため，他の手法との互換性を持つことができなかった．そこで，複数 RFID システム間での情報共有を行うために，データの解析を行うコマンド・レスポンスユニットは最小の機能形態にすることが重要であると考えられる．

## 2.7　本章のまとめ

本章ではまず，RF タグへのデータ追加するためのエアインタフェースとデータフォーマットの詳細について述べた．次に，シナリオを元にデータ追加時のトランザクションの機能要件について述べた．そして，データ追加における関連研究について述べ，比較を行った．次章では，補償型トランザクション機構の提案を行う．

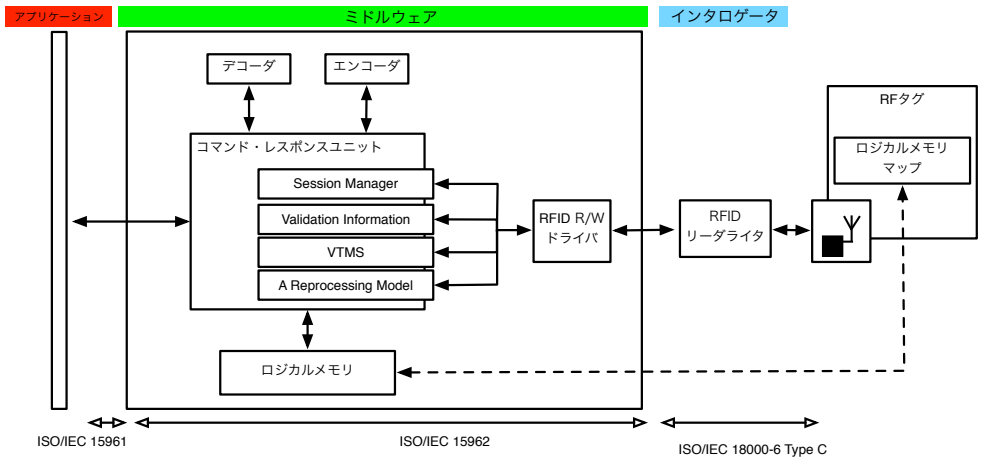


図 2.12　関連研究の実装位置関係

# 第 3 章

# 補償型トランザクション機構の提案

本章では，不完全データの無効化，データフォーマットの一貫性を保証するための補償型トランザクション機構の提案をする．そして，補償型トランザクション機構の各要素について詳細に述べる．

## 3.1 概要

本論文では，不完全データの無効化と，データフォーマットの一貫性を実現するために補償型トランザクション機構を提案する．補償型トランザクション機構は，ユーザメモリへのデータ追加処理中に RFID リーダライタと RF タグ間が通信を行えなくなり，ユーザメモリに不完全なデータが残存した場合，補償型トランザクション機能により，不完全データの無効化を行う．データ追加時には，読み込みシーケンスと書き込みシーケンス間でデータフォーマットの一貫性を RFID リーダライタドライバのレイヤで保証することにより，新規データ量のみに比例する書き込み時間を実現する．

補償型トランザクション機構は，データパディング，Handle Manager，バックライトで構成する．図 3.1 に，全体の流れを示す．フォーマッティングを行った追加データは，まずデータパディングにより，ユーザメモリの物理アドレスに最適化を行う．そして，Handle Manager により，読み込みシーケンスと書き込みシーケンス間でのデータフォーマットの一貫性を保証する．最後に，書き込みシーケンスにおいて，データ解析を考慮した書き込み手法であるバックライト手法を用いる．

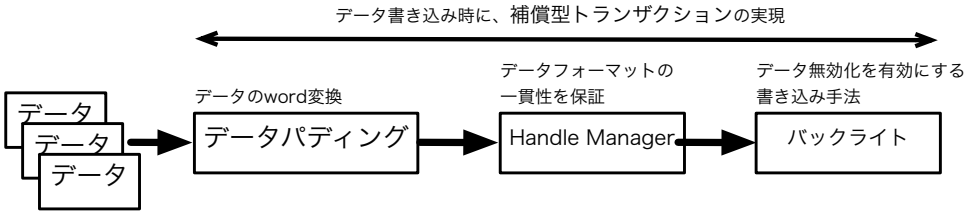


図 3.1　補償型トランザクション機構の仕組み

## 3.2 データパディング

データパディングは，1 データ自身の長さ調節を行い，物理アドレスに一致するように 1 データの長さを変更する．UHF 帯 RF タグのユーザメモリでは，図 3.2 に示すように，1word の中に 2 つのデータが存在することが考えられる．そのため，Handle Manager によるより，データフォーマットの一貫性が保証されたにも関わらず，既存データの上書き処理が入る．そこで，図 3.2 に示すデータ構造を発生させないため，データフォーマットを word 単位で書き込みができるように，データフォーマットの変換を行う．これにより，1word 内に 2 つのデータを存在しなくなる．

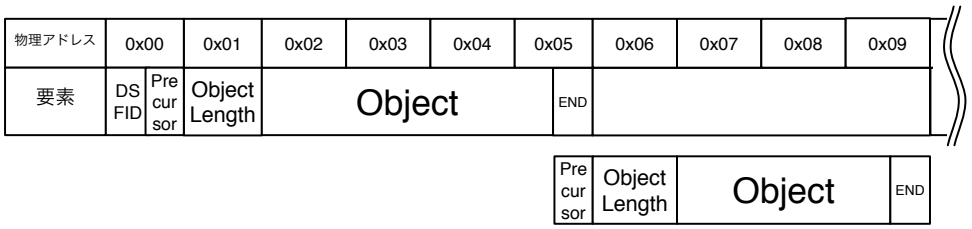


図 3.2　Handle Manager のシーケンス

## 3.3　Handle Manager

Handle Manager は，読み込みシーケンスと書き込みシーケンス間でのデータフォーマットの一貫性を保つ．データフォーマットの一貫性が保たれていなかった原因として，RFID リーダライタドライバインタフェース間の独立した設計が挙げられる．そこで本論文では，インタフェースの独立性を保持しながら，RFID リーダライタドライバ内部で協調した動作を実現するため，Read コマンド，Write コマンドの引数である handle 値の共通化を行う．handle 値は，RFID リーダライタの性能により，1 秒または 4 秒の有効時間と，他の RFID リーダライタがアクセスを行うと無効となる．

Handle Manager は，handle 値の生成時刻を Handle Manager に保持させ，利用時に利用時刻との比較を行う．もしも生成時刻と利用時刻の差が有効時間以内であれば，データフォーマットの一貫性が保たれていることになり，冗長なデータの上書きを行わない．もしも handle 値が無効の場合は，前データの読み込みを行い，データフォーマットに変化の有無をチェックし，変更がなければ継続して書き込みを行う．

## 3.4　バックライト

不完全データの無効化と他システム間との互換性を保つために，データ解析のシーケンスとデータ構造に着目したバックライト手法を提案する．図 3.3 に，不完全データの無効化のため，データ書き込みのシーケンスを示す．1 データの構成要素である，Precursor，Object-Length，Object，END の並び順に対して, 後ろのアドレスから書き込みを行う．これにより, END, Object-Length, Object のどの場所で書き込みができない状況が発生しても，既存データの END が後方のデータを無効と判断させることが可能となる．

バックライトは，データ追加に対し，補償型トランザクションの機能を保持している．もしもデータ追加時に，不完全データが発生した場合，それらの処理は書き込みを行った部分は無効なデータとして扱われることとなるため，既存データを維持することができる．

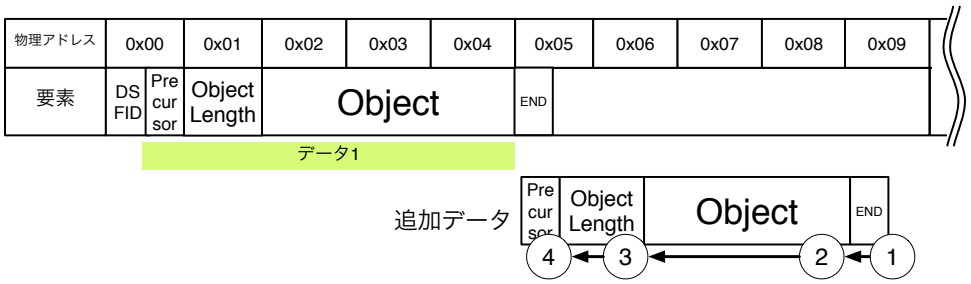


図 3.3　データ追加シーケンス

### 3.4.1　他 RFID システムとの互換性

データ解析は，コマンド・レスポンスユニットによって行われる．そのため，既存の関連手法が行っているようにコマンド・レスポンスユニットに機能追加することにより読み書きデータの互換性を保つことは可能である．しかし，複数 RFID システム間でのデータ交換を考慮した場合，データ解析は，最低限の設計にする必要があり，その条件として完全なデータのみを RF タグ内に用意しておくことである．図 3.4 に，エラーが発生時の不完全データの様子と，不完全データを含む RF タグに新規データを追加の様子を示す．追加データ 1 の Object 追加完了後にエラーにより，ユーザメモリ内に不完全データが生成されている (1)．別の RFID システムは，データ 1 の後ろの END により，データ 1 までしか認識されない．そのため追加データ 1 の不完全データを上書きする形で追加データ 2 を追加することが可能となる (2)．よって，他 RFID システムとの互換性を保つことができる．

## 3.5　本章のまとめ

本章ではまず，データ追加時の課題を達成するための補償型トランザクション機構の提案手法について述べた．そして，補償トランザクション機構として，バックライト手法，Handle Manager 手法について述べた．次章では，補償型トランザクション機構の設計について述べる．

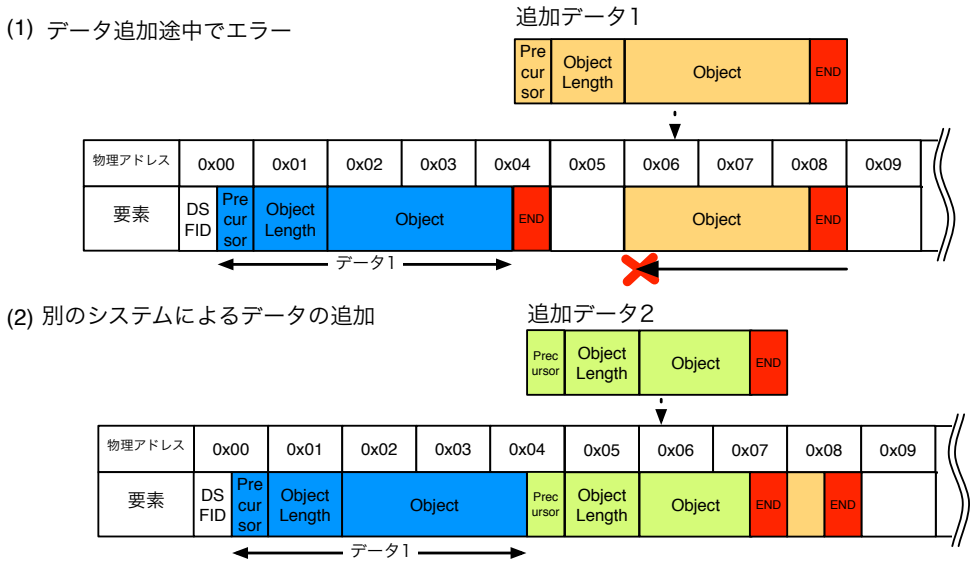


図 3.4　バックライト手法を用いたときの互換性

# 第 4 章

# 設計

本章では，補償型トランザクション機構である，データパディング，Handle Manager，バックライトを実現するための設計について述べる．また，RFID ミドルウェアに補償型トランザクション機構を搭載するために，RFID ミドルウェアの設計についても詳しく述べる．

## 4.1 概要

図 4.1 に，本提案手法である補償型トランザクション機構を実現するための設計概要を示す．提案手法を実現するために，コマンド・レスポンスユニット内部にデータパディング，RFID リーダライタドライバと同じレイヤに，Handle Manager を追加する．また Handle Manager 内部には，データフォーマットの一貫性を保証するために RF タグ内のデータを監視する Virtual Tag を保持させる．そしてバックライトは，RFID リーダライタドライバ内部に実装を行う．

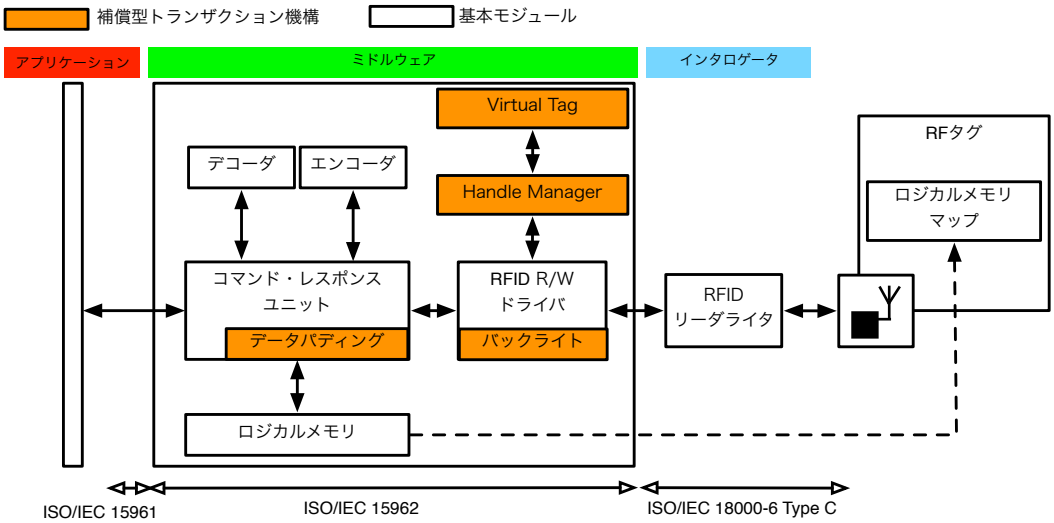


図 4.1　設計概要

## 4.2 データパディング

データを word 単位に変換する方法として，データフォーマットのオフセットを利用したパディングと，フォーマットのスキップデータである 0x80 の 2 つの方法が考えられる．しかし，コマンド・レスポンスユニットの実装によっては，0x80 を上書きしたデータ追加を行う可能性があり，既存データの上書きを行う可能性がある．そこで，オフセットを用いたデータパディングを行う．

パディングは，コマンド・レスポンスユニット内で行う．書き込みデータのエンコードが完了し，書き込みを行う直前に，UHF 帯 RF タグの word 単位へデータフォーマットを変換する．図 4.2 にフローチャートを示す．まず，書き込みを行うデータセット内から，データの配置を仮想的に行う．そして，仮想配置の結果，word 単位になっていない場合は，オフセットを有効にする．最後に，仮想配置をもう一度行い，次のデータセットのチェックに移る．

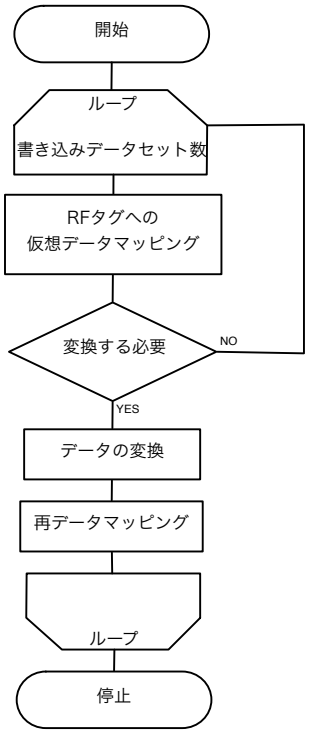


図 4.2　パディングの変更フローチャート

## 4.3　RFID リーダライタドライバ設計

RFID リーダライタドライバは，コマンド・レスポンスユニットによって利用され，RF タグへの読み書きを行う．図 4.3 に，RFID リーダライタドライバ，Handle Manager，Virutal Tag のクラス図を示す．Handle Manager は，1 つの RFID リーダライタドライバに対して 1 つ存在し，Handle Manager 内で複数の RF タグに対応づけを行うために，固有番号ごとに RF タグの管理を行う．そのため，Virtual Tag は Handle Manager に複数紐づくことになる．

### 4.3.1　インタフェース

コマンド・レスポンスユニットは，RFID リーダライタドライバが提供するインタフェースを利用して，データの解析，追加を行う．また，RF タグの種類により，DSFID などの RF タグに格納する個有データの位置は異なる．そこで本論文では，以下に示すインタフェースを提供することにした．

- getDSFID:指定された RF タグから，DSFID を取得する．
- setDSFID:指定された RF タグに，指定された DSFID を書き込む．

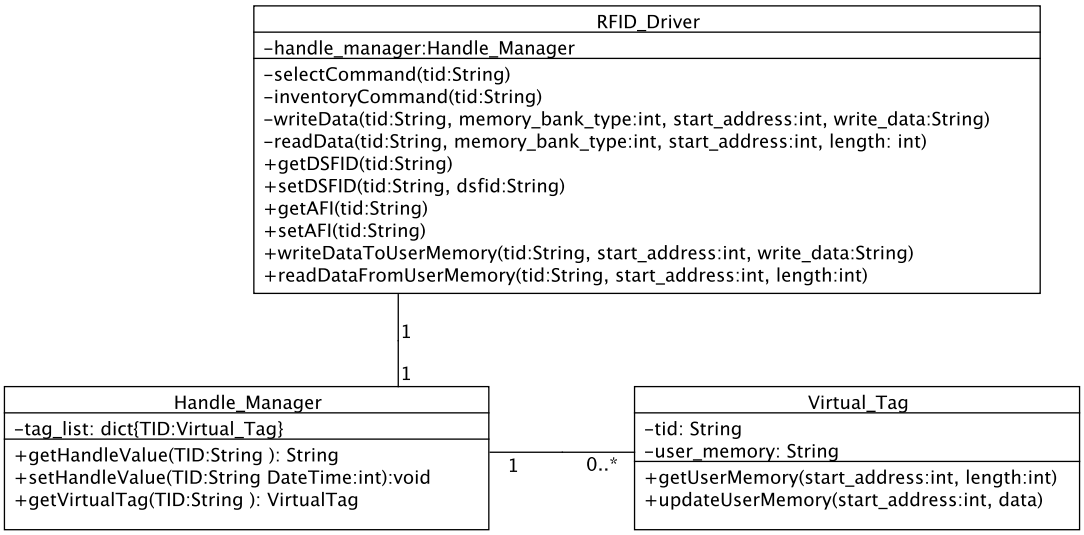


図 4.3　UML:クラス図

- getAFI:指定された RF タグから，AFI を取得する．
- setAFI:指定された RF タグに，指定された AFI を RF タグに書き込む．
- writeDataToUserMemory:指定された RF タグに，指定されたアドレスから，指定されたデータを書き込む．
- readDataFromUserMemory:指定された RF タグから，指定されたアドレスから，指定された長さ分のデータを取得する．
- writeData:setDSFID や writeDataToUserMemory などの書き込み処理から呼び出される．Write コマンド実行時に、渡されたデータをバックライト手法と Handle Manager により管理された handle 値と Virtual Tag を用いて書き込みを行う．
- readData:getDSFID や readDataFromUserMemory などの読み込み処理から呼び出される．Read コマンド実行時には，Handle Manager により管理された handle 値と読み込んだデータを Virtual Tag の中に保存する．

RFID リーダライタドライバは，RF タグと RFID リーダライタ間の handle 値を生成しなくてはならない．そこで，Select コマンドから，Inventory コマンドを実行する createHandle メソッドを用意する．createHandle メソッドは次節に述べる Handle Manager から呼び出されるようにする．

### 4.3.2　Handle Manager

Handle Manager は，Select コマンド，Inventory コマンドで取得できる handle 値の管理を行う．Handle Manager は，RFID リーダライタドライバからの handle 値要求時に，生成された時間と，その経過時間のチェックを行い handle 値を戻す．もしも，handle 値が無効の場合，Handle

Manager から RFID リーダライタ内に用意されているメソッドを用いて handle 値を生成し，再度管理を行う．Handle Manager が用意するインタフェースを以下に示す．

- getHandleValue:指定した RF タグの handle 値を取得する．
- getVirtualTag:指定した RF タグの VirtualTag のインスタンスを取得する．

図 4.4 に，handle 値取得時の Handle Manager の動作シーケンスを示す．まず，RFID リーダライタドライバからアクセスコマンド利用のために，Handle Manager の getHandleValue メソッドを呼び出す (1)．そして，Handle Manager は，handle 値が利用可能かを検証する．もしも無効ならば，RFID リーダライタドライバの createHandle メソッドを呼び出し handle 値と handle 値の生成時刻，handle 値の有効時間を取得する (2)．Handle Manager は戻ってきた値を内部で再び管理し RFID リーダライタドライバに handle 値を戻す．

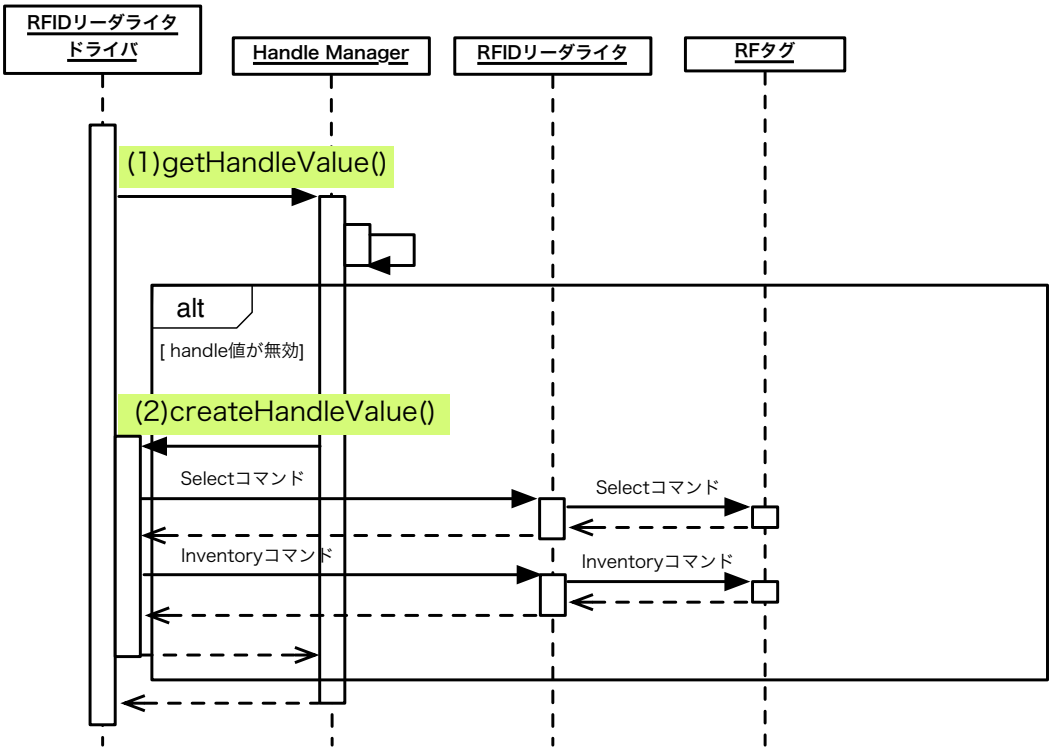


図 4.4　handle 値取得時の Handle Manager のシーケンス

### 4.3.3　Virtual Tag

Virtual Tag は，物理的に読み書きを行なっている RF タグのユーザメモリを仮想的に再現する．Virutal Tag と実際の RF タグを同期することにより，handle 値の更新時に書き込みを即座に開始することが可能となる．例えば，追加を行うデータ量によっては，同一 handle 値を利用して書

き込み処理が完了しない場合がある．その際には，Select コマンド，Inventory コマンドを行い，handle 値の更新を行う．そして Virtual Tag と現状の RF タグ内のデータが一致するかのチェックを行い，もし一致したならば，途中からの書き込みを再開する．Virtual Tag は以下のインタフェースを提供し，RF タグのデータを保持する．

- getUserMemory:指定したアドレスと長さから，指定された RF タグのユーザメモリからを取得する．
- updateUserMemory:指定したアドレスとデータを仮想 RF タグに追加する．

### 4.3.4 Handle Manager と Virtual Tag のシーケンス

図 4.5 に，Handle Manager，Virtual Tag を用いた書き込みのシーケンスを示す．まず，RFID リーダライタドライバは，入力された値を word 単位に分割を行う．そして，書き込みを行うデータを Handle Messagener を通して，Virtual Tag に既に書かれているのかをチェックする (1)．ここで True が戻ってきたならば，次に書き込む値のチェックを行う．もしも False がもどってきた場合は，Handle Manager から handle 値を取得し，書き込みを行う (2)(3)．そして，書き込みが完了した場合は，Virtual Tag 内のデータを更新する (4)．これにより，Virtual Tag は RF タグ内のデータと同期することができ，handle 値有効時のデータ取得時には，Virtual Tag からデータの取得を行う．

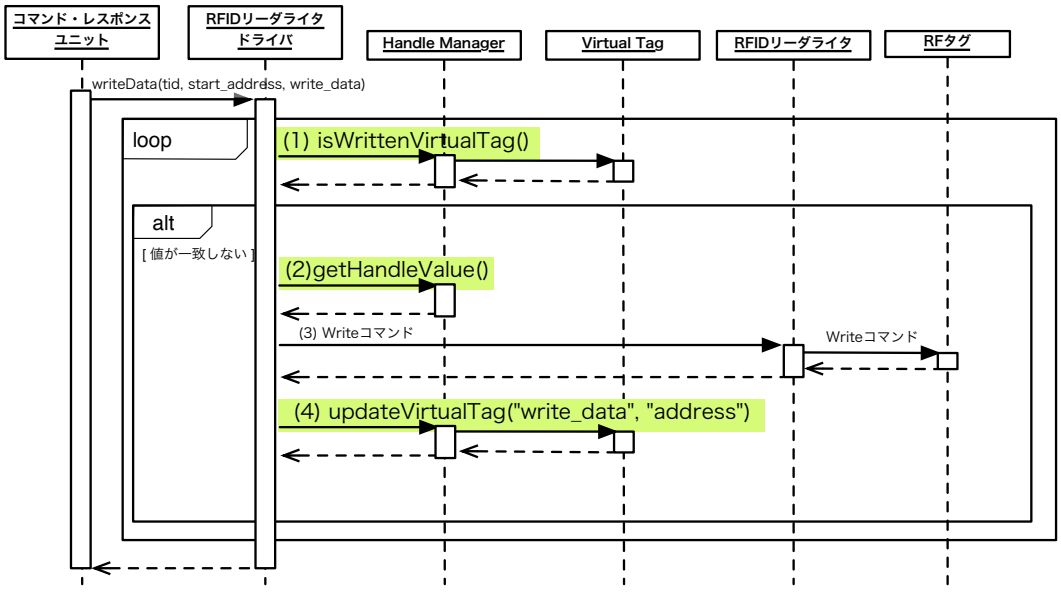


図 4.5　データ追加シーケンス

## 4.4 本章のまとめ

本章では，補償トランザクション機構を実現するためのデータパディング，Handle Manager，バックライトの設計について詳細に述べた．次章では，補償トランザクション機構の実装について述べる．

# 第 5 章

# 実装

本章では，補償型トランザクション機構の実装と RFID ミドルウェアの実装ついて詳しく述べる．RFID ミドルウェアの実装は，データ追加に必要となるコマンド・レスポンスユニットを中心に述べる．そして最後に，補償型トランザクション機構のデータフォーマット一貫性の保証が正しく行えているかを検証する．

## 5.1 実装環境

本論文が実装に利用した実装環境について述べる．表 2.1 に，実装環境で利用した機材について示す．図 5.1 に，実装環境を示す．RFID ミドルウェアを PC 上に実装を行った．そして，PC と RFID リーダライタモジュールを USB シリアルで接続し，RFID リーダライタモジュールとアンテナを接続する．

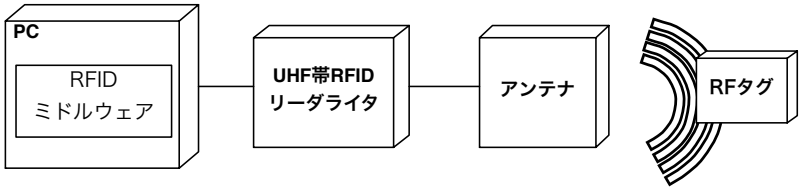


図 5.1　実装環境の構成

## 5.2 コマンド・レスポンスユニット

データ追加時において，コマンド・レスポンスユニットは，以下の項目を実行する．

1. アクセスメソッド解析モジュール
2. データ解析モジュール
3. データ構築モジュール

アクセスメソッド解析モジュールは，DSFID を取得し，ユーザメモリで利用されているアクセスメソッドを解析する．データ解析モジュールは，設計依存となり，様々な方法が考えられる．本論文では，逐次読み込み・解析型，一括読み取り解析型を用意する．そして，どちらも解析の際には，最小限の要件として，外部ネットワークなどを用いないデータの解析を行う．読み込みの際には，RFID リーダライタドライバの readDataFromUserMemory インタフェースを用いる．データ構築モジュールは，アクセスメソッドに従いデータをフォーマティングする．フォーマッティング後に，提案手法であるデータパディングを行う．そして，書き込みを行う際には，RFID リーダライタドライバの writeDataToUserMemory インタフェースを用いる．

### 5.2.1 アクセスメソッド解析モジュール

アクセスメソッド解析モジュールは，RFID リーダライタドライバの getDSFID インタフェースから DSFID を取得する．取得した DSFID から，適切なデータ解析モジュール（Non-Directory，Directory，Tag-Data-Profile，Packed-Object アクセスメソッド）を取得する．また，DSFID が

書き込まれていない場合は，本モジュールが呼ばれたタイミングで書き込みを行う．本論文では，Non-Directory アクセスメソッドを対象とするため，次節に Non-Directory アクセスメソッドのデータ解析について述べる．

### 5.2.2 データ解析モジュール

データ解析モジュールでは，RF タグ内のデータを読み込み，解析を行う．データの読み込み手法には，第 2 章で示したように，逐次解析・読み込み型と一括読み取り・解析型の 2 つが考えられる．データ解析は，Precursor，Object-Length，Object の順に先頭アドレスから解析を行う．そして Precursor に END になっていたときに解析を終了する．

### 5.2.3 データ構築モジュール

解析が行われたデータは，アプリケーションから渡された新規追加データを加えてデータフォーマットの再構築を行う．そして構築後，データを RFID リーダライタドライバが提供する writeDataToUserMemory に渡す．その際，データの渡し方は次の 2 つが考えられる．1 つ目は，全データを一度にすべて渡す方法である．Packed-Objects アクセスメソッドのように，1 データごとに分割できないようなデータフォーマットに対して有効である．2 つ目は，1 データずつ渡す方法である．Non-Directory アクセスメソッドのように，1 データずつ分割が可能なデータフォーマットでは，1 データごとの書き込みが可能となる．本論文では，ISO/IEC 15961 のアプリケーションインタフェースにどのデータを書き込み，どのデータが書きこまれていないかをレスポンスで返す必要があるため，後者の 1 データずつ RFID リーダライタドライバに渡す実装にした．

#### データパディング

データパディングは，1 データ単位を word 単位にする変換を行う．設計で示した通り，コマンド・レスポンスユニットに実装を行う．図 5.2 は，GLN を word 単位に変換したときにデータフォーマットである．Precursor の bit8 を 1 にすることにより，オフセット機能が有効となり，オフセットに指定された分だけ Pad データである 0x80 を追加する．これにより，word 単位への変換が完了となる．図 5.2 では，パディングとして 1 バイト (1/2word) 分のデータが必要となったため，オフセットに 0x00 を格納している．

## 5.3 RFID リーダライタドライバ

RFID リーダライタドライバは，getAFI，setAFI，getDSFID，setDSFID，writeDataToUserMemory，readDataFromUserMemory を提供する．そして，隠蔽した機能として writeData と readData と createHandle を用意する．RFID リーダライタドライバ生成時に，Handle Manager のインスタンスを作成する．

(1) Object-Identifier: 1 0 15961 412 1
Object: 4911111000014

| | Precursor | | | | | | | | Offset | Length of Relative-OID | Relative-OID | Length of Object | Object | Pad |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | | | | | | |
| (1) | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 00 | 84 | 831C01 | 06 | 04777506CFCE | |

図 5.2　パディング後のデータフォーマット

### 5.3.1　writeDataToUserMemory

writeDataToUserMemory インタフェースはユーザデータの書き込みを行う．UHF 帯 RF タグでは，ユーザデータはユーザメモリに書き込みを行うが，先頭に DSFID が存在する．さらに DSFID は今後拡張されることが考えられている．そこでメソッド内部で，getDSFID コマンドを利用して，DSFID の長さを取得後，書き込みを行うアドレス (バイト単位) への変換を行い，そのアドレスデータと書き込むデータを writeData メソッドに渡す．

### 5.3.2　writeData

writeData メソッドは，渡されたデータを指定されたアドレスに 1word づつ，バックライトを用いた書き込みを行う．書き込みが成功したらその都度 Virtual Tag のアップデートを行う．handle 値が無効になった場合，createHandle を用いて handle 値を取得後，全データの読み込みを行い，Virtual Tag と比較を行う．もしも変更が加えられていなければ処理を続ける．writeData に渡される書き込み開始のアドレスはバイト単位であるので，ワード単位に変換を行う．変更後，開始アドレス，終了アドレスがワード単位と一致しない場合は，readData コマンドを用いて，不足しているデータを取得し，書き込みデータと結合後，書き込みを行う．

### 5.3.3　readDataFromUserMemory

readDataFromUserMemory インタフェースはユーザデータの読み込みを行う．UHF 帯 RF タグでは，ユーザデータはユーザメモリから読み込みを行うが，先頭に DSFID が存在する．さらに DSFID は今後拡張されることが考えられている．そこでメソッド内部で，getDSFID コマンドを利用して，DSFID の長さを取得後，読み込みを行うアドレス (バイト単位) への変換を行い，そのアドレスデータと読み込みデータの長さを readData メソッドに渡す．

### 5.3.4 readData

readData メソッドは，指定されたアドレスと，指定された長さ分のデータを Read コマンドを用いて取得する．読み込みが成功したらその都度 Virtual Tag のアップデートを行う．readData に渡されるデータの単位は，すべてバイト単位であるため，word 単位へ変換し，不足している前後データ分，指定されたデータ長に加えてから，読み込みを行う．また handle 値が有効時に，Virtual Tag に取得したいデータが存在すれば，Read コマンドは実行せず，Virtual Tag からデータを取得する．

### 5.3.5 createHandle

handle 値無効時に，Handle Manager から呼び出されるメソッドで，指定された RF タグに対して，Select コマンドと Inventory コマンドを実行し，handle 値を取得し，Handle Manager にて取得した時間と共に管理する．

## 5.4 Handle Manager

Handle Manager は，Select コマンド，Inventory コマンドで取得できる handle 値の管理を行う．Handle Manager は，RFID リーダライタドライバからの handle 値要求時に，生成された時間と，その経過時間のチェックを行い handle 値を戻す．もしも，handle 値が無効の場合，Handle Manager から RFID リーダライタ内に用意されているメソッドを用いて handle 値を生成し，再度管理を行う．

## 5.5 Virtual Tag

Virtual Tag は，物理的に読み書きを行なっている RF タグのユーザメモリを仮想的に再現する．Virutal Tag と実際の RF タグを同期することにより，handle 値の更新時に書き込みを即座に開始することが可能となる．例えば，追加を行うデータ量によっては，同一 handle 値を利用して書き込み処理が完了しない場合がある．その際には，Select コマンド，Inventory コマンドを行い，handle 値の更新を行う．そして Virtual Tag と現状の RF タグ内のデータが一致するかのチェックを行い，もし一致したならば，途中からの書き込みを再開する．

## 5.6 Handle Manager の実装確認

Handle Manager は，handle 値の読み込みシーケンスと書き込みシーケンス間で共通の値を利用しなくてはならない．しかし実際に読み込みシーケンスと書き込みのシーケンス間が繋がっているかは不明である．そこで本節では，Handle Manager が正しく動作しているかを検証する．表

5.1 の機材を図 5.3 に示すとおりに接続して，データを追加書き込みする実験を行った．書き込みの際には，事前に 1 組のデータがユーザメモリに存在している状況で行った．図 5.4 は，実際に装置を接続したときの様子である．

表 5.1　実装の確認環境

| | |
|---|---|
| RFID リーダライタ | UHF 帯 RFID 対応 RW 中型モジュール　評価キット [12] |
| アンテナ | 原田工業株式会 950AN2005L-018 |
| スペクトルアナライザ | ADAANTEST U3751 |

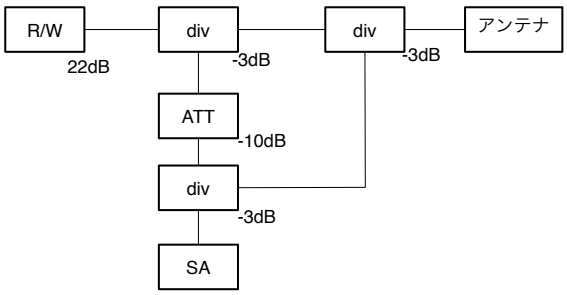


図 5.3　確認用装置のセットアップ概要

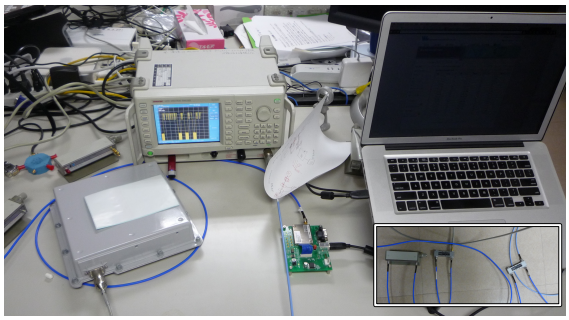

図 5.4　確認用装置のセットアップ

実験では，既存 RFID リーダライタドライバと本論文が提案している Handle Manager を搭載した RFID リーダライタドライバを用いた．

### 5.6.1　既存 RFID リーダライタドライバ

既存 RFID リーダライタドライバは第 2 章で述べたように，それぞれのインタフェースで handle 値が構築されため，最低でも読み込みシーケンスと書き込みシーケンスで 2 回以上の出力の途切れることが考えられる．

図 5.5 に，データの読み込みと解析部分の電波出力の状況を示す．データは読み込まれ逐次解析を行う手法をとっているため，その都度，handle 値が再構築されている．

図 5.6 に，データ追加処理時の writeDataToUserMemory の内容を示す．まず，writeDataToUserMemory 利用時に，指定された書き込みスタートのバイトが word 単位に変換した際に，書き込みデータが一部足りなかったため，一度 readDataFromUserMemory メソッドを用いて，Read コマンドを実行している．その後，再び handle 値を取得し write コマンドの実行を行っている．上記の結果から，既存 RFID リーダライタドライバは設計通り動作していることがわかる．

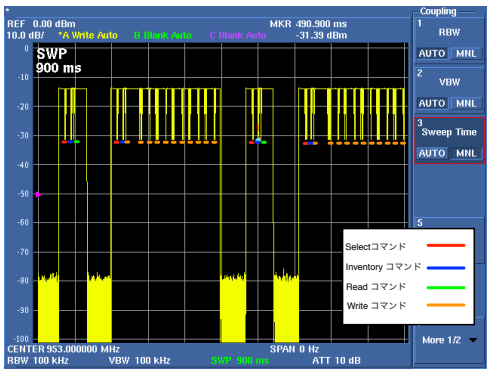


図 5.5　既存 RFID リーダライタドライバ: 読み込み部分の電波の様子

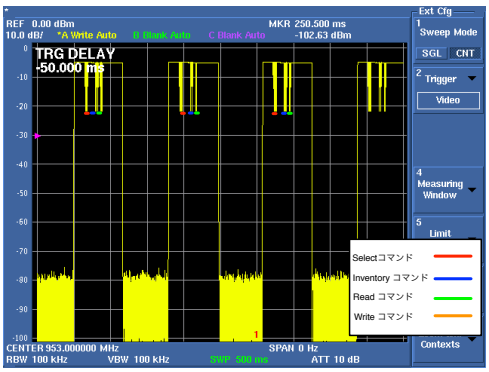


図 5.6　既存 RFID リーダライタドライバ: 書き込み部分の電波の様子

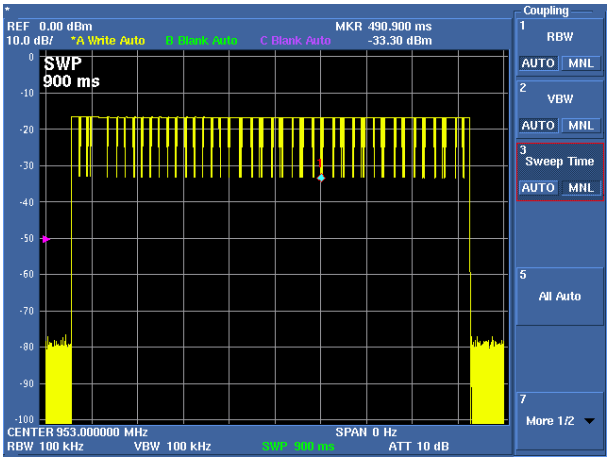


図 5.7　Handle Manager を用いたときの電波の様子

### 5.6.2　Handle Manager 搭載 RFID リーダライタドライバ

Handle Manager 搭載 RFID リーダライタドライバは，読み込みシーケンスと書き込みシーケンスで 1 回の handle 値の構築しか行われないため，出力は 1 回も切れないことが考えられる．

図 5.7 に，電波出力の結果を示す．読み書きシーケンスでそれぞれ別の RFID リーダライタドライバインタフェースを利用しているが，一度も RF タグへの出力が途切れることなくデータ追加を行っている．つまり読み込みシーケンスと書き込みシーケンス間で，データフォーマットの一貫性保証ができている．

## 5.7 本章のまとめ

本章では，補償型トランザクション機構の実装として，コマンド・レスポンスユニットの実装を中心に述べた．そして，RFID リーダライタドライバ，Handle Manager，Virtual Tag について述べた．最後に，Handle Manager が正しく実装されていることを確認するために，スペクトルアナライザなどを用いて出力の様子を測定し，データフォーマットの一貫性の確認を行った．次章では，補償型トランザクションの評価について述べる．

# 第 6 章

# 実験評価

本章では，補償型トランザクション機構の実験評価について述べる．まず，データの無効化が正しく動作するかを評価するために，書き込み途中に任意のタイミングで RF タグを RFID リーダライタから離し，不完全データが存在する状況を作り，有効性の評価を行う．次に，データフォーマットの一貫性保証による，データ追加の実行時間の変化を評価する．

## 6.1 実験概要

RF タグへのデータ追加の課題として，データの無効化とデータフォーマットの一貫性が挙げられる．本論文では，これらをそれぞれ評価する．

### 6.1.1 実験データ

サプライチェーンマネジメントを実現する RFID システムで利用されているデータ例として，グローバルロケーションナンバ（以下，GLN）とタイムスタンプを 1 組にして追加するアプリケーション例が存在する [3]．本論文の実験では，そこで利用されるデータと同じデータを用いることにした．GLN は，13 桁の数字からなる．Object Identifier を 1.0.15961.9.412.1 とし，データを 4911111000014 とすると，フォーマット後のデータは 9F0084831C080604777506CFCE となる．タイムスタンプは，年月日時分をそれぞれ 2 桁で表す．例えば、2011 年 1 月 12 日 10 時 10 分であれば 1101121010 と表す．Object Identifier を 1.0.15961.9.7003.1 とし，データを 1001011210 としたとき 1F84B65B08043BAA380A となる．

## 6.2 データの無効化の定量評価

本節では，データの無効化に対する定量評価について述べる．評価方針として，補償型トランザクション機構により，どの程度不完全データを無効化できるかを評価した．

### 6.2.1 実験の手順

まず，GLN とタイムスタンプが 1 組入っている RF タグを用意する．そして，不完全データを作成するために，新規データ追加時に RF タグを，RFID リーダライタから離す．エラーが起こった際に，不完全データがどのように解析されるかを調べる．表 6.1 に，本実験で利用するデータを示す．

表 6.1　RF タグ既存データと追加データ

| | |
|---|---|
| 既存データ | 1 0 15961 9 412 1:4911111000014 |
| | 1 0 15961 9 7003 1:1001011210 |
| 追加データ | 1 0 15961 9 412 1:4911111000015 |
| | 1 0 15961 9 7003 1:1101050240 |

### 6.2.2 実験結果

表 6.2 に，補償型トランザクション機構が無効の時の実験結果を示す．任意のタイミングで RF タグと RFID リーダライタを離した．その結果，試行 1，2，4，5 では，書き込んだデータとは全く異なるデータが解析された．そして，試行 3 では，解析結果を取得することができなかった．これは，3 番目のデータを解析する際にデータフォーマット的に解析が不可能となった．

表 6.3 に，補償型トランザクション機構を有効にしたときの実験結果を示す．試行 1，3，6 では，タイムスタンプの一部のデータが不完全データとして残存しているが，タイムスタンプの Precursor が書き込まれていないため解析は行われていない．そして，試行 2，4，7，8，9，10 では，GLN が不完全データとして残存しているが，Precursor が書き込まれていないため，読み込まれない．

表 6.2　補償型トランザクション無効時

| 試行回数 | データフォーマット | 解析データ |
|---|---|---|
| 1 | 099F0084831C010604777506CFCE1F84B65B010441A0190A<br>1F84831C020604770000000000000000000000000000000000 | 1 0 15961 9 412 1:4911111000014<br>1 0 15961 9 7003 1:1101011210<br>1 0 15961 9 412 2:4909147619328 |
| 2 | 099F0084831C010604777506CFCE1F84B65B010441A0190A<br>1F84831C020600000000000000000000000000000000000000 | 1 0 15961 9 412 1:4911111000014<br>1 0 15961 9 7003 1:1101011210<br>1 0 15961 9 412 2:0 |
| 3 | 099F0084831C010604777506CFCE1F84B65B010441A0190A<br>1F84831C000000000000000000000000000000000000000000 | 解析不能 |
| 4 | 099F0084831C010604777506CFCE1F84B65B010441A0190A<br>1F84831C020604777506CFCF1F84B65B0204000000000000 | 1 0 15961 9 412 1:4911111000014<br>1 0 15961 9 7003 1:1101011210<br>1 0 15961 9 412 2:4911111000015<br>1 0 15961 9 7003 2:0 |
| 5 | 099F0084831C010604777506CFCE1F84B65B010441A0190A<br>1F84831C020604777506CFCF1F84B65B020441A0FFE00000 | 1 0 15961 9 412 1:4911111000014<br>1 0 15961 9 7003 1:1101011210<br>1 0 15961 9 412 2:4911111000015<br>1 0 15961 9 7003 2:1101066624 |

表 6.3　補償型トランザクション機構有効時

| 試行回数 | データフォーマット | 解析データ |
|---|---|---|
| 1 | 099F0084831C010604777506CFCE1F84B65B01043BAA380A<br>1F84831C020604777506CFCE0000B65B02043BAA380A00 | 1 0 15961 9 412 1:4911111000014<br>1 0 15961 9 7003 1:1001011210<br>1 0 15961 9 412 2:4911111000014 |
| 2 | 099F0084831C010604777506CFCE1F84B65B01043BAA380A<br>00000000FFFF04777506CFCE00000000000000000000 | 1 0 15961 9 412 1:4911111000014<br>1 0 15961 9 7003 1:1001011210 |
| 3 | 099F0084831C010604777506CFCE1F84B65B01043BAA380A<br>1F84831C020604777506CFCE0000000002043BAA380A00 | 1 0 15961 9 412 1:4911111000014<br>1 0 15961 9 7003 1:1001011210<br>1 0 15961 9 412 2:4911111000014 |
| 4 | 099F0084831C010604777506CFCE1F84B65B01043BAA380A<br>0000831C020604777506CFCE00000000000000000000 | 1 0 15961 9 412 1:4911111000014<br>1 0 15961 9 7003 1:1001011210 |
| 5 | 099F0084831C010604777506CFCE1F84B65B01043BAA380A<br>00000000000004777506CFCE00000000000000000000 | 1 0 15961 9 412 1:4911111000014<br>1 0 15961 9 7003 1:1001011210 |
| 6 | 099F0084831C010604777506CFCE1F84B65B01043BAA380A<br>1F84831C020604777506CFCE00000000000000000000 | 1 0 15961 9 412 1:4911111000014<br>1 0 15961 9 7003 1:1001011210<br>1 0 15961 9 412 2:4911111000014 |
| 7 | 099F0084831C010604777506CFCE1F84B65B01043BAA380A<br>0000831C020604777506CFCE00000000000000000000 | 1 0 15961 9 412 1:4911111000014<br>1 0 15961 9 7003 1:1001011210 |
| 8 | 099F0084831C010604777506CFCE1F84B65B01043BAA380A<br>000000000020604777506CFCE00000000000000000000 | 1 0 15961 9 412 1:4911111000014<br>1 0 15961 9 7003 1:1001011210 |
| 9 | 099F0084831C010604777506CFCE1F84B65B01043BAA380A<br>00000000FFFF04777506CFCE00000000000000000000 | 1 0 15961 9 412 1:4911111000014<br>1 0 15961 9 7003 1:1001011210 |
| 10 | 099F0084831C010604777506CFCE1F84B65B01043BAA380A<br>00000000000000007506CFCE00000000000000000000 | 1 0 15961 9 412 1:4911111000014<br>1 0 15961 9 7003 1:1001011210 |

### 6.2.3　考察

結果をもとに，データの無効化についての考察を行う．補償型トランザクション無効時では，不完全データにより，意図したデータとは異なるデータを取得することになった．しかし，補償型トランザクション機構有効時では，全試行で不完全データの無効化ができていることを確認した．これにより，補償型トランザクション機構がないときのデータ追加ではデータ自身の信頼性に欠けることが言え，補償型トランザクション機構有効時では，データ自身の信頼性を上げることが出来ていると言える．

## 6.3　データフォーマット一貫性の定量評価

データフォーマットの一貫性によるデータ追加に要する実行時間を評価するために，2.3.4 で用いたのと同じ 2 つのコマンド・レスポンスユニットを用意し，補償型トランザクション機構によりどの程度有効に働いているかの評価する．

### 6.3.1　実験結果

図 6.1 に，実験結果を示す．x 軸は GLN とタイムスタンプの組数，y 軸は実行時間を示している．逐次解析・読み込みでは，追加組数に伴い，読み込みを行うデータ量と呼び出し回数が増えるため，実行時間が組数に応じて伸びている．実験で利用している RFID リーダライタは，1 秒で出力が切れてしまうというハードウェアの制限がある．そのため，6，7 組間で handle 値の再構築を行い，実行時間が伸びたと考えられる．一括読み取り・解析では，書き込みを行うデータ量が増えていないため，実行時間が一定になっている．

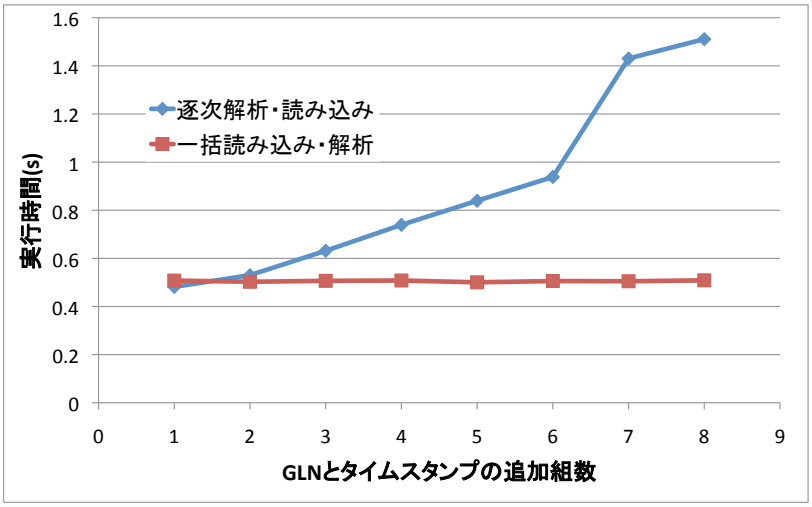


図 6.1　補償型トランザクション機構を用いたデータ追加

### 6.3.2　考察

実験結果をもとに，データフォーマットの一貫性ついての考察を行う．第 2.3.4 節の実験では，8 組目の書き込み時に，一括読み取り・解析型で，約 5 秒必要としていたデータ追加が 0.5 秒になり約 1/10 倍短くすることができ，最も遅い設計である逐次解析読み込みでは，約 30 秒必要としていたデータ追加が約 1.5 秒とり，約 1/20 倍なった．補償型トランザクション機構によるデータフォーマットの一貫性の保証により，データ追加の実行時間の短縮が可能になった．

## 6.4 本章のまとめ

本章では，補償型トランザクション機構の重要機能であるデータの無効化とデータフォーマットの一貫性の定量評価を行った．データの無効化では，不完全データが発生時した際に，データが無効になることを確認し，さらにデータ自身の信頼性を向上させることが可能であることを確認した．次章では，今後の課題について述べ，本論文をまとめる．

# 第 7 章

# 結論

本章では本論文のまとめを行う．まず本論文の背景として，複数 RFID システム間の情報共有を行うためのユーザメモリにデータを追加するアーキテクチャについて述べる．そして，ユーザメモリにデータを追加する際の問題について述べる．次に，本論文の提案手法である補償型トランザクション機構の詳細について述べる．そして，評価実験から得られた評価についてまとめ，最後に今後の課題について述べる．

## 7.1 本論文のまとめ

Passive 型 UHF 帯 RF タグは，サプライチェーンマネジメントの分野を中心に，あらゆる物に付けられることが期待されている．そして，日本のサプライチェーンのように，複数の RFID システム間で情報共有が必要となる状況では，RFID システムのモビリティを高めるために，共有する情報を RF タグの中に格納するアーキテクチャが必要になる．しかし，RF タグへのデータ追加には，不完全データが発生や，データフォーマットの一貫性を保証の必要性により，データ追加の大きなペナルティに繋がる．

そこで本論文では，データフォーマットとその解析手法に着目し，データパディング，Handle Manager，バックライトの 3 つのモジュールを利用した補償型トランザクション機構を提案した．データパディングは，完全なデータフォーマットの一貫性を行うため，1 データごとにデータ長を UHF 帯 RF タグの物理アドレスに対して最適化を行う．Handle Manager は，データフォーマットの一貫性を保証するため，読み込みシーケンスと書き込みシーケンスで必要となる handle 値の共通化を行う．バックライトは，不完全データが発生させないため，データフォーマットの解析手順とは逆の書き込み手順をとり，不完全データの無効化を行う．

補償型トランザクション機構の評価では，不完全データの無効化とデータフォーマットの一貫性についての定量評価を行った．不完全データ無効化の定量評価では，データ追加の最中に RF タグを任意のタイミングで RFID リーダライタから離すという手順をとり，不完全データを残存させた．既存の RFID リーダライタドライバを用いた場合は，意図していたデータではないデータを取得や，データフォーマットの破壊が確認できた．本提案手法有効時では，全試行回数で意図していないデータやデータフォーマットの破壊は発生しなかった．つまり，本提案手法は，データ自身の信頼性を向上させたと言える．補償型トランザクション機構のデータフォーマット一貫性の保証に対する定量評価では，既存のコマンド・レスポンスユニットを用いて，最大で約 30 秒から約 1.5 秒へ書き込み実行時間を短くすることが可能となった．RFID リーダライタドライバの特性を考えたコマンド・レスポンスユニットでは，最大で約 5 秒から約 0.5 秒へ書き込み実行時間を短くすることが可能となった．本提案機構により，RF タグへのデータ追加時間を短くすることが可能となった．また本提案手法は，全ての処理をミドルウェア内で完結することが可能であり，RFID ミドルウェアのモビリティ性についても優れていると考えられる．そのため，複数の RFID システム間での情報共有が可能であると考えられる．

## 7.2 今後の展望

本節では，本論文における今後の展望を述べる．以下に，本論文における今後の展望を整理する．

### 7.2.1 他のアクセスメソッドへの検討

本論文では，Non-Directory アクセスメソッドのみを対象としたが，その他にも，Directory，Tag-Data-Profile，Packed-Objects アクセスメソッドが存在する．そして，これら全てのアクセスメソッドは，Non-Directory と同じくデータフォーマット上でのトランザクション機能は用意されていない．そのため，これらのアクセスメソッドを安全に利用するためにも検討が必要である．

### 7.2.2 ペアデータによる書き込み手法の検討

本論文では，1 データ単位でのデータのトランザクションを実現した．しかし書き込みデータには，今回実験で利用しように，GLN とタイムスタンプで 1 組がそろうことで意味を成すデータが存在する．そこで，複数データ間でのトランザクションが必要になる．

### 7.2.3 新規データフォーマットの検討

RF タグへ格納するデータはある程度偏ってくることが考えられる．例えば，本論文が用いている GLN とタイムスタンプであれば，タイムスタンプは，前データの差分を格納するだけも算出が可能である．アプリケーションレイヤで差分だけの書き込みを指定することもできるが，差分書き込みにより，書き込むデータ量を減らすことができ，ユーザメモリの有効活用が期待できる．

# 謝辞

本研究の機会を与えてくださり，絶えず丁寧なご指導を賜りました，慶應義塾大学大学院政策・メディア研究科 徳田英幸博士に深く感謝致します．また，貴重なご助言を頂きました應義塾大学環境情報学部准教授 高汐一紀博士，同大学環境情報学部専任講師 中澤仁博士に深く感謝致します．また，慶應義塾大学徳田研究室の諸先輩方には折に触れ貴重なご助言を頂き，また多くの示唆を頂きました．そして，研究の日々を共に過ごした 生天目直哉氏，伊藤友隆氏，徳田義幸氏，山本純平氏をはじめとする ΔS103 の皆様と金澤貴俊氏，野沢高弘氏，小川正幹氏に深く感謝します．

また執筆にあたり，実験機材の提供をしていただいた東京工科大学コンピュータサイエンス学部教授 星徹博士，佐藤明雄博士に深く感謝します．そして，研究指導だけでなく，家族同然に同じ時間を過ごした，東京工科大学コンピュータサイエンス学部実験講師 江原正規氏，慶應義塾大学大学院理工学研究科 堀口悟史氏には深く感謝します．

最後に，長い研究生活を送るチャンスを与えてくださった兄・米澤拓郎氏，身近な心の支えとなってくれた桑原ひとみ氏に深く感謝します．

2011 年 2 月 14 日

米澤　祐紀

# 参考文献


[1] A. Ilic, T. Andersen, and F. Michahelles. Increasing supply-chain visibility with rule-based rfid data analysis. *Internet Computing, IEEE*, Vol. 13, No. 1, pp. 31 –38, 2009.

[2] Bilal Hameed, Imran Khan, Frank Durr, and Kurt Rothermel. An rfid based consistency management framework for production monitoring in a smart real-time factory. In *Internet of Things (IOT), 2010*, 29 2010.

[3] 吉田真樹, 戸田暁博, 江原正規, 井上亮文, 星徹. RF タグのユーザメモリを用いた流通経路記録手法の提案. 全国大会講演論文集, No. 3, pp. 299–301, 2009.

[4] 社団法人食品需給研究センター. 新技術活用ビジネスモデル実証・普及事業「流通効率化の推進」報告書. March 2010.

[5] EPCglobal. `http://www.gs1.org/epcglobal`, January 2011.

[6] 椎橋章夫. IC カード出改札システム suica の開発と導入. *The journal Reliability Engineering Association of Japan*, pp. 718–727, 2003.

[7] Donghun Yoon. A generic conceptual model and actual systems of IC-Card system for model-based systems engineering. *Model-Based Systems Engineering*, pp. 69–74, 2009.

[8] ATA Spec 2000. `http://www.spec2000.com/`, January 2011.

[9] Craig K. Harmon. The necessity for a uniform organisation of User Memory in RFID. *Radio Frequency Identification Technology and Applications*, Vol. 1, No. 1, pp. 41–51, 2006.

[10] Kenneth P. Fishkin, Bing Jiang, Matthai Philipose, and Sumit Roy. I Sense a Disturbance in the Force: Unobtrusive Detection of Interactions with RFID-tagged Objects. *UBIQUITOUS COMPUTING*, Vol. 3205, pp. 268–282, 2004.

[11] 苧阪浩輔, 三次仁, 中村修, 村井純. UHF 帯 RFID タグの適応読み取り制御. 電子通信情報学会報告書, Vol. 107, No. 547, pp. 29–33, 2008.

[12] UHF 帯 RFID 対応 RW モジュール　評価キット. `http://www.hitachi-jten.co.jp/products/uhf-rfid/index.html`, May 2009.

[13] ISO/IEC15961. Infomation technology - Radio frequency identification(RFID) for item management - Data protocol:application interface, 2004.

[14] ISO/IEC15962. Infomation technology - Radio frequency identification(RFID) for item

management - Data protocol:data encoding rules and logical memory functions, 2004.

[15] ISO/IEC18000-6. Information technology - Radio frequency identification for item management - part6: Parameters for air interface communications at 860MHz to 960MHz, 2004.

[16] Low Level Reader Protocol Standard. `http://www.gs1.org/gsmp/kc/epcglobal/llrp`, December 2010.

[17] Koseke Osaka, Jin Mitsugi, and Jun Murai. Generalized Handling of User-Specific Data in Networked RFID. *The Internet of Things*, Vol. 4952/2008, pp. 173–183, 2008.

[18] Seok-Young Jang, Sang-Hwa Chung, and Seong-Joon Lee. An Effective Design of an Active RFID Reader Using a Cache of Tag Memory. *Emerging Technologies in Knowledge Discovery and Data Mining*, Vol. 4819, pp. 584–595, 2009.

[19] Christian Florkemeier and Matthias Lampe. RFID middleware design: addressing application requirements and RFID constraints. *Smart objects and ambient intelligence*, pp. 219–224, 2005.

[20] Wooseok Ryu and Bonghee Hong. A Reprocessing Model Based on Continuous Queries for Writing Data to RFID Tag Memory. *Database Systems for Advanced Applications*, pp. 201–214, 2009.

[21] The felica system. `http://www.sony.net/Products/felica/about/scheme.html`, January 2011.