平成7年度

電子メディアに関する調査研究

内容アクセスマネージャの標準化に関する調査研究

報　告　書

平成8年3月

財団法人　ニューメディア開発協会

はじめに

ICカード等の可搬性メディアは、現在国内において医療・行政等の公共分野から様々な民間事業分野に至るまで利用が拡大されている。しかし、ICカード等のカードシステムの大半は、事業主体が個別に事業化を進めているため、カード内データとアプリケーションプログラムが相互に依存性の高い関係となり、オープン化が困難な状況となっている。

通商産業省では、関係省庁の協力の下にICカード等の普及促進を目的として、次世代ICカード等の検討を提唱してきた。これに基づき、財団法人ニューメディア開発協会では、「ICカード等多目的利用研究会」を平成4年度に発足させ、ICカードシステムのオープン化・マルチベンダ化を実現するため、内容アクセスマネージャ（Content Access Manager、略称CAM）についての機能仕様をまとめ、CAM標準ソフトウェアを開発してきた。

一方、平成7年10月には、国際標準規格案ISO/IEC　DIS7816-4に対応した日本工業規格JIS　X6306が制定され、このJIS　X6306に準拠したICカード（以下、JIS準拠ICカードという）が国内において標準化されていく状況となった。

本報告書では、上記のJIS準拠ICカードに対応するCAM（Version2.0）の機能仕様について記述すると共に、この機能仕様に基づいて開発したCAM標準ソフトウェア（Version2.0）のプログラムドキュメントを添付した。今後、この報告書がICカード等の普及促進の一助となれば幸いである。

最後に、CAMの機能仕様に関する検討や、CAM標準ソフトウェアの開発にご協力をいただきました研究会及びワーキンググループの各メンバ、調査研究の委託先である株式会社富士総合研究所の各位、さらには積極的なご支援を賜りました関係省庁の方々に、厚く御礼申し上げると共に今後なお一層のご支援をお願いする次第である。

平成8年3月

財団法人　ニューメディア開発協会

【目次】

【概要】

【本編】

【概要】

1.調査研究の背景

現在、国内では、21世紀に到来する高度情報通信社会を支えるためのディジタルテクノロジーであるインターネット・マルチメディア等を核とした情報通信インフラストラクチャの整備に向けて、国を中心として研究機関・民間企業が共同で様々なアクションプログラムの提案やパイロット事業の実施等を積極的に展開している。

こうした中、大容量な情報記録機能に加えて高いセキュリティ機能を持つ携帯性メディアであるICカードは、高度情報通信社会における個人向けの情報提供や取引決済等の業務サービスを質的かつ量的に向上するための電子的ツールとして位置づけられ、公共並びに各種の民間分野において導入、実用化が進められている。

しかしながら、国内の各種分野・業種で利用されているICカードシステムでは、カード内のデータがホストコンピュータ内のアプリケーションプログラム構成に大きく依存しており、加えてカードメーカ各社の製造・販売するICカードの詳細な機能仕様がメーカ各社毎に異なっている等のコンピュータシステムにおける典型的な課題を抱えており、ICカードシステム間の相互運用性を実現することが困難となっている。

以上のような背景の下、財団法人ニューメディア開発協会はICカードを始めとするカードメディアの更なる普及促進に向けて「ICカード等多目的利用研究会」を発足させ、第一の課題を克服する一つの方向性としてICカードシステムのオープン化・マルチベンダ化の必要性を提唱した。その結果として、新しい概念としての内容アクセスマネージャ（Content Access Manager　以下、CAMという）を提案すると共に同ソフトウェアの開発を行うことによってその有効性を実証した。

また、第二の課題に関しては、従来からISO/IEC JTC1/SC17/WG4を中心として進められてきたICカードの国際標準規格7816シリーズのインターインダストリコマンド等に関する標準化作業が昨今になり、最終段階に移行したことから、国内においてISO/IEC 7816-4に対応した日本工業規格(JIS)の制定作業が進行している状況にある。

今後、利用分野の拡大及びアプリケーションサービスの多様化に柔軟に対応でき、かつ異なるシステム間でのデータ相互運用を可能とするICカードシステムを実現し普及啓蒙していくために、二つの課題解決の方策である内容アクセスマネージャと日本工業規格（JIS）に準拠したICカードを連結することによって、国内標準として位置づけられるICカードシステムを構築することが急務であると指摘されている。

2.調査研究の経緯

本調査研究は、平成4年度より継続的に実施されてきた経緯があり、初年度では、現在利用されているICカードシステムに関して多目的利用の観点から整理された課題を踏まえ、次世代ICカード

システムの要求定義・機能仕様について検討考察を行い、次世代ICカードシステムの一方向性を示した。その結果、次世代ICカードに要求される機能の一部を実現するための新しい概念として内容アクセスマネージャ（以下、CAMという）を提案すると共に、CAMに関する標準的なソフトウェア（以下、CAM標準ソフトウェアという）を設計・開発する必要性を説いた。

引き続き、平成5年度では、平成4年度の研究成果であるCAMの概念に基づいて、カード内のデータに対するアクセス制御機能を組み込んだCAM　Version1.0に関する機能仕様書をまとめ、CAM標準ソフトウェアVersion1.0を開発した。そして、最終的に参画企業による動作確認・機能検証を通じて、機能及びインタフェース条件等を確定した。

さらに、平成6年度では、CAM　Version1.0が様々な業種・分野に適用され始めたことにより顕在化してきた機能追加等のニーズに対応すべく、CAM Version1.0に関する機能仕様書及び標準ソフトウェアのレビュー及びバージョンアップを図り、CAM　Version1.1に関する機能仕様書をまとめ、CAM標準ソフトウェアVersion1.1を開発した。

3.調査研究の目的

本調査研究では、将来的な利用分野の拡大およびアプリケーションサービスの多様化に柔軟に対応可能である、柔軟性が高く使い勝手の良い『次世代カードシステム』の開発・普及促進を行うことを主たる目的としている。

従って、平成7年度では、過去三年間における調査研究成果を集約すべく、CAM Version1.0,1.1の対象である16社仕様準拠ICカードに加え、標準化作業が大詰めを迎えつつある日本工業規格（JIS）に準拠したICカード（以下、JIS準拠ICカードという）に対応したCAM Version2.0に関する機能仕様書を取り纏めると共に標準ソフトウェアを開発する。

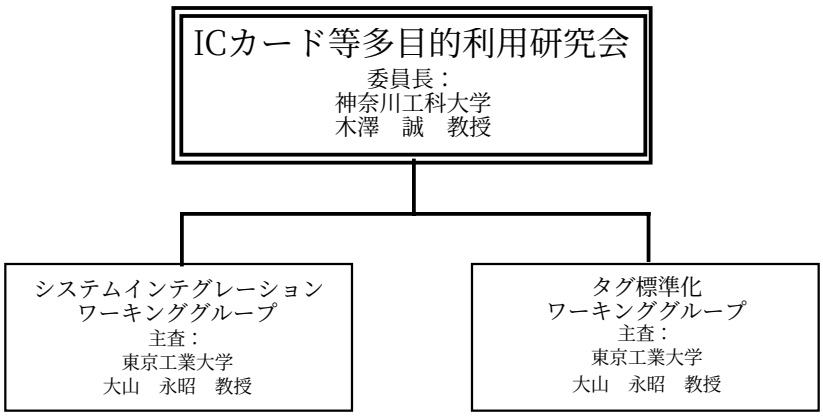
さらに、平成7年度では、カード所有者が公共・民間分野を問わず複数の様々なアプリケーションサービスを一枚のカードメディアによって広域的にかつ共通的に享受できる多目的利用ICカードシステムに関する利用環境を健全にかつ円滑に実現していくために、CAM Version2.0等の技術的な基盤整備だけでなく、環境的な基盤整備を推進するための基本方針、作業手順、整備項目等について意見交換を行い、あるべき方向性を探る。

4.実施体制

本調査研究を実施するための組織対応として、平成6年度に引き続き財団法人ニューメディア開発協会に『ICカード等多目的利用研究会』を設ける。

また、本調査研究を効率的に遂行するために、同研究会の下にCAM Version2.0に関する機能仕様書の作成及び標準ソフトウェアの開発を行う『システムインテグレーションワーキンググループ』

と、多目的利用ICカードシステムの環境的な基盤整備に関する検討を行う『タグ標準化ワーキンググループ』を設置する。『IC カード等多目的利用研究会』、『システムインテグレーションワーキンググループ』及び『タグ標準化ワーキンググループ』の組織構成案は図。の通りである。



図。　実施体制

さらに、必要に応じて『システムインテグレーションワーキンググループ』の配下に、少数の委員から構成されるサブワーキンググループを適宜設けて、ワーキンググループにおける議論のための原案作成等の作業を行う。

ICカード等多目的利用研究会　委員名簿

［順不同、敬称略］

委員長	木澤　誠	神奈川工科大学情報工学科教授
委　員	伊東　一良	大阪大学大学院工学研究科応用物理学専攻教授
委　員	大山　永昭	東京工業大学像情報工学研究施設教授
委　員	高橋　隆	京都大学医学部付属病院医療情報部教授
委　員	上田　博三	厚生省健康政策局総務課医療技術情報推進室長
委　員	河野　秀樹	通商産業省機械情報産業局情報処理システム開発課長
委　員	星野　茂夫	運輸省運輸政策局情報管理部情報企画課長
委　員	木下　誠也	建設省建設大臣官房技術調査室環境安全技術調整官
委　員	吉田　哲	自治省自治大臣官房情報管理室長
委　員	松尾　庄一	警察庁交通局運転免許課長
事務局	国分　明男	財団法人ニューメディア開発協会理事兼開発本部長
事務局	陸田　耕吾	財団法人ニューメディア開発協会開発本部システム開発部担当部長
事務局	原　秀昭	株式会社富士総合研究所研究開発第2部主任研究員
事務局	丹波　伸行	株式会社富士総合研究所研究開発第2部研究員

システムインテグレーションワーキンググループ　委員名簿（その1）

［順不同、敬称略］

主　査	大山　永昭	東京工業大学像情報工学研究施設教授
幹　事	宮崎　順介	富士通株式会社情報処理事業本部企画部担当部長
委　員	今村　健一	日本アイ・ビー・エム株式会社 流通インダストリー・ソリューション第1ソリューション開発担当課長
委　員	奥田　雅夫	株式会社日立製作所システム事業部都市開発システム部主任技師
委　員	国分　明男	財団法人ニューメディア開発協会理事兼開発本部長
委　員	高木　一圭	株式会社東芝ソフトウェア第三部特器ソフト担当主務
委　員	高木　伸哉	松下電池工業株式会社応用機器事業部SCB　SC開発課技師
委　員	田中　武	NTTデータ通信株式会社技術開発本部 マルチメディア技術センタセキュリティ担当
委　員	小杉　哲	日本電気株式会社 C&Cシステム市場開発推進本部カードビジネス統括部長
委　員	滝沢　俊男	沖電気工業株式会社情報通信システム事業本部 カードシステムビジネス推進部技術担当課長
委　員	宮野　哲紀	大日本印刷株式会社CBS開発本部カードシステム第一部課長補佐
委　員	村松　正男	共同印刷株式会社技術本部技術第三部副技師
委　員	谷内田益義	株式会社リコー研究開発本部中央研究所 情報エレクトロニクス研究センター係長研究員
委　員	山口　雅浩	東京工業大学像情報工学研究施設助手
委　員	寄本　義一	凸版印刷株式会社証券システム研究所ICカードシステム開発室課長

システムインテグレーションワーキンググループ　委員名簿（その2）

［順不同、敬称略］

オブザーバ	渡部　豪	厚生省健康政策局総務課医療技術情報推進室長
オブザーバ	藤原　達也	通商産業省機械情報産業局情報処理システム開発課開発係長（平成7年12月まで）
オブザーバ	梅沢　茂之	通商産業省機械情報産業局情報処理システム開発課開発係長（平成8年1月より）
オブザーバ	青柳　肇	通商産業省機械情報産業局電子機器課通信機器班長
オブザーバ	相原　慎哉	工業技術院標準部情報規格課
オブザーバ	小玉　真一	運輸省運輸政策局情報管理部情報企画課専門官
オブザーバ	鎌田　高造	建設省建設大臣官房技術調査室技術審議官付課長補佐
オブザーバ	石川　家継	自治省自治大臣官房情報管理室企画係長
オブザーバ	佐々木洋吾	警察庁交通局運転免許課課長補佐
オブザーバ	松島　修平	警察庁交通局運転免許課係長
オブザーバ	川辺　俊一	警察庁情報通信局通信施設課課長補佐
オブザーバ	今井　雅俊	ICカードシステム利用促進協議会企画調査担当部長

事務局	陸田　耕吾	財団法人ニューメディア開発協会開発本部システム開発部担当部長
事務局	原　秀昭	株式会社富士総合研究所研究開発第2部主任研究員
事務局	丹波　伸行	株式会社富士総合研究所研究開発第2部研究員

タグ標準化ワーキンググループ　委員名簿

［順不同、敬称略］

主　査	大山　永昭	東京工業大学像情報工学研究施設教授
委　員	渡部　豪	厚生省健康政策局総務課医療技術情報推進室長
委　員	藤原　達也	通商産業省機械情報産業局情報処理システム開発課開発係長（平成7年12月まで）
委　員	梅沢　茂之	通商産業省機械情報産業局情報処理システム開発課開発係長（平成8年1月より）
委　員	相原　慎哉	工業技術院標準部情報規格課
委　員	小玉　真一	運輸省運輸政策局情報管理部情報企画課専門官
委　員	三宅　智	労働省労働基準局安全衛生部労働衛生課労働技官
委　員	鎌田　高造	建設省建設大臣官房技術調査室技術審議官付課長補佐
委　員	石川　家継	自治省自治大臣官房情報管理室企画係長
委　員	佐々木洋吾	警察庁交通局運転免許課課長補佐
委　員	井上　剛	財団法人金融情報システムセンター業務調査部業務調査第2課調査役
オブザーバ	松島　修平	警察庁交通局運転免許課係長
オブザーバ	川辺　俊一	警察庁情報通信局通信施設課課長補佐
オブザーバ	加藤　公也	財団法人医療情報システム開発センター研究開発部研究開発第一課長
オブザーバ	国分　明男	財団法人ニューメディア開発協会理事兼開発本部長
オブザーバ	中川　浩徳	財団法人流通システム開発センター情報システム部次長
オブザーバ	林　義昭	大日本印刷株式会社CBS開発本部カードシステム第一部次長
オブザーバ	平松　雄一	沖電気工業株式会社情報通信システム事業本部技師長
オブザーバ	茂田井省三	株式会社東芝機器事業部機器システム技術第一部部長附
事務局	陸田　耕吾	財団法人ニューメディア開発協会開発本部システム開発部担当部長
事務局	原　秀昭	株式会社富士総合研究所研究開発第2部主任研究員
事務局	丹波　伸行	株式会社富士総合研究所研究開発第2部研究員

5.調査研究の範囲

5.1　概要

本調査研究の概要は、以下の通りである。（図「参照）

5.1.1　ICカード等多目的利用研究会

ICカード等多目的利用研究会では、平成7年度の調査研究対象であるCAM　Version2.0に関する機能仕様書の作成及び標準ソフトウェアの開発、多目的利用ICカードシステムに係わる利用環境の基盤整備に関する検討等の活動方針を明らかにすると共に、それらの最終成果について審議を行う。

(1) CAM Version2.0に関する機能仕様書の作成及び標準ソフトウェアの開発方針の明確化

(2) 多目的利用ICカードシステムに係わる利用環境の基盤整備に関する検討方針の明確化

(3) 上記二点に関する審議

5.1.2　システムインテグレーションワーキンググループ

システムインテグレーションワーキンググループでは、CAM Version2.0に関する機能仕様書を作成すると共に標準ソフトウェアを開発する。

(1) CAM標準ソフトウェアVersion2.0の設計・開発

(2) CAM標準ソフトウェアVersion2.0の総合検証・評価

(3) CAM Version2.0機能仕様書の作成

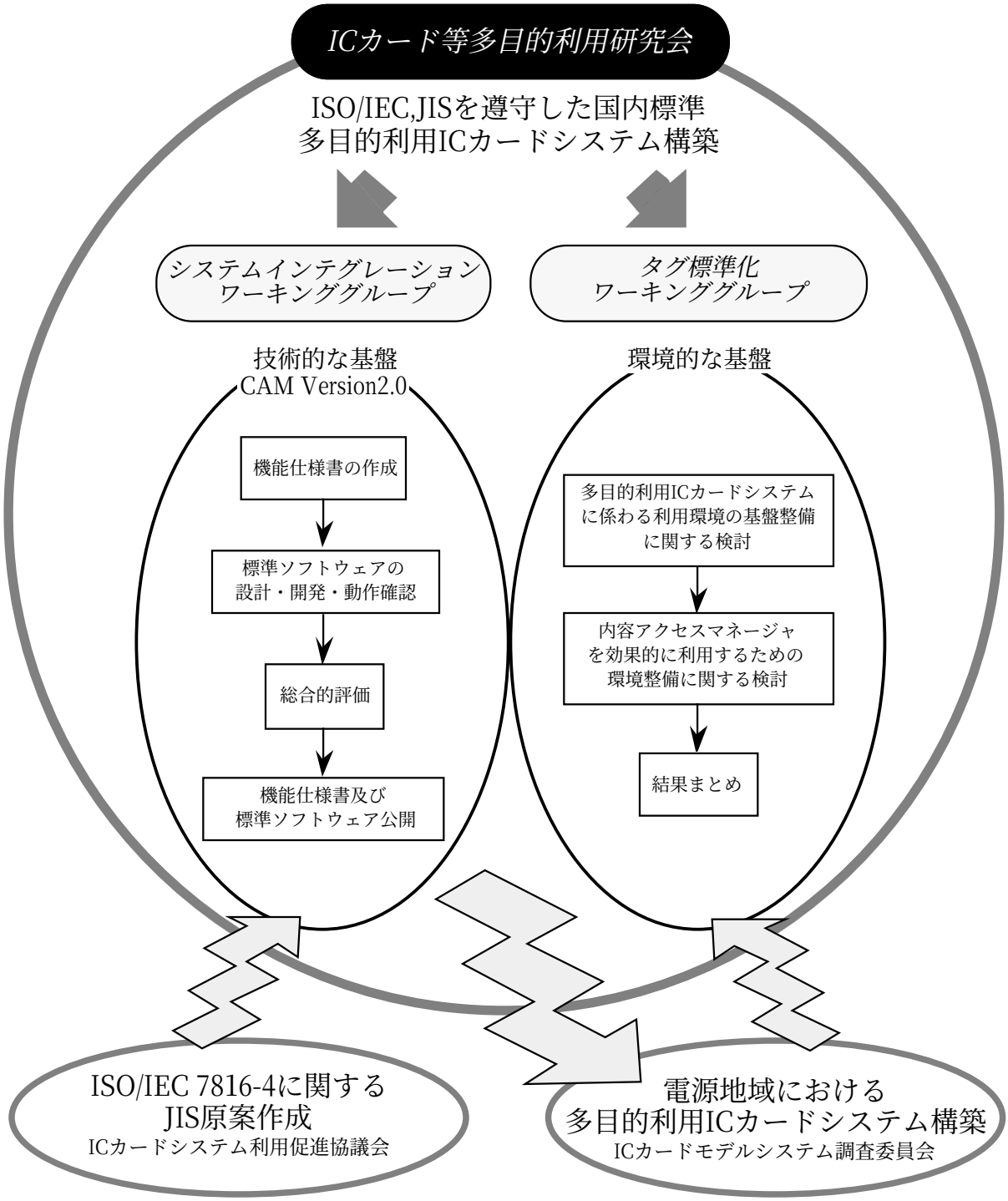
5.1.3　タグ標準化ワーキンググループ

タグ標準化ワーキンググループでは、多目的利用ICカードシステムに関する利用環境を健全にかつ円滑に実現していくために不可欠である環境基盤整備に関する基本方針、作業手順、整備項目等について検討を行う。

さらに、以上の検討経緯を踏まえ、本研究会における成果である内容アクセスマネージャを効果的に利用するための環境整備についても併せて検討する。

(1) 多目的利用ICカードシステムに係わる利用環境の基盤整備に関する検討

(2) 内容アクセスマネージャを効果的に利用するための環境整備に関する検討


ICカード等多目的利用研究会
ISO/IEC,JISを遵守した国内標準
多目的利用ICカードシステム構築
システムインテグレーション
ワーキンググループ
タグ標準化
ワーキンググループ
技術的な基盤
CAM Version2.0
機能仕様書の作成
標準ソフトウェアの
設計・開発・動作確認
総合的評価
機能仕様書及び
標準ソフトウェア公開
環境的な基盤
多目的利用ICカードシステム
に係わる利用環境の基盤整備
に関する検討
内容アクセスマネージャ
を効果的に利用するための
環境整備に関する検討
結果まとめ
ISO/IEC 7816-4に関する
JIS原案作成
ICカードシステム利用促進協議会
電源地域における
多目的利用ICカードシステム構築
ICカードモデルシステム調査委員会

図「　調査研究概要

5.2　調査方法

5.2.1　ICカード等多目的利用研究会

(1)CAM Version2.0の開発方針の明確化

多目的利用ICカードシステムを実現するための一技術的基盤として位置づけられるCAMをレビューすることによってVersion2.0へのバージョンアップを図るべき方向性を明確にする。

(2)多目的利用ICカードシステムに係わる利用環境整備の検討方針の明確化

多目的利用ICカードシステムに関する利用環境を健全にかつ円滑に実現していくために、整備すべき環境的な基盤に関する検討方針を明確にする。

(3)上記二点に関する審議

システムインテグレーションワーキンググループによる成果であるCAM　Version2.0に関する機能仕様書及び標準ソフトウェア、またタグ標準化ワーキンググループによる成果である多目的利用ICカードシステムに係わる利用環境整備に関する検討結果を総括すると共に審議を行う。

5.2.2　システムインテグレーションワーキンググループ

(1)CAM標準ソフトウェアVersion2.0の設計・開発

16社仕様準拠ICカードを対象として研究・開発したCAM標準ソフトウェアVersion1.1をJIS準拠ICカードへ対応可能とするために、機能仕様、関数仕様等の見直しを図り、CAM標準ソフトウェアVersion2.0の設計・開発を行う。

(2)CAM標準ソフトウェアVersion2.0の総合検証・評価

上記に従い設計・開発されたCAM標準ソフトウェアVersion2.0について動作確認、機能検証等を実施することによって、CAM標準ソフトウェアVersion2.0の機能仕様及び関数仕様を明らかにする。

(3)CAM Version2.0の機能仕様書の作成

CAM標準ソフトウェアVersion2.0の動作確認および機能検証結果に基づき、最終的にCAM Version2.0の機能仕様書を作成する。

5.2.3　タグ標準化ワーキンググループ

(1)多目的利用ICカードシステムに係わる利用環境の基盤整備に関する検討

将来の高度情報通信社会の到来に備え、多目的利用ICカードシステムに関する利用環境を健全にかつ円滑に実現していくために不可欠である環境基盤整備に関する基本方針、作業手順、整備項目等について包括的な議論を行う。

(2)内容アクセスマネージャを効果的に利用するための環境整備に関する検討

多目的利用ICカードシステムに関する利用環境を健全にかつ円滑に実現していくために不可欠である環境基盤を整備していく中で、とりわけ内容アクセスマネージャを効果的に利用するために必要不可欠な環境整備について検討する。

6.委員会活動状況

6.1　ICカード等多目的利用研究会

平成7年度のICカード等多目的利用研究会は、平成7年7月より次の通り開催した。

回次	開催日	議事次第
第1回	平成7年 7月19日	1. 平成7年度ICカード等多目的利用に関する調査研究計画について 2. システムインテグレーションワーキンググループの進め方 3. タグ標準化ワーキンググループの進め方
第2回	平成8年 3月27日	1. システムインテグレーションワーキンググループの審議経緯報告 2. タグ標準化ワーキンググループの審議経緯報告 3. 平成7年度報告書案に関する審議

6.2　システムインテグレーションワーキンググループ

システムインテグレーションワーキンググループは、平成7年10月より次の通り開催した。

回次	開催日	議事次第
第1回	平成7年10月 9日	1. 平成7年度ICカード等多目的利用に関する調査研究について 2. JIS準拠ICカードを対象とするCAMの考え方 3. CAM Version1.1及びVersion2.0に関する報告・説明
第2回	平成7年10月30日	1. CAM Version2.0の開発における問題点への対応 2. CAM Version2.0の動作確認・機能検証について
第3回	平成7年12月 4日	1. CAM Version2.0の開発方針に関する審議 2. CAM Version2.0の動作確認・機能検証に関する報告 3. CAM Version2.0において新たに機能向上・改良すべき点について
第4回	平成7年12月15日	1. CAM Version2.0の動作確認・機能検証に関する報告 2. CAM Version2.0において新たに機能向上・改良すべき点について
第5回	平成8年 1月16日	1. CAM Version2.0の動作確認・機能検証に関する報告 2. CAM Version2.0において新たに機能向上・改良すべき点について 3. 内容アクセスマネージャの利用に対する考え方
第6回	平成8年 2月27日	1. CAM Version2.0の動作確認・機能検証に関する報告 2. CAM Version2.0において新たに機能向上・改良すべき点について 3. カード所有者認証のためのPINの設定 4. 内容アクセスマネージャの利用に対する考え方
第7回	平成8年 3月22日	1. 1. CAM Version2.0の動作確認・機能検証に関する報告 2. CAM Version2.0におけるデバイスドライバ・インタフェース 3. 内容アクセスマネージャの利用に対する考え方

6.3　タグ標準化ワーキンググループ

タグ標準化ワーキンググループは、平成7年10月より次の通り開催した。

回次	開催日	議事次第
第1回	平成7年10月20日	1. 平成7年度ICカード等多目的利用に関する調査研究について 2. 内容アクセスマネージャに関する概要説明 3. タグ標準化ワーキンググループのテーマ及び進め方
第2回	平成7年11月13日	1. アプリケーションサービスの分類体系の必要性について 2. ICカードアプリケーション識別子の付番制度に関する説明 3. ICカードアプリケーション識別子と分類体系の関係について
第3回	平成7年12月6日	1. タグ標準化ワーキンググループにおける検討経緯の整理 2. タグ標準化ワーキンググループの今後の作業の進め方 3. 内容アクセスマネージャを対象としたICカードアプリケーション識別子の考え方
第4回	平成8年 1月11日	1. ICカードアプリケーション識別子の申請・登録に関する国内規則（案）について 2. 内容アクセスマネージャ用ICカードアプリケーション識別子の付番制度について 3. 内容アクセスマネージャを利用するための環境整備
第5回	平成8年 2月15日	1. タグ標準化ワーキンググループの検討経緯の整理 2. ICカードアプリケーション識別子の申請・登録に関する国内規則（案）について

7.調査研究のまとめ

7.1　内容アクセスマネージャの概念

ICカード等多目的利用研究会より平成4年度に提案された内容アクセスマネージャについて、背景及び目的、位置づけ、システム構成、要求機能等を概説する。

7.2　内容アクセスマネージャVersion2.0の機能仕様

平成5年度成果であるCAM Version1.0及び平成6年度成果であるCAM Version1.1は、いずれもISO/IECにおける国際標準化作業の進捗等の諸事情に鑑みて、国内における暫定標準として位置づけられた16社仕様準拠ICカードを対象として研究・開発されたソフトウェアであるため、工業技術院によってJISが制定されたあかつきには、JIS X6303,X6304, X6306に準拠したICカード（以下、JIS準拠ICカードという）への対応を図ることが肝要であると指摘されていた。

平成7年度になり、工業技術院によってJIS X6306が制定されたことを踏まえ、本年度では、従来からの16社仕様準拠ICカードだけでなく、新たなJIS準拠ICカードへの対応を図るべく、既に公開されているCAM Version1.1に関する機能仕様書及び標準ソフトウェアに関するバージョンアップを行い、CAM標準ソフトウェアVersion2.0を開発すると共に、CAM機能仕様書Version2.0をまとめる。

以下、CAM Version2.0において新たに対象に組み込まれたJIS準拠ICカードに関する機能仕様について説明する。また、巻末の【参考資料】として、CAM標準ソフトウェアVersion2.0 のプログラムドキュメントを添付する。

7.2.1　データ動的変更機能

ICカード内のデータがホストシステムを構成するハードウェア及びソフトウェアに依存せず、ICカード内の各ファイルに記録されるデータの実体を動的に変更可能である機能を実現するために、CAM Version2.0が対象とするデータフォーマットを規定する。

7.2.2　ファイルサイズ動的変更機能

ICカード内の各ファイルに記録されるデータの総量を変更可能である機能を実現するために、CAM Version2.0が対象とするカード所有者が持つ一般のカード（以下、データカードという）及びそれを構成するDF情報ファイル、拡張ファイル等に関するフォーマットを規定

する。

7.2.3　セキュリティ機能

ファイル単位だけでなくデータ単位でのきめ細かなセキュリティ管理を可能とする機能を実現するために、CAM　Version2.0が対象とするオペレーションカード及びそれを構成するDF情報ファイル、所有者情報ファイル、キーテーブルファイル等に関するフォーマットを規定する。

7.2.4　CAM Version2.0における新規機能

CAM Version2.0では、CAM Version1.0,1.1に既に組み込まれていた機能であるデータ動的変更機能、ファイルサイズ動的変更機能、セキュリティ機能以外に、カード所有者認証のためのPIN設定機能と16社仕様準拠ICカード・JIS準拠ICカードの共存機能を新たに組み込むことから、それらを実現するための前提条件及び具体的方策を明確にする。

7.3　内容アクセスマネージャVersion2.0の関数仕様

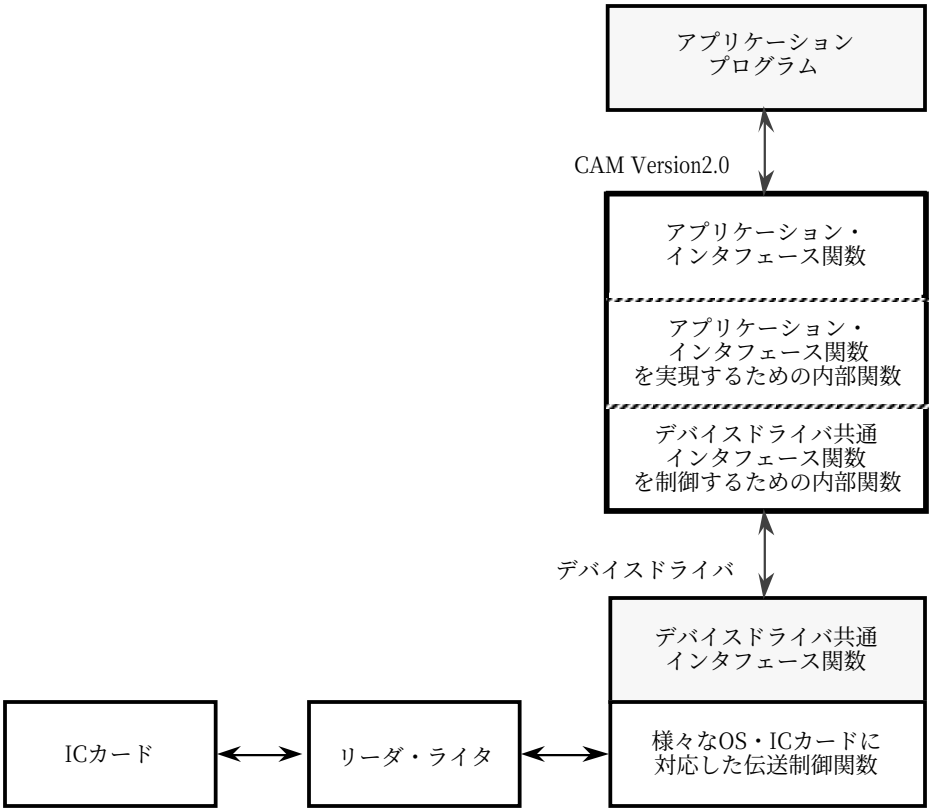
CAMは、カードメーカ、電機メーカ各社等により製造される仕様が異なるICカードを、また16社仕様準拠ICカードやJIS準拠ICカード等のように世代が異なるICカードを相互に運用可能とすることを主たる目的として提案・開発された概念である。将来、電機メーカ、システムインテグレータ等が、CAMに規定された関数仕様に基づき、処理速度等の性能向上を目的としてCAM標準ソフトウェアを改変する、また新規に開発する場合に、CAMに規定された同一関数によるリターン値が異なる、また関数を実行する順序が異なる等の事態を是非とも回避しなければならないが、その反面実現方法等を制限しないことが極めて重要である。

以上を踏まえ、CAM　Version2.0は、上記要件を満たすべく、構成する関数を図」に示す通り、CAMとアプリケーションプログラムとのインタフェースを規定するためのアプリケーション・インタフェース関数、アプリケーション・インタフェース関数を実現するための内部関数、そしてデバイスドライバ共通インタフェース関数を制御するための内部関数の3階層に分割することによって、下位に位置するデバイスドライバ関数とのインタフェース条件及び上位に位置するアプリケーションプログラムとのインタフェース条件を規定する。ただし、インタフェース条件として、呼出形式、引数、機能、実行条件、アクセス許可モード、リターン値等を規定することとする。

デバイスドライバ関数とのインタフェース条件は、CAM　Version1.0シリーズとは異なり、

CAM Version2.0が16社仕様準拠ICカード、JIS準拠ICカード等のデバイスドライバ関数を区別することなく対応するために必要不可欠であり、CAM Version2.0では、様々なデバイスドライバ関数とCAM Version2.0との入出力を一元的に調整する関数（以下、デバイスドライバ共通インタフェース関数という）を規定することにより実現を図る。

一方、アプリケーションプログラムとのインタフェース条件は、CAM Version1.0シリーズに倣い、CAM Version2.0を構成する関数として最上位に位置するアプリケーション・インタフェース関数を規定することにより実現を図る。



図」　CAM Version2.0の関数構成

7.4　内容アクセスマネージャを効果的に利用するための環境整備

将来の高度情報通信社会の到来に向けて、行政・保健・医療・福祉等の公共的な業種・分野を所轄する行政機関は、国民への情報提供の高度化、諸手続きの迅速化・簡略化等を始めとした公共分野サービスの質的向上をより一層推進するために、様々な公共的な業種・分野を対象に外部端子付ICカード、非接触ICカード等の高機能なカードメディアを導入していくと予想される。従来からの事務処理・手続き等を踏襲するならば、行政機関は各機関毎に利用者である国民にカードメディアを配布していくことになるが、この場合に国民は各機関毎に発行された複数枚のカードメディアを目的・用途毎に選択して使い分けなければならず、使い勝手の面で非常に煩雑になり、延いては高機能なカードメディアを導入する一つの重要な意義が失われる

可能性がある。国民に対する配慮を最優先する観点に立つならば、行政機関は各種の公共分野サービスに用いるカードメディアのあり方・位置づけ等について十分に議論していくことが極めて重要であると考えられる。

従って、ここでは、カード所有者である国民が公共分野に限らず複数の様々なアプリケーションサービスを一枚のカードメディアによって広域的にかつ共通的に享受できる利用環境を健全にかつ円滑に実現していくために必要不可欠となるであろう基本方針、作業手順、さらに整備項目等について、以下の通り纏める。

7.4.1 前提条件

(1) 対象カードメディア

ここでは、現在、既に実用化されているPETカード、磁気ストライプカード、外部端子付ICカード等を始めとした様々なカードメディアのうち、1チップ化されたCPU及び記憶メモリを搭載するカードメディアである外部端子付ICカード及び外部端子を持たない非接触型ICカード（以下、総称してCPU付ICカードという）を対象とする。

(2) 対象利用用途

CPU付ICカードは、その機能的な特長からセキュリティ機能を持つデータキャリアとしての利用用途とカード所有者及びアプリケーション端末に対する認証手段としての利用用途に大別され、アプリケーションサービス提供者はその導入目的や利用形態等により何れかの利用用途を選択することが一般的である。

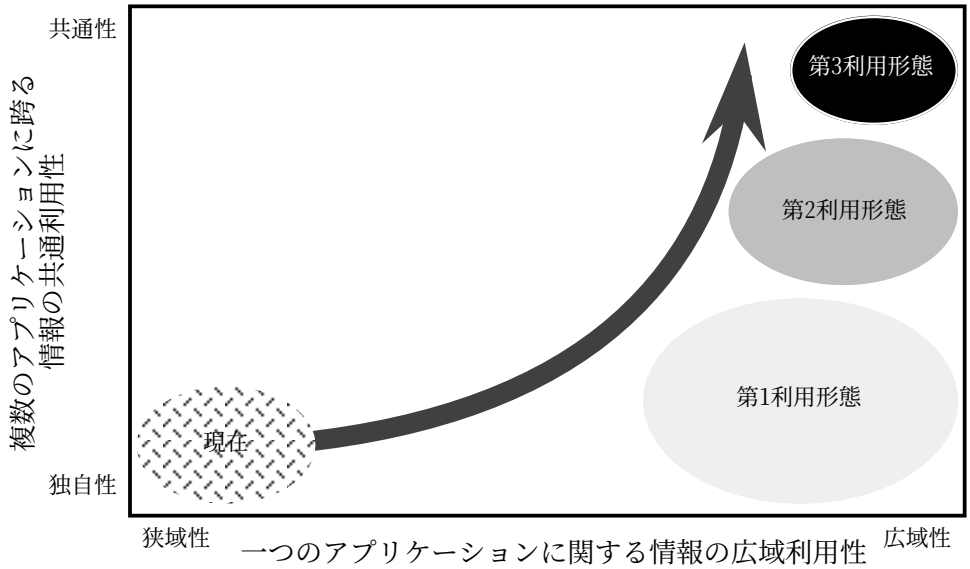
従来より「ICカード等多目的利用研究会」にて審議を進めてきたCPU付ICカードに関する基盤技術整備、並びに平成7年10月より施行されているJIS　X6306等はいづれも前者としての利用用途を前提としていること等を勘案して、ここでは、セキュリティ機能を持つデータキャリアとしての利用用途を対象とする。

7.4.2 基本方針

以上に示した目的、前提条件等を踏まえ、「1つのアプリケーションに関する情報を広域的に利用する尺度」と「複数のアプリケーションにまたがって情報を共通的に利用する尺度」という相互に独立な2つの尺度を設定することによって、CAMを効果的に利用するために、延いては多目的利用ICカードシステムを実用化していくために、多目的利用ICカードシステムを利用する形態を3つに分割し、それぞれの利用形態毎に整備すべき利用環境等を明確にすることを基本方針とする。

(1) 整備すべき利用環境

多目的利用ICカードシステムを利用する形態は、「1つのアプリケーションに関する情報を広域的に利用する尺度」と「複数のアプリケーションにまたがって情報を共通的に利用する尺度」に基づいて、図、に示す通り「第1利用形態」「第2利用形態」「第3利用形態」に3分割することが可能である。しかしながら、3つの利用形態は相互に独立な関係にあるのではなく、「第1利用形態」から「第2利用形態」へ、「第2利用形態」から「第3利用形態」へと順次移行していくことによって、「第2利用形態」さらには「第3利用形態」を実現することが可能である。



図、 基本方針

以下に、各利用形態において整備すべき利用環境を示す。

「第1利用形態」では、1つのアプリケーションにおいて提供される情報を広域的に利用するための環境整備に着手する。

また、「第2利用形態」では、「第1利用形態」により整備された環境下において、1つのアプリケーションに関する情報の一部分を他のアプリケーションに公開することにより複数のアプリケーションにまたがって情報を共通的に利用するための環境整備に着手する。

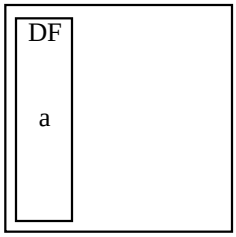
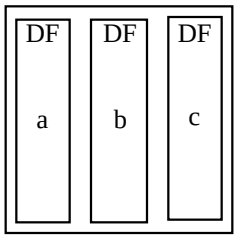
さらに、「第3利用形態」では、「第2利用形態」をより一層推進することにより、複数のアプリケーションにまたがって情報を共通的に利用するための環境整備を目指す。

(2) 各利用形態における課題・解決方針

カード発行者をA，B，Cまたそれぞれにより提供されるアプリケーションサービスをa，b，ｃ とし、またカード発行者はカードに記録されているデータの所有者と同一であると

仮定するならば、従来からの一般事例と第1利用形態以降それぞれにおける概念、課題、解決方針を以下の通り整理することができる。

表。 各利用形態における概念・課題・解決方針例

利用形態	概念	課題	解決方針
従来	カード発行者　A サービス提供者 A 	■メーカ各社により製造された同一仕様のCPU付ICカードを相互利用する。 ■S型、JIS等の世代が異なるCPU付ICカードを相互利用する。	■JISに準拠したCPU付ICカードと内容アクセスマネージャにより解決する。 ■内容アクセスマネージャにより解決する。 等
第1利用形態	カード発行者　　A サービス提供者 A,B,C 	■アプリケーションサービスまたはアプリケーションサービス提供者を識別する。 ■アプリケーションサービスを広域的に共通利用する。	■ICカードアプリケーション識別子の付番制度を制定する。 ■アプリケーションサービス毎にCAMを利用するための環境を整備する。
第2利用形態	第2利用形態では、第1利用形態のファイル構造の下に、1つのアプリケーション情報の一部を他のアプリケーションに公開する。	■アプリケーション提供者間において公開すべき情報を標準化する。 ■アプリケーション提供者間において公開した情報に対する読み込み処理について運用・管理の仕組みを明確にする。	■カード発行者、アプリケーション提供者、さらにカード利用者が持つべき権限、責任等を明確にする。 ■カード本体及びアプリケーションに係わる情報の所有権の所在を明確にする。 ■アプリケーションサービス提供者間における情報の相互利用をカード所有者に提示する第三機関を設置する。 等
第3利用形態	第3利用形態では、第1利用形態のファイル構造の下に、複数のアプリケーションにまたがって情報を相互に運用・管理する。	■アプリケーション提供者間においてそれぞれが運用・管理すべき情報を規定する。 ■アプリケーション提供者間において相互に行われる読み込み・書き込み処理について運用・管理の仕組みを明確にする。	

各利用形態における課題、解決方針等に鑑みて、カード所有者が一枚のCPU付ICカードによって複数の様々なアプリケーションサービスを広域的にかつ共通的に享受できる利用環境を実現する足掛かりとなる「第1利用形態」における課題、解決方針について検討を行った結果を以下に示す。

7.4.3　ICカードアプリケーション識別子の付番制度について

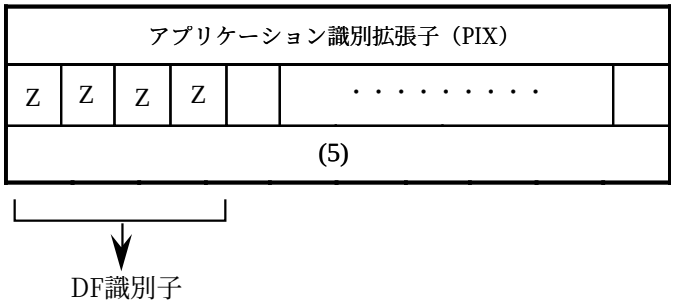
現在、ISO/IEC　7816-5に従って（社）日本事務機械工業会が主体となり審議が進められているICカードアプリケーション識別子の付番制度案に基づくならば、内容アクセスマネージャの利用を想定したICカードアプリケーション識別子の付番制度に関する考え方は以下の通り整理することができる。

(1) 目的

ICカードアプリケーション識別子にアプリケーションサービスの業種・業務を識別するための情報を記述することによって、アプリケーションサービスの広域的な利用を図る。とりわけ、内容アクセスマネージャを導入する場合には、上記の効果・効能が最大限に発揮される。

(2) 登録様式案

アプリケーション提供者識別子に含まれるアプリケーション識別拡張子（PIX）に対象となるDFの内部構成や定義主体等を明示するための情報として「DF識別子」を図･ に示すフォーマットで登録する。



図･　DF識別子の登録フォーマット

(3) DF識別子

アプリケーション識別拡張子は「ICカードアプリケーション識別子の申請・登録（付番）に関する国内規則」において業種・分野及び機関・団体等で統一化された任意の識別コードとして規定されている。

従って、対象となるアプリケーションサービスを利用する業種・分野を管掌、調整する機関・団体毎に標準化される必要がある。

(4) 留意事項

DF識別子を運用・管理する実施体制、仕組み等を明確にし、その普及啓蒙活動を推進していくことが重要である。

7.4.4 内容アクセスマネージャを対象とした利用環境の整備

CAMを導入することにより、カード事業実施・推進主体が国際または国内的な規模で共通利用可能であるCPU付ICカードシステムを構築する、または複数のアプリケーションサービスを相乗りできるCPU付ICカードシステムを構築するためには、カード所有者が持つ一般のICカードであるデータカードに関して以下に示す要素を順次整備していくことが必要不可欠である。[注1]

表「 データカードについて整備すべき要素

ファイル構成	整備すべき要素
DF（Dedicated File）	DF名称（ICカードアプリケーション識別子）
	業務提供者・端末キー
EF（Elementary File）	EF識別子
	業務提供者・端末キー
タグ番号	タグ番号に対応した標準項目名
データ実体	データ実体の記述形式

また、CAMでは、上記のデータカードの他に、それに対する読み込み・書き込み等の様々なオペレーションを制御するためのICカードとしてオペレーションカードを導入することから、データカードと併せて以下に示す要素を順次整備しなければならない。

表」 オペレーションカードについて整備すべき要素

ファイル構成	整備すべき要素
DF（Dedicated File）	DF名称
	業務提供者・端末キー
EF（Elementary File）	EF識別子
	業務提供者・端末キー
データ実体	データ実体に対するアクセス権限

[注1] CAMを用いず、国際・国内的な規模での共通利用を目指すアプリケーションサービスを対象としたCPU付ICカードシステムを構築するためには、これらの項目に加えて、導入するCPU付ICカード及びリーダ・ライタ等に関する機能・仕様を統一化する等の基盤技術を整備する必要がある。

7.5　今後の課題及び総括

ここでは、本年度の調査研究内容を踏まえ、将来、外部端子付ICカードや非接触型ICカード等のCPU付ICカードを社会的なインフラストラクチャとして位置づけていくために、今後解決すべき技術的な課題を内容アクセスマネージャの機能向上に関する課題とCPU付ICカードの機能向上に関する課題に分けて整理する。

7.5.1　内容アクセスマネージャの機能向上に関する課題

(1) JIS X6306に規定されたレコード記録形式への対応

将来的なICカード内のメモリの有効利用、及びICカードに記録されているデータの検索等の簡易化を図ることを目的として、CAM Version2.0において規定しているレコード記録形式をJIS X6306に規定されている簡易符号化TLVデータ形式に統合可能にすることが望ましい。

ただし、CAMにおいて利用しているタグの拡張機能がJIS X6306に規定されていないため、ISO/IEC 7816-4に規定されている簡易符号化TLVデータ形式におけるタグの拡張機能との整合性を明確にする必要がある。

(2) ISO/IEC 7816-4に規定された透過ファイルへの対応

CAM Version2.0では、JIS X6306に規定されたファイル構造に従い、固定長順編成ファイル、可変長順編成ファイルを対象としているが、将来的なICカード内のメモリの有効利用を図るために、JIS X6306に規定されているファイル構造以外にISO/IEC 7816-4に規定されている透過ファイルへも対応可能にする必要がある。

(3) タグの拡張への対応

現在、CAM Version2.0は、1バイトでタグ番号を記述する場合に先頭ビットをタグ番号を拡張するための識別子として利用しているため、0x8000以上のタグ番号を利用不可とする機能仕様になっている。

従って、今後0x8000以上のタグ番号を利用可能とすると共に、ISO/IEC 7816-4に規定されているレコード識別子の拡張方法と整合性のあるタグ番号の拡張方法を確立することが必要である。

(4) オプションコマンドの拡張への対応

現在、CAM Version2.0はJIS X6306に規定されている基本コマンドのみを対象とした機能・関数仕様となっているが、今後より一層CPU付ICカードの柔軟性や拡張性を向上していくために、基本コマンド以外に存在する各種オプションコマンドへの対応を図っていくことが極めて重要である。

(5) CPU付ICカードの識別への対応

現時点では、カード製造メーカ各社により製造されるCPU付ICカードを登録する制度が整備されるか否か不明であるが、このような制度が整備されていくことを念頭に置きつつ、カード製造メーカ各社のCPU付ICカードを識別可能とする手段を講じていくことが肝要である。

7.5.2　CPU付ICカードの機能向上に関する課題

(1) システムコマンドの公開に向けた対応

将来的に「4.内容アクセスマネージャを効果的に利用するための環境整備」に示す「第1利用形態」から「第2利用形態」さらには「第3利用形態」へと一枚のCPU付ICカードを取り巻く利用形態が移行していく可能性が十分にあることを踏まえるならば、従来から現在までに顕在化してきた様々な弊害を順次解決していくことが極めて重要であると考えられる。

現在、その重大な一つの弊害として、カード製造メーカ各社がDF・EF等のファイル創成やそれらに関する鍵の設定等を制御するシステムコマンドを非公開としていることが指摘されていることから、システムコマンドに対して鍵を付与する等のシステムコマンドを公開していくための対応策を明確にし、実装していくことが重要である。

(2) 本人確認等の相互認証技術の確立

コンピュータシステムのダウンサイジング化、ネットワーク化、オープンシステム化等の電子情報技術に関する技術革新により、従来では想定され得なかった、商取引・決済等の経済活動を様々なコンピュータ・ネットワークを複合的に用いて実施するエレクトロニック・コマースが可能になってきている。

企業と個人、企業と企業、個人と個人等の様々な取引形態、かつ様々な利用頻度において取り行われるエレクトロニック・コマースを実現するためには、オープン・クローズを問わずコンピュータ・ネットワーク上で本人確認や相互認証を確立するためのセキュリティ技術を整備することが必要不可欠であり、マイクロプロセッサを搭載したメモリであるCPU付ICカードは、その機能的な特徴から一つの実用的ツールとして脚光を浴

びており、積極的に導入されていくことが予想される。

現在では、CPU付ICカードにより本人確認を確立する方策として、「電子認証局や電子公証局を介して認証する方法」と「CPU付ICカード・アプリケーション端末・情報システム間において相互認証する方法」が挙げられているが、前者に関しては、行政機関等における電子認証局等の運用・管理に関する実施体制及び仕組み等を事前に明確にすることが必要であり、実用までに相当の期間を要すると予想される。

以上を踏まえ、後者に関する本人確認の方策を可及的速やかに実現していくことが必要であると考えられる。

7.6 総括

平成7年度の調査研究では、平成7年10月よりJIS X6306が施行されたことを踏まえ、従来からの暫定的な国内標準として位置づけられた16社仕様準拠ICカードだけでなく、新たに制定されたJIS X6306に準拠したICカードに内容アクセスマネージャを対応させるべく、既に公開されているCAM Version1.1に関する機能仕様書及び標準ソフトウェアのレビュー及びバージョンアップを図り、CAM Version2.0に関する機能仕様書及び標準ソフトウェアを取り纏めた。

また、それと共に、将来の高度情報通信社会の到来に備えて、カード所有者である国民が公共分野に限らず複数の様々なアプリケーションサービスを一枚のカードメディアにより広域的にかつ共通的に享受できる利用環境を健全にかつ円滑に実現していくために必要不可欠となるであろう基本方針、作業手順、整備項目を整理し、その端緒として位置づけられるICカードアプリケーション識別子の付番制度及び内容アクセスマネージャを効果的に利用するための環境整備について議論を行った。

今後は、平成4年度より継続している本調査研究の成果である内容アクセスマネージャが「CPU付カードメディアの普及促進への貢献を実践するための技術的基盤」の一つとしてより広く認知され、かつより積極的に導入・実用化されていくことによって、次世代ICカードへのより高次な要求機能の実現、延いては次世代ICカードの実用化に向けた周辺環境整備を促進するための一端を担っていくことを期待する処である。

【本編】

1.内容アクセスマネージャの概念

1.1　背景及び目的

近年、半導体製造技術の向上によるチップのコモディティ化と価格性能比の飛躍的上昇がコンピュータ利用層を拡大し、エンドユーザコンピューティングの潮流を強力に推進している。それに伴い、ソフトウェア開発では、従来の「ハードウェア依存型」から新しい「ハードウェア独立型」への技術的ニーズが高まり、異なる機種のコンピュータ間の接続を容易にするためのオープンシステム化がコンピュータ製造メーカ各社等によって進められつつある。

このようなソフトウェアによるオープンシステム化の新しい技術的思想は、コンピュータシステムにより発生したデータを異なるコンピュータシステム間において相互にかつ共通的に利用可能とするための「オープンデータシステム化」への技術革新を促しているといえる。

一般に、CPU付メモリを小型のコンピュータシステムとみなすならば、データを加工するホストコンピュータとデータを格納するメディア間の入出力処理は、LAN・WAN等の様々なコンピュータネットワーク通信と同じく処理することが可能となる。従って、CPU付メモリの一形態である外部接点付ICカードとそれを用いたコンピュータシステム（以下、ICカードシステムという）に関しても上記のオープンデータシステムの方向性を踏襲して設計・開発していくことが望ましいと考えられる。

しかしながら、現在国内の各種分野・業種で利用されているICカードシステムでは、カード内のデータがホストコンピュータ内のアプリケーションプログラム構成に大きく依存する、加えてカードメーカ各社が製造するICカードの詳細な機能仕様等がメーカ各社毎に異なっている等のコンピュータシステムにおける典型的な技術課題を抱えており、現に先進事例等においても、カード所有者、アプリケーションサービス提供者、さらにソフトウェア開発者等から以下に示す要求定義が提示されている状況にある。

＜要求定義1＞
ICカード内のデータがホストシステムを構成するハードウェアおよびソフトウェアに依存しない。
＜要求定義2＞
ICカード内の各ファイルに記録されるデータの実体（内容）を動的に変更可能である。
＜要求定義3＞
ICカード内の各ファイルに記録されるデータの総量を変更可能とする。
＜要求定義4＞
ファイル単位だけでなくデータ単位でのきめ細かなセキュリティ管理を可能とする。

今後、ICカードを多機能、さらには多目的に利用する等の多種多様な利用の可能性を拡げていくためには、ICカードシステムのオープン化、マルチベンダ化を図るべく、上記＜*要求定義1～4*＞の*機能*を実現することが極めて重要である。

そこで、現在既に国内市場に流通している16社仕様に準拠したICカードだけでなく、平成8年度を目途に市場への登場が予想されるJIS X6303, X6304, X6306に準拠したICカード（以下、JIS準拠ICカードという）を対象として、上記＜*要求定義1～4*＞の機能をホストコンピュータ上のソフトウェアにより実現する。このソフトウェアをその保有する機能から内容アクセスマネージャ Content Access Manager（略称:CAM　以下、CAMという）と名付ける。

1.2 位置づけ

ICカード上に様々な組み合わせのアプリケーションサービスに関するデータが記録されている場合、任意のホストコンピュータ上の任意のアプリケーションプログラムが様々な業務サービスのデータを汎用的に利用可能であるようなオープンデータシステムの環境を実現することが望ましい。

そのため、ICカードシステムにおいてCAMを図1.2.1の通り位置づけ、アプリケーションプログラムによるICカード内のデータへのアクセス形式として、データ実体が記録されている物理アドレスを指定するのではなく、データ実体を直接指定するアクセス形式を実現する。

1.2.1 物理アドレス指定方式

現在のICカードでは、図1.2.1(a)に示す通り、アプリケーションプログラムからICカードに対して物理アドレスを直接指定する方式でアクセスするため、アプリケーションプログラム側でICカードのファイル構造を既知である必要がある。また、ホストコンピュータとICカード間を固定長フォーマットにおける低水準データ操作のソフトウェア・インタフェースでつなぐため、アプリケーションプログラムへの機能分担が大きく、汎用性を出しにくい。

1.2.2 データ実体指定方式

これに対して次世代ICカードでは、図1.2.1(b)に示す通り、データの一部に含まれている記録内容を規定するタグを検索し、そのタグが記録されている物理アドレスの記録内容を読み出す方式でアクセスする。さらに、ホストコンピュータとICカード間を可変長フォーマットにおける高水準データ操作のソフトウェア・インタフェースでつなぐことにより、アプリケーションプログラムへの機能分担を小さくし、汎用性を大幅に向上する。

具体的な実現方法として、ホストコンピュータ内にオープンな高水準ライブラリとしてCAMを用意し、アプリケーションプログラムと連結することによって、見かけ上次世代ICカードを用いているかのような環境を実現する。

さらに、将来的な次世代ICカードにおいては、図1.2.1(c)に示す通り、CAMの高水準ライブラリ機能をICカード内に収容し、高水準インタプリタとして組み込むことが想定される。しかしながら、現実的にはセキュリティ機能等の関係から次世代ICカードに持たせるCAMの機能は当初限定されたものとし、ホストコンピュータ内に一部の機能が残存することになると考えられる。

以上のような位置づけの高水準ライブラリ（将来的には高水準インタプリタ）が、CAMである。

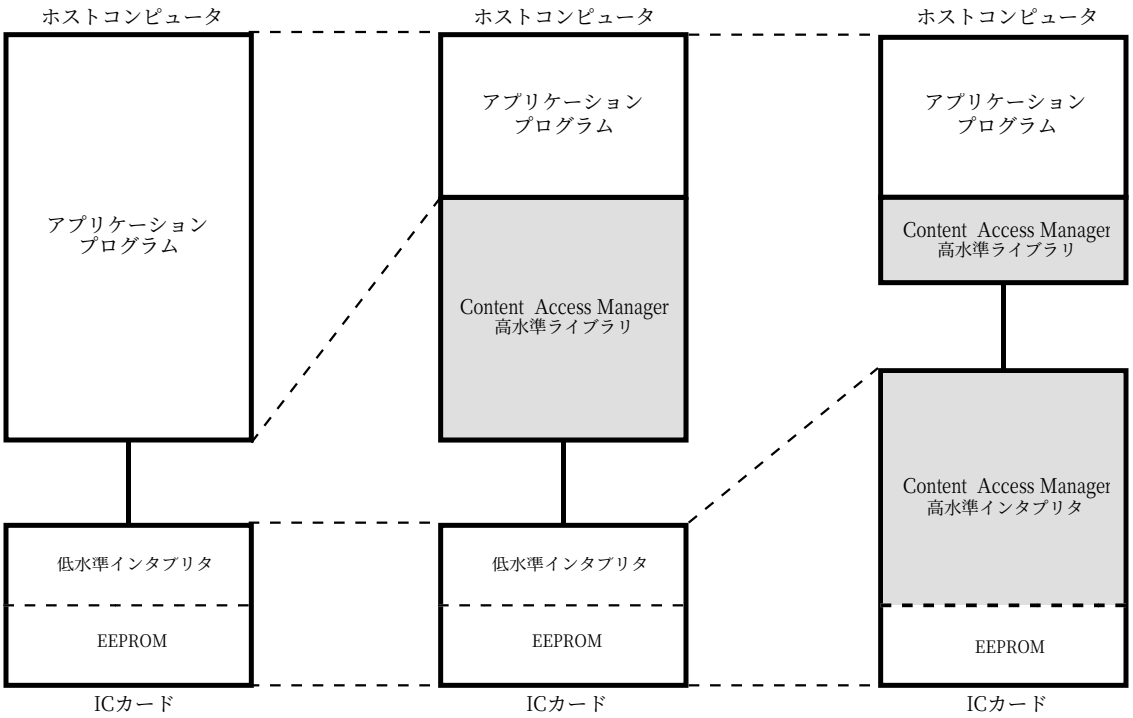


図1.2.1　CAMの位置づけ

1.3　システム構成

現在、ICカードシステムでは、ホストコンピュータ内のアプリケーションプログラムとICカード間のデータ通信制御をJIS X6304に準拠して行っている。JIS X6304では､ICカードリーダ・ライタとホストコンピュータを接続するためのRS-232Cに関する通信制御プロトコルに限定して規格を制定しており、RS-232Cを制御するためのコマンド等はホストコンピュータに搭載されたOSに依存する仕組みとなっている。

このようなデータ通信制御の仕組みを踏まえて、図1.3.1に示す通り、CAMをホストコンピュータ内のアプリケーションプログラムとRS-232C等のデバイスドライバ等を制御するOSの中間に位置するミドルウェアとして位置づける。

また、CAMは、ICカードとホストコンピュータを論理的に分離させると共に、現在メーカ各社により製造されているICカードや将来開発・利用される次世代ICカード等を相互運用可能にすることを目的としていることから、共通的に利用可能な汎用ミドルソフトウェアであることが必須条件である。従って、CAMでは、CAMとアプリケーションプログラム、及びCAMとデバイスドライバとのインタフェースの標準仕様をそれぞれ規定する。

以上のようにCAMをICカードシステムに導入することによって、アプリケーションプログラム及びデバイスドライバ等に依存することなく、ICカード内のデータに関して確実な互換性及び相互運用性を維持することが可能となる。

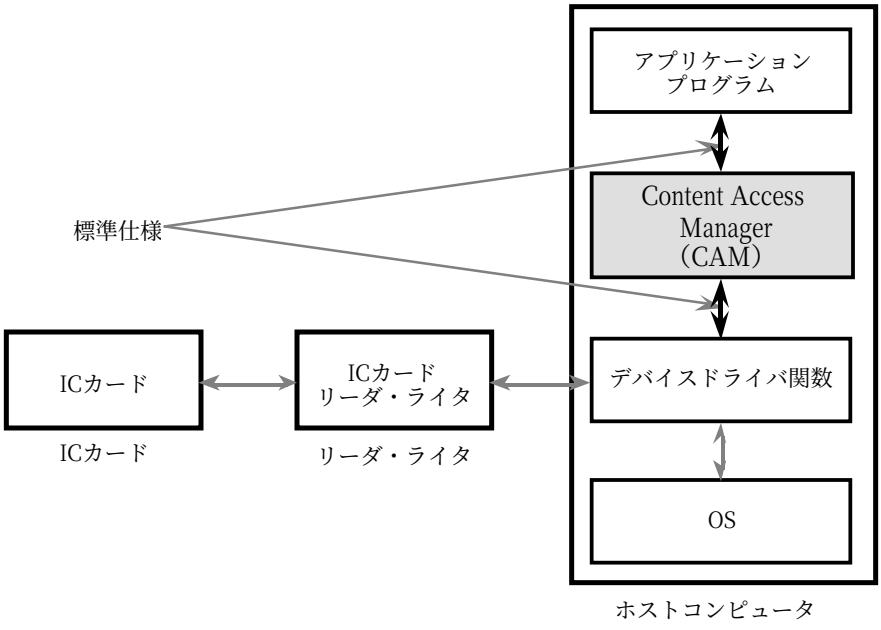


図1.3.1　CAMを用いたICカードシステム概念図

1.4　要求機能

1.4.1　ホストシステムとICカード間の論理的な分離機能

＜要求定義1＞
ICカード内のデータがホストシステムを構成するハードウェアおよびソフトウェアに依存しない。

*＜要求定義1＞*を実現するために、データ実体指定方式でアクセスすること、そしてホストコンピュータとICカード間を可変長フォーマットにおける高水準データ操作のソフトウェア・インタフェースで繋ぐことが必要条件であることを踏まえ、CAMはC言語等の高級言語で記述した高水準ライブラリとする。ただし、RS-232Cの通信制御を行うデバイスドライバ入出力関数に関しては、ホストコンピュータを構成するハードウェアやOS等の基本ソフトウェアに依存するため、メーカ各社が開発する必要がある。

1.4.2　データ動的変更機能

＜要求定義2＞
ICカード内の各ファイルに記録されるデータの実体（内容）を動的に変更可能である。

*＜要求定義2＞*を実現すると共に、データ実体が可変長であることを考慮してICカード内のメモリを有効利用するために、ファイル内のデータを図1.4.1に示すようなフォーマットに従って記述する。ただし、データ項目名（タグ）は、氏名や住所等のように自然言語を用いて記述することが望ましいが、ICカードの場合には記憶容量が少ないためコード化せざるを得ないと想定される。

データ項目名（タグ） 1バイト	データ長 1バイト	データ実体 Nバイト

図1.4.1　データフォーマットの一例

図1.4.1に示す例では、Nバイトから構成されるデータ実体を記述するために、データ実体を規定する識別子であるデータ項目名（タグ）とデータ実体の長さを示すデータ長をデータ実体の前にそれぞれバイナリで1バイト付与することによって、データ全体をN+2バイトで表現している。本例では、データ項目名（タグ）とデータ長を示すバイト数を1バイトに設定しているが、通常ではICカードに記録するデータ実体の総容量等から決定することが望ましい。

1.4.3　ファイルサイズ動的変更機能

＜要求定義3＞
ICカード内の各ファイルに記録されるデータの総量を変更可能とする。

*＜要求定義3＞*を実現する単純明解な方策として、ICカード内のファイルを登録・削除するための発行系コマンドによる対応策が考えられるが、発行系コマンドはICカードの最大の特長であるといえるセキュリティ機能の低下が危惧される等の事由により、ISO/IEC7816シリーズにおいて標準化されていないため、現時点における本方策の実現性は極めて低い。

従って、CAMでは、発行系コマンドによる対応ではなく、運用・管理が煩雑にならない現実的な方策として、図1.4.2に示すように、ファイル領域が不足する事態に対応するための予備的なファイル（以下、拡張ファイルという）を予めICカード内に作成しておき、対象ファイルに追記できないデータを拡張ファイルに書き込むことによって暫定的にファイルを拡張する。ただし、本方法では発行系コマンドを利用しないことを前提条件としているため、ファイルサイズはデータ削除等による各EFの予め割り付けられたサイズ以下には縮小できないが、拡張ファイル領域の分だけ変更が可能となる。

また、拡張ファイルには複数ファイルのデータが記録される可能性が十分にあることから、拡張ファイルに記録するデータは、対象データが属するEF番号を拡張ファイル用の識別子としてタグに付加した、図1.4.2に示すデータフォーマットで記録する。ただし、拡張ファイル用の識別子であるEF番号は、セキュリティを向上する観点からCAMが自動的に付番することが望ましい。

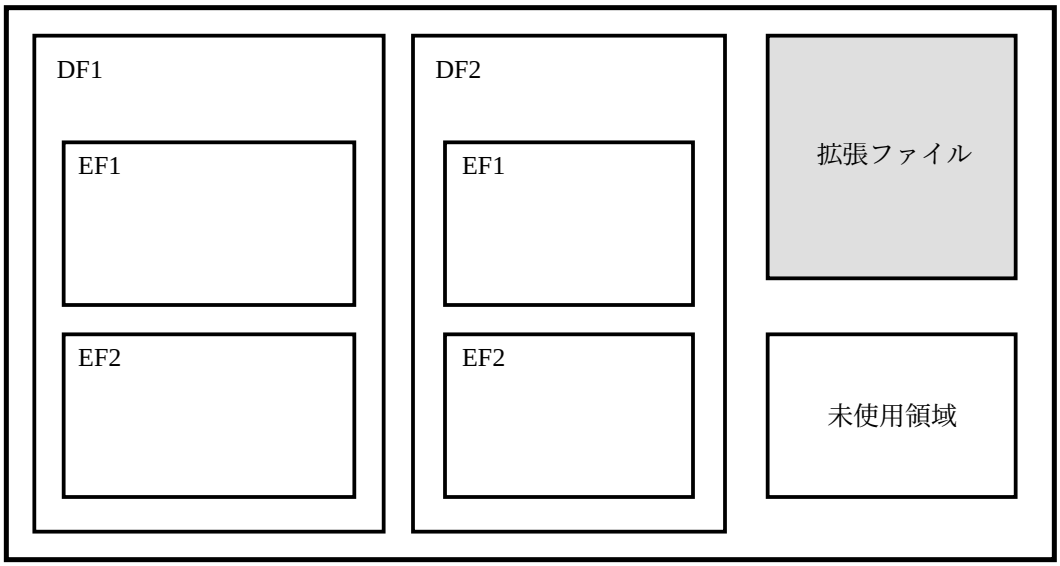


図1.4.2　CAMを導入したICカード内のファイル構造例

EF番号＋タグ 2バイト	データ長 1バイト	データ実体 Nバイト

図1.4.3　拡張ファイル内のデータ記述フォーマットの一例

1.4.4　セキュリティ機能

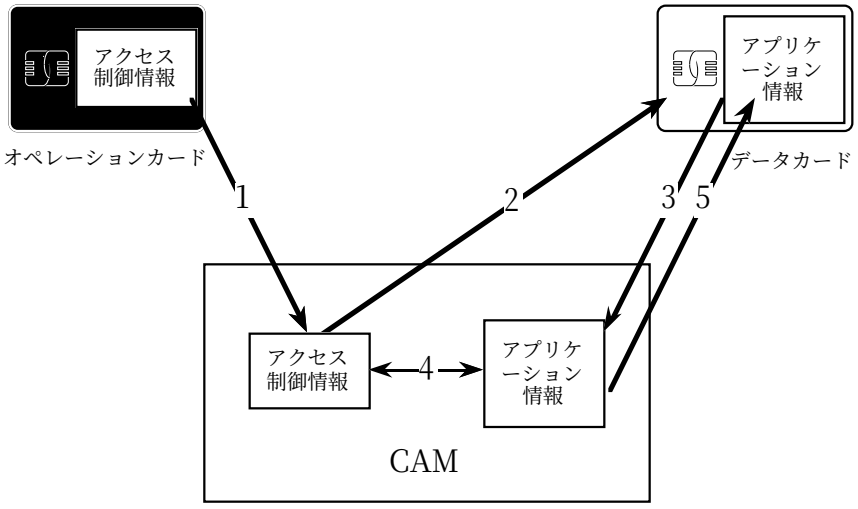
＜要求定義4＞

ファイル単位だけでなくデータ単位でのきめ細かなセキュリティ管理を可能とする。

＜要求定義4＞を実現するために、現在既に市場に流通しているICカードシステムのセキュリティ機能がCAMを導入することによって劣化することなく、様々な種類のICカードと共存可能であることを前提条件として、ICカードとCAMによりセキュリティ管理を機能分担する。すなわち、従来通りのDFやEF等のファイル単位に関するセキュリティ管理をICカードが行い、またEF内のデータ単位のセキュリティ管理をCAMが行う。

CAMでは、カード所有者が持つ一般のICカード（以下、データカードという）の他に、それに対する読み込み・書き込み等の様々なオペレーションを制御するためのICカード（以下、オペレーションカードという）を新たに導入することによって、ICカードとCAMによる機能分担を図る。

データカード、オペレーションカード、CAM間のセキュリティ管理に関する処理手順の概念は以下の通りである。



1　アクセス制御情報の受信
2　ICカード内の対象ファイルに対するアクセス制御情報照合
3　アプリケーション情報の受信
4　対象ファイルのデータ項目に対するアクセス制御情報照合
5　更新されたアプリケーション情報の送信

図1.4.4　CAMによるセキュリティ管理の概念

具体的な手順として、まずオペレーションカードの所有者が正当であるという確認が行われた後に、オペレーションカードからCAMに対してアクセス制御情報を送信する。

次に、CAMは、オペレーションカードから送信されたアクセス制御情報をデータカード内の対象ファイルに対して照合する。照合後アクセスが許可された場合のみ、データカード内の対象ファイルからCAMに対してアプリケーション情報を送信する。

CAMは、オペレーションカードからのアクセス制御情報とデータカードからのアプリケーション情報の照合を行い、その結果アクセスが許可され、かつデータに対して書換等の処理が行われた場合に、更新されたデータをCAMからデータカードへ送信する。

2.内容アクセスマネージャVersion2.0の機能仕様

2.1　はじめに

内容アクセスマネージャは、平成4年度に財団法人ニューメディア開発協会内に発足した「ICカード等多目的利用研究会」において、現在既に国内に流通しているICカードシステムだけでなく、ISO/IEC国際標準規格に準拠した次世代のICカードシステムに要求される機能の一部を実現することを目的として提案された新しい概念である。

平成5年度の調査研究では、その概念に基づいて、CAMが具備すべき機能・インタフェース条件等を規定した仕様書（以下、機能仕様書という）Version1.0として「平成5年度　内容アクセスマネージャの具体化に関する調査研究報告書」をとりまとめ、CAM標準ソフトウェアVersion1.0を開発した。ここでいう標準ソフトウェアとは、機能仕様書に記述されていない詳細を補足するためのフリーソフトウェアとして、またメーカ各社により製品化されたCAM間の互換性を確認するためのリファレンスとして利用することを目的とした開発した標準的なソフトウェアを意味する。

引き続き、平成6年度では、CAM Version1.0が様々な業種・分野に適用され始めたことにより顕在化してきた機能追加等のニーズに対応すべく、CAM　Version1.0の機能仕様・インタフェース条件等を改訂し、機能仕様書Version1.1として「平成6年度　内容アクセスマネージャの機能に関する調査研究報告書」をとりまとめると共に、CAM標準ソフトウェアVersion1.1を開発した。

しかしながら、CAM Version1.0, 1.1は、ISO/IECにおける国際標準化作業の進捗等の諸事情に鑑みて､国内における暫定標準として位置づけられた16社仕様準拠ICカードを対象として研究・開発されたソフトウェアであるため、工業技術院によってJISが制定されたあかつきには、JIS X6303,X6304, X6306に準拠したICカード（以下、JIS準拠ICカードという）への対応を図ることが肝要であると指摘されていた。

平成7年度になり、工業技術院によってJIS　X6306が制定されたことを踏まえ、本年度では、従来からの16社仕様準拠ICカードだけでなく、新たなJIS準拠ICカードへの対応を図るべく、既に公開されているCAM　Version1.1に関する機能仕様書及び標準ソフトウェアに関するバージョンアップを行い、CAM機能仕様書　Version2.0をまとめ、標準ソフトウェア　Version2.0を開発する。

以下では、CAM Version2.0において新たに対象に組み込まれたJIS準拠ICカードに関する機能仕様について詳述する。また、巻末の【参考資料】として、CAM標準ソフトウェア　Version2.0のプログラムドキュメントを添付する。

2.2　データ動的変更機能

2.2.1　データフォーマット

JIS　X6306では、一般のデータを記録するEF（以下、基礎ファイルという）のファイル構造として、透過構造ではなくレコード構造からなる固定長順編成ファイル、可変長順編成ファイル、または固定長循環順編成ファイルいずれかに対応することが規定されている。

また、基礎ファイルは、1つのレコードを図2.2.1に示すタグ・長さ・データという3つの連続したフィールドから構成する簡易符号化TLVデータ形式で記述することが規定されており、基礎ファイル内のレコードは、レコード番号形式とレコード識別子形式によって参照することが規定されている。レコード番号形式による参照方式とは、基礎ファイル内のレコードを一義的に指定するために各レコードに割り振られる連続した番号によりレコードを参照する方式である。一方、レコード識別子形式による参照方式とは、1つの基礎ファイル内に同一の番号が複数存在することを許容する番号によりレコードを参照する方式である。

タグ TAG	長さ LENGTH	データ VALUE

タグ：1バイト（「0x01」～「0xfe」）
長さ：1バイト（「0x00」～「0xfe」）
データ： データ実体
図2.2.1　簡易符号化TLVデータフォーマット

以上に基づき、CAMは、図2.2.2に示す通り、簡易符号化TLVデータ形式に準拠して、JIS X6306に規定されたレコードフォーマットのデータVALUEにデータを記録する。ただし、その場合における簡易符号化TLVデータ形式のタグは「0x0f」に設定する。

JIS X6306に規定されたレコードフォーマット

タグ 0x0f	長さ LENGTH	データ VALUE

CAMが対象とするデータフォーマット

タグ：1～2バイト
データ長：2～5バイト
データ実体：可変長
図2.2.2　JIS X6306とCAMの関係

また、CAMがJIS　X6306に規定されたレコードフォーマットのデータVALUEにデータを記録するフォーマットは、図2.2.3に示す通り、1つのデータに一組のタグ、データ長、データ実体が記録される場合、1つのデータに複数組のタグ、データ長、データ実体が記録される場合、さらには複数のデータに一組のタグ、データ長、データ実体が跨る場合等、様々な形態が想定される。

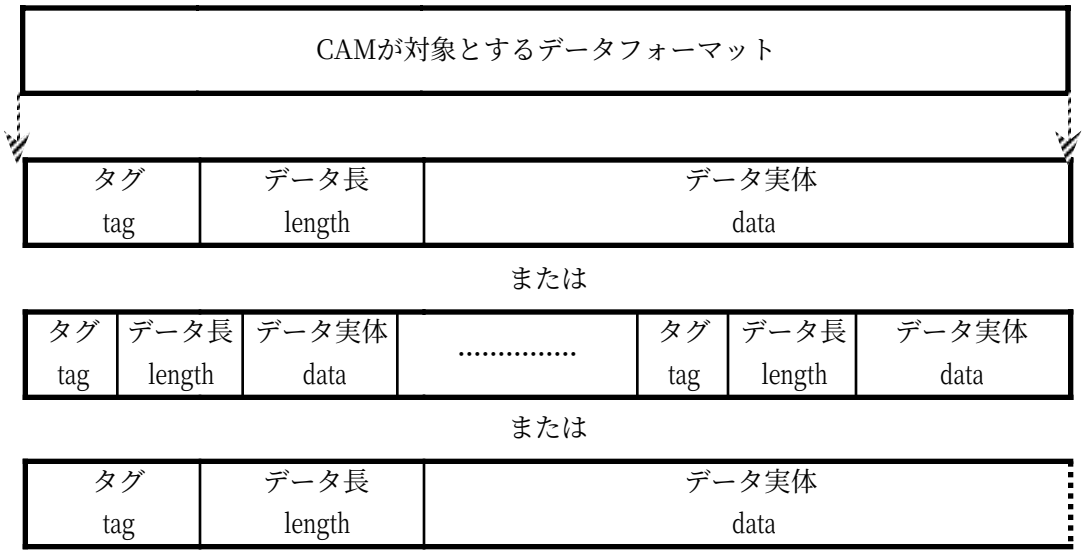


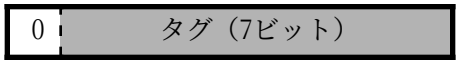
図2.2.3　CAMが対象とするデータフォーマット

2.2.2　タグ tag

タグは、1バイトまたは2バイトのバイナリで記述する。

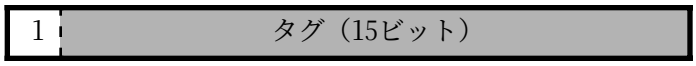
(1)タグが1バイトの場合

上位1ビットを「0」に設定し、下位7ビットを用いて表現する。



(2)タグが2バイトの場合

上位1ビットを「1」に設定し、下位15ビットを用いて表現する。



2.2.3　データ長 length

データ長を示す領域は、データ実体の長さによって1～5バイトのバイナリで記述する。デ

ータ長領域のフォーマットは図2.2.3に示す通りである。図2.2.3において定義されているデータ長領域の長さを決定するフラグSL1,SL2は、表2.2.1に示す通りデータ長領域を設定するためのフラグである。また、それと共にデータ個数判定フラグNFは、1つのタグに対応するデータ実体に唯一のデータを記録する場合と複数個のデータを1つのグループとして記録する場合を判別するためのフラグであり、1つのデータ実体に唯一のデータを記録する場合には「0」、1つのデータ実体に複数個のデータを記録する場合には「1」を設定する。

1：データ長領域の利用形式を決定するフラグ
SL1・SL2：データ長領域の長さを示すフラグ
NF：データ個数判定フラグ

図2.2.3　データ長領域フォーマット

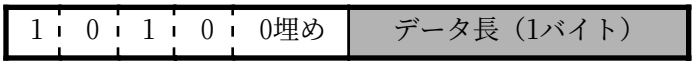
表2.2.1　データ長領域の長さを示すフラグ

SL1	SL2	データ長	備考
0	0	データ長≦4ビット	データ実体領域が次の4ビット
0	1	5ビット≦データ長＜256バイト	データ長領域が　次の1バイト
1	0	256バイト≦データ長＜65536バイト	データ長領域が　次の2バイト
1	1	データ長≧65536バイト	データ長領域が　次の4バイト

(1) 1つのデータ実体に唯一のデータを記録する場合

まず、データ長が4ビット以下であるデータは、データ長領域が占有する容量を節約するために、1バイトのデータ長領域の下位4ビットにデータ実体を記録する。

データ長が5ビット以上256バイト未満であるデータは、2バイトのデータ長領域の下位1バイトにデータ長を記録する。



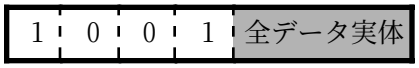
データ長が256バイト以上65536バイト未満であるデータは、3バイトのデータ長領域の下位2バイトにデータ長を記録する。

データ長が65536バイト以上であるデータは、5バイトのデータ長領域の下位4バイトにデータ長を記録する。

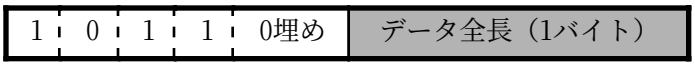
ただし、CAM標準ソフトウェアVersion2.0では、データ長が65536バイト以上であるデータに対応しない。

(2) 1つのデータ実体に複数個のデータを記録する場合

複数個のデータの全長が4ビット以下であるデータは、データ長領域が占有する容量を節約するために、1バイトのデータ長領域の下位4ビットに全データ実体を記録する。



複数個のデータの全長が5ビット以上256バイト未満であるデータは、2バイトのデータ長領域の下位1バイトにデータ全長を記録する。



複数個のデータの全長が256バイト以上65536バイト未満であるデータは、3バイトのデータ長領域の下位2バイトにデータ全長を記録する。

複数個のデータの全長が65536バイト以上であるデータは、5バイトのデータ長領域の下位4バイトにデータ全長を記録する。

ただし、CAM標準ソフトウェアVersion2.0では、データ全長が65536バイト以上であるデータに対応しない。

2.2.4　データ実体 data

(1) 1つのデータ実体に唯一のデータを記録する場合

データ実体を示す領域は、データ長領域に続くバイトから可変長に記録される。

(2) 1つのデータ実体に複数個のデータを記録する場合

1つのデータ実体に複数個のデータを記録する場合では、個別のデータ長及びデータ実体を図2.2.4に示すフォーマットに従って可変長に記録する。

図2.2.4　N個のデータを記録する場合のフォーマット

2.2.5　特殊なファイルにおけるデータフォーマット

通常データを記録する基礎ファイル内のデータフォーマットは既述した通りであるが、CAMでは、その他に基礎ファイルが対象とするCAMバージョン番号とその領域サイズを記述するデータ、拡張ファイル内のデータ、さらに拡張ファイルへのデータ拡張する基礎ファイル内の最終データに関するデータフォーマットを以下の通り定義する。

(1) 基礎ファイルが対象とするCAMバージョン番号とその領域サイズ

「基礎ファイルが対象とするCAMバージョン番号とその領域サイズ」は拡張ファイルを含めた全ての基礎ファイルの先頭データに記録され、データフォーマットは以下の通りである。

!タグ

タグを1バイトのバイナリで記録する。CAM標準ソフトウェアVersion2.0では「0x00」に設定する。

"データ長

データ長を1バイトのバイナリで記録する。

#データ実体

まず、バージョン番号を2バイトのバイナリで記録する。CAM標準ソフトウェアVersion2.0では、「0x0200」に設定する。

次に、5バイト目以降に対象である基礎ファイルにCAMにより記録されるレコード長の総和をEF領域サイズとして1～3バイトのバイナリで記録する。

タグ	データ長	バージョン番号	EF領域サイズ
1バイト	1バイト	2バイト	3バイト

図2.2.5　CAMバージョン番号と領域サイズに関するデータフォーマット

(2)拡張ファイル

また、拡張ファイルに記録されるデータは以下に示すデータフォーマットに従う。

! EF識別コード

拡張ファイルは様々なEFから記録されることが想定されることから、各データが属する基礎ファイルを識別するための識別コードとして「EF識別コード」をデータの先頭に必ず付ける。また、EF識別コードは対象ファイルが初めて拡張ファイルにデータを記録する時にCAMが付与する。

ただし、CAM標準ソフトウェアVersion2.0では、「0x01」から順次採番することとする。

EF識別コードは1バイトまたは2バイトのバイナリとする。上位1ビットを「0」に設定した場合に下位7ビットを、また上位1ビットを「1」に設定した場合に下位15ビットを用いて表現する。

"タグ

基礎ファイルにおけるデータフォーマットに準拠する。

#データ長

基礎ファイルにおけるデータフォーマットに準拠する。

$データ実体

基礎ファイルにおけるデータフォーマットに準拠する。

EF識別コード 1～2バイト	タグ	データ長	データ実体

図2.2.6　拡張ファイルにおけるデータフォーマット

(3)ファイル拡張する基礎ファイルに記録される最終データ

ファイル拡張する基礎ファイルに記録される最終データのデータフォーマットは以下の通りである。

!タグ

タグを1バイトのバイナリで記述する。

CAM標準ソフトウェアVersion2.0では、「0x01」に設定する。

"データ長

データ長を1バイトのバイナリで記述する。

#データ実体

データ実体にEF識別コードを記録する。本データが対象ファイルの最終データとして記録されていることによって、拡張ファイルを使用していることを示す。

項目タグ 1バイト	データ長 1バイト	EF識別コード 1～2バイト

図2.2.7　ファイル拡張におけるデータフォーマット

2.3　ファイルサイズ動的変更機能

2.3.1　カードフォーマットに関する前提条件

CAMが対象とするICカードのカードフォーマットに関する前提条件は以下の通りである。

(1)ファイル管理単位

1枚のICカードに複数のアプリケーションサービスに係わるデータを記録する事態等を想定するならば、各アプリケーションサービスに関するデータは、JIS　X6306に規定された論理ファイルの階層構成及びセキュリティ管理等の点からDF単位に記録することが常道であると考えられることから、DFをCAMによるファイル管理単位として定義する。

(2)基礎ファイル構造

CAM　Version2.0は、JIS　X6306に規定されている基礎ファイル構造のうち、固定長順編成ファイルと可変長準編成ファイルを対象とする。ただし、CAM標準ソフトウェアVersion2.0は、固定長順編成ファイルのみを対象とする。

(3)ファイル種類

CAMでは、対象とするファイル種類として、既に述べた基礎ファイル、拡張ファイルの他に、DF情報ファイルを定義する。

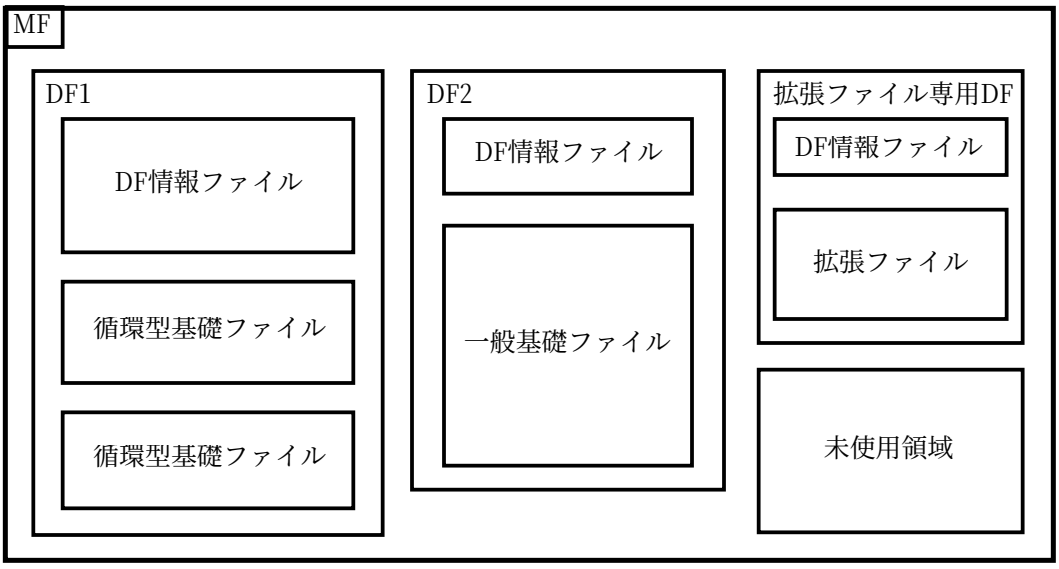


図2.3.1　カードフォーマットの概念図

基礎ファイルは、対象DF内に一つの論理的なEF-IDが唯一に付与される基礎ファイル（以

下、一般基礎ファイルという）と対象DF内に一つの論理的なEF-IDが複数個に付与される基礎ファイル（以下、循環型基礎ファイルという）から構成され、目的・用途により選択可能である。これらの詳細については後述する。

また、DF情報ファイルはCAMが対象とするDF内に存在する全ての基礎ファイル、拡張ファイルを一元管理するために設定するファイルであり、対象DF内に唯一設定される。

CAMが対象とするICカードのフォーマットの概念を図2.3.1に示す。

(4)EF-IDの取り扱い

CAM Version2.0において取り扱うEF-IDは、JIS X6306に規定される1バイトのShort EF-ID「0x 01」～「0x1e」を意味する。

(5)ファイル管理

CAMでは、一般のDFまたは拡張ファイル専用DF内のDF情報ファイルが同一DF内の基礎ファイルまたは拡張ファイルを管理する。また、DF内の基礎ファイルが拡張ファイルへデータ拡張した場合においては、拡張元である基礎ファイルが属するDF内のDF情報ファイルに拡張ファイルの所在を記録する。

2.3.2 DF情報ファイル

(1)EF-ID

DF情報ファイルに付与するShort EF-IDを「0x1e」に設定する。

(2)ファイル構成

DF情報ファイルは、CAMが対象とするデータフォーマットに従い、表2.3.1に示すフォーマットで記録する。

当該DF内に存在する全ての基礎ファイルに関するEF情報は、それぞれタグ「0x12」以降の連番のタグに対応するデータ実体にそれぞれ記録する。

また、DF情報ファイルは、拡張ファイルへの拡張を可能とする仕様であるが、タグ「0x00」及び「0x10」はDF情報ファイル本体に格納される必要があるため、DF情報ファイルは最低45バイトの領域を確保しなければならない。

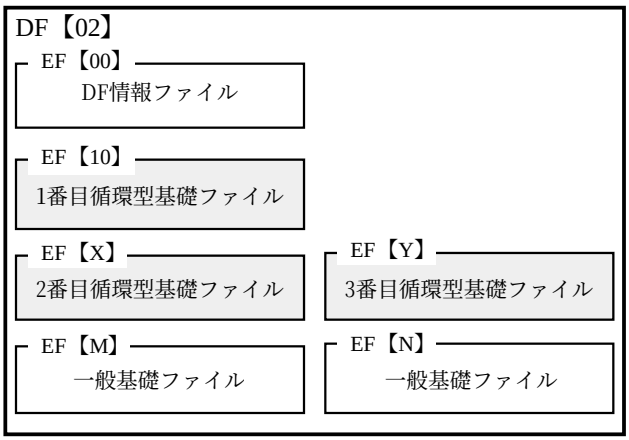
表2.3.1　DF情報ファイルの構成

タグ番号	バイト数	データ内容
0x00	3～5	CAMバージョン番号とEF領域サイズ
0x01	システムリザーブ領域	
0x02		
0x03	可変長	EF-ID変換テーブル
⟩	システムリザーブ領域	
0x10	可変長	DF情報
0x11	10	DF情報ファイルに関するEF情報
0x12	可変長	基礎ファイルに関するEF情報
⟩	⟩	⟩

(3)タグ「0x03」EF-ID変換テーブル

一般的なアプリケーションサービスでは、対象DF内に一つの論理的なEF-IDが唯一に付与される一般基礎ファイルを用いることが通則であると考えられるが、アプリケーションサービス次第では、対象DF内に一つの論理的なEF-IDが複数個に付与される循環型基礎ファイルを必要とする事態が存在する。例えば、保健業務の健康診断等において、過去数年にわたり計測した健康診断結果を履歴として利用・管理する事態がある。

そこで、同一のデータ構造を持つ循環型基礎ファイルを複数設定し、これらの循環型基礎ファイルに同一のデータ集合体を順次記録可能とする機能を実現する。図2.3.2に一例を示す。



実EF-ID	0x10	……	M	N	……	X	Y
論理EF-ID	0x10	……	M	N	……	0x10	0x10

図2.3.2　循環型基礎ファイル構成例

このような循環型基礎ファイルとその他一般基礎ファイルとを識別すると共に、循環型基礎ファイルを構成する複数のグループを識別可能とするために、対象DF内に存在する全基礎ファイルを対象に、ICカードに物理的に割り当てられるEF-ID（以下、実EF-IDという）とCAMにより論理的に割り当てられるEF-ID（以下、論理EF-IDという）を関係づけるEF-ID変換テーブルを、図2.3.3に示すレコードフォーマットによりDF情報ファイルのタグ番号「0x03」に記録する。ただし、実EF-IDと論理EF-IDは、「0x01」～「0x7f」である場合に1バイト、また「0x80」～「0x7fff」である場合に2バイトにより表現する。

実EF-ID1 1or2バイト	論理EF-ID1 1or2バイト	実EF-ID2 1or2バイト	論理EF-ID2 1or2バイト	……	実EF-IDM 1or2バイト	論理EF-IDM 1or2バイト

図2.3.3　EF-ID変換テーブルのフォーマット

また、循環型基礎ファイルにおいてデータ書込・更新・変更等処理が行われた場合にのみ、対象である循環型基礎ファイルの実EF-IDと論理EF-IDをEF-ID変換テーブルの先頭に移動する。

例として、図2.3.2に示すDF【02】における実EF-IDに関する新旧の順序が【M】【Y】【X】……【0x10】……【N】であり、実EF-ID【X】である循環型基礎ファイルを更新する場合を想定する。

M	M	Y	0x10	X	0x10	……	0x10	0x10	……	N	N

図2.3.4　図2.3.2におけるEF-ID変換テーブル（変更前）のフォーマット例

この場合、実EF-ID【X】である循環型基礎ファイルを更新することにより、実EF-IDに関する新旧の順序は【X】【M】【Y】……【0x10】……【N】となることから、EF-ID変換テーブルに記録するレコードフォーマットは図2.3.4に示す通りに変更される。

X	0x10	M	M	Y	0x10	……	0x10	0x10	……	N	N

図2.3.5　図2.3.2におけるEF-ID変換テーブル（変更後）のフォーマット例

(4)タグ「0x10」DF情報

DF情報をタグ「0x10」に以下に示すフォーマットで記述する。

表2.3.2　DF情報のフォーマット

先頭からバイト数	バイト数	データ形式	情報の種類
0	4	ASCII[注2]	認識コード
4	2	バイナリ	初期化日時（年）
6	1	バイナリ	初期化日時（月）
7	1	バイナリ	初期化日時（日）
8	1	バイナリ	初期化日時（時）
9	1	バイナリ	初期化日時（分）
10	16	ASCII[注2]	初期化施設
26	1	バイナリ	拡張ファイル専用DF名長
27	16	バイナリ	拡張ファイル専用DF名
43	1	バイナリ	拡張ファイル論理EF-ID
44	1[注1]	バイナリ	使用基礎ファイル数
総計	最大45バイト		

注1：CAM Version2.0では1バイトに設定するが、将来は2バイトになる可能性がある。
注2：データ形式はシステムを構成するハードウェア・ソフトウェアによって異なる。

!認識コード

「バイト0」～「バイト3」の合計4バイトにCAMにより作成されたことを示す認識コードをコードで記録する。

ただし、CAM標準ソフトウェアVersion2.0では、ASCIIコードで「ICAM」に設定する。

"初期化日時

「バイト4」～「バイト9」の合計6バイトに初期化日時をバイナリで記録する。

●年は西暦を「バイト4」～「バイト5」の合計2バイトにバイナリで表現する。

ただし、バイトスワップを行わないとする。

●月は「バイト6」にバイナリで表現する。

●日は「バイト7」にバイナリで表現する。

●時は「バイト8」にバイナリで表現する。

●分は「バイト9」にバイナリで表現する。

#初期化施設

「バイト10」～「バイト25」の合計16バイトに初期化を行った施設名をコードで記録する。ただし、CAM標準ソフトウェアVersion2.0では、コードとしてASCIIコードを利用する。

$拡張ファイル専用 DF名長

「バイト26」～「バイト27」の合計1バイトに拡張ファイル専用DF名長をバイナリで記録する。

%拡張ファイル専用 DF名

「バイト27」～「バイト42」の合計16バイトに拡張ファイル専用DF名をバイナリで記録する。

&拡張ファイル論理 EF-ID

「バイト43」の1バイトに拡張ファイル論理EF-IDをバイナリで記録する。

'使用基礎ファイル数

「バイト44」または「バイト44」～「バイト45」の1～2バイトに当該DFにおいて利用する基礎ファイル数（DF情報ファイルを含む）をバイナリで記録する。

(5)タグ「0x11」以降のEF情報

EF情報をタグ「0x11」以降に以下に示すフォーマットで記録する。

表2.3.3　EF情報のフォーマット

先頭からバイト数	バイト数	データ形式	情報の種類
0	1	バイナリ	論理EF-ID
1	1	バイナリ	ファイルタイプ
2	2	バイナリ	更新日時（年）
4	1	バイナリ	更新日時（月）
5	1	バイナリ	更新日時（日）
6	1	バイナリ	更新日時（時）
7	1	バイナリ	更新日時（分）
8	2	バイナリ	更新回数
総計			10バイト

!論理 EF-ID

「バイト0」の1バイトに対象とする論理EF-IDをバイナリで記録する。

"ファイルタイプ

「バイト1」の1バイトに表2.3.4に示す6種類の内該当するファイルタイプをバイナリで記録する。

表2.3.4　ファイルタイプ

ファイルタイプ	値
DF情報ファイル	00
基礎ファイル	01
拡張ファイル	02
キーテーブルファイル	03
アクセスコントロールテーブルファイル	04
所有者情報ファイル	05

#更新日時

「バイト2」～「バイト7」の合計6バイトに更新日時をバイナリで記録する。

●年は西暦を「バイト2」～「バイト3」の合計2バイトにバイナリで表現する。
ただし、バイトスワップを行わないとする。

●月は「バイト4」にバイナリで表現する。

●日は「バイト5」にバイナリで表現する。

●時は「バイト6」にバイナリで表現する。

●分は「バイト7」にバイナリで表現する。

$更新回数

対象とするEF情報の更新回数を「バイト8」または「バイト8」～「バイト9」に1～2バイトのバイナリで記録する。

ただし、CAM標準ソフトウェアVersion2.0では、更新回数を1バイトまたは2バイトのバイナリで表現する。上位1ビットを「0」に設定した場合に下位7ビットを、また上位1ビットを「1」に設定した場合に下位15ビットを用いて表現する。

(6)DF情報ファイルに要求される領域サイズ

EF数をnとした場合に、ボリューム情報は最大以下のバイト数になる。

タグ「0x00」		タグ「0x10」		タグ「0x11」～「0x11+n」
7	＋	47	＋	12*n

2.3.3　拡張ファイル

(1)EF-ID

拡張ファイルに付与するShort EF-IDは原則として任意とする。

(2)ファイル構成

既述した通り、拡張ファイルは、発行系コマンドによって割り付けられた基礎ファイル領域が不足する事態において暫定的な対応を図るために、予めICカード内に作成された予備的なファイルである。

一般的に拡張ファイルは、通常のデータを記録するDFとは完全に独立するために、拡張ファイル専用として設定されたDFに作成されるが、対象とするICカードの運用・管理次第では、通常のデータを記録するDF毎に作成される事態が想定されることから、複数の拡張ファイルを作成・使用することも対応可能な仕様としている。ただし、CAM標準ソフトウェアVersion2.0ではサポートしないこととする。

(3)フォーマット

拡張ファイルに記録されるデータは以下に示すデータフォーマットに従う。

! EF識別コード

拡張ファイルは様々なEFから記録されることが想定されることから、各データが属する基礎ファイルを識別するための識別コードとして「EF識別コード」をデータの先頭に必ず付ける。また、EF識別コードは対象ファイルが初めて拡張ファイルにデータを記録する時にCAMが付与することとする。CAM標準ソフトウェアVersion2.0では、「0x01」から順次採番することとする。

EF識別コードは1バイトまたは2バイトのバイナリとする。上位1ビットを「0」に設定した場合に下位7ビットを、また上位1ビットを「1」に設定した場合に下位15ビットを用いて表現する。

"タグ

基礎ファイルにおけるデータフォーマットに準拠する。

#データ長

基礎ファイルにおけるデータフォーマットに準拠する。

$データ実体

基礎ファイルにおけるデータフォーマットに準拠する。

EF識別コード 1～2バイト	タグ	データ長	データ実体

図2.3.5　拡張ファイルにおけるデータフォーマット

2.4　セキュリティ機能

2.4.1　オペレーションカードフォーマットに関する前提条件

CAMの一つの特徴であるオペレーションカード内のファイル構成に関する前提条件は以下の通りである。

(1)ファイル管理単位

CAMが対象とする通常のデータカードフォーマットに関する前提条件を踏襲して、CAMによるオペレーションカードのファイル管理単位をDF単位とする。

(2)基礎ファイル構造

CAM Version2.0は、JIS X6306に規定されている基礎ファイル構造のうち、固定長順編成ファイルと可変長準編成ファイルを対象とする。ただし、CAM標準ソフトウェアVersion2.0は、固定長順編成ファイルのみを対象とする。

(3)ファイル種類

オペレーションカードでは、その用途・目的が通常のデータカードと異なることから、図2.4.1に示す通り、1つのDF内をDF情報ファイル、所有者情報ファイル、キーテーブルファイル、アクセスコントロールテーブルファイルの4つのファイルのみから構成する。しかしながら、アプリケーションサービス次第では、上記の4つのファイル以外にデータカードにおける対象ファイルを作成する事態が想定されることから、データカードとしての基礎ファイルや拡張ファイル等を作成することも許容する。

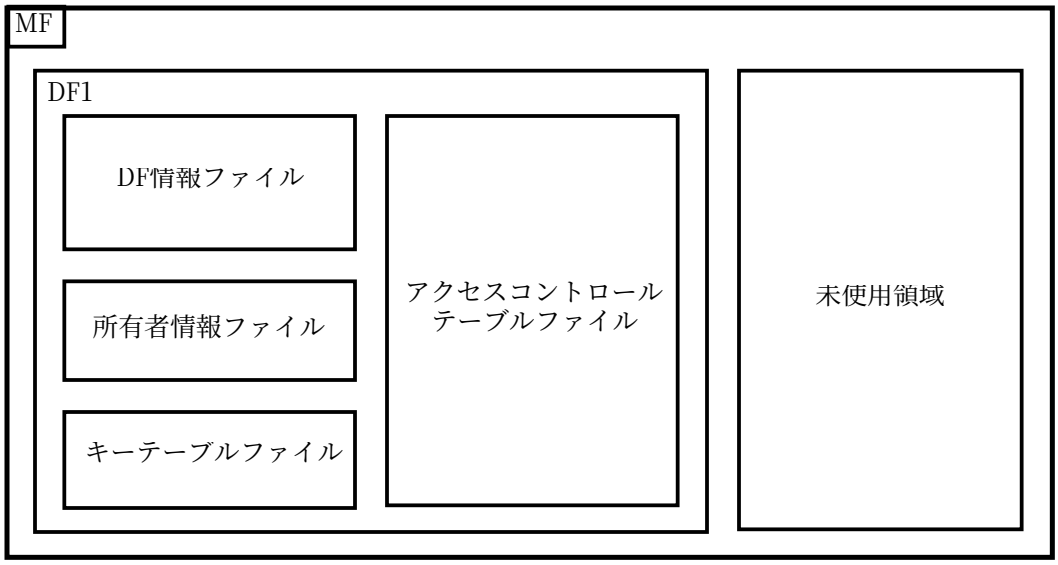


図2.4.1　オペレーションカードフォーマットの概念図

2.4.2　DF情報ファイル

(1)EF-ID

DF情報ファイルに付与するShort EF-IDを「0x1e」に設定する。

(2)ファイル構成

DF情報ファイルは、CAMが対象とするデータフォーマットに従い、表2.4.1に示すフォーマットで記録する。

当該DF内に存在する所有者情報ファイル、キーテーブルファイル、アクセスコントロールファイルに関するEF情報は、それぞれタグ「0x12」以降の連番のタグに対応するデータ実体にそれぞれ記録する。

表2.4.1　DF情報ファイルの構成

タグ番号	バイト数	データ内容
0x00	3～5	CAMバージョン番号とEF領域サイズ
0x01	システムリザーブ領域	
0x02		
0x03	可変長	EF-ID変換テーブル
⟩	⟩	⟩
0x10	36	DF情報
0x11	10	DF情報ファイルに関するEF情報
0x12	可変長	所有者情報ファイルに関するEF情報
0x13	可変長	キーテーブルファイルに関するEF情報
0x14	可変長	アクセスコントロールテーブルファイルに関するEF情報
⟩	⟩	⟩

(3)タグ「0x03」EF-ID変換テーブル

オペレーションカード内のDF情報ファイル、所有者情報ファイル等における実EF-IDと論理EF-IDは一致していることから、原則としてEF-ID変換テーブルを使用しない。

(4)タグ「0x10」DF情報

データカードにおけるDF情報と同一フォーマットで記述する。

(5)タグ「0x11」以降のEF情報

データカードにおけるEF情報と同一フォーマットで記述する。

2.4.3　所有者情報ファイル

(1)EF-ID

所有者情報ファイルに付与するShort　EF-IDは原則として任意とする。ただし、CAM標準ソフトウェアVersion2.0では、「0x02」に設定する。

(2)ファイル構成

所有者情報ファイルは、オペレーションカードを導入するアプリケーションサービス提供者毎に規定された所有者に関する情報、例えば氏名、連絡先、所属機関名、職種・資格等を、CAMが対象とするデータフォーマットに従って記録する。

CAM標準ソフトウェアVersion2.0では、タグ「0x10」に氏名、タグ「0x11」に所属機関情報をJISコードにより記録する。

2.4.4　キーテーブルファイル

(1)EF-ID

キーテーブルファイルに付与するShort　EF-IDは原則として任意とする。ただし、CAM標準ソフトウェアVersion2.0では、「0x03」に設定する。

(2)ファイル構成

キーテーブルファイルは、オペレーションカード所有者に与えられたデータカード内のDF等に対するサービス提供者キー及びサービス端末キー情報を、CAMが対象とするデータフォーマットに従い、表2.4.2に示すフォーマットで記録する。

表2.4.2　キーテーブルファイルの構成

タグ番号	バイト数	データ実体
0x00	3～5	CAMバージョン番号とEF領域サイズ
0x10	可変長	キーテーブル情報

(3)タグ「0x10」キーテーブル情報

キーテーブル情報をタグ「0x10」に表2.4.3に示すフォーマットで記述する。

表2.4.3　キーテーブル情報のフォーマット

先頭からバイト数	バイト数	データ形式	情報の種類
0	1	バイナリ	DF名長
1	16	バイナリ	DF名
17	1	バイナリ	キー数
18	1	バイナリ	KID1
19	1	バイナリ	キー長
20	可変長	バイナリ	キー実体
	1	バイナリ	KID2
	1	バイナリ	キー長
	可変長	バイナリ	キー実体
〜	〜	〜	〜

! DF名長

「バイト0」の1バイトにデータカード内の対象DF名の長さをバイナリで記録する。

" DF名

「バイト1」〜「バイト16」の合計16バイトにデータカード内の対象DF名をバイナリで記録する。

#キー数

「バイト17」の1バイトに、オペレーションカード提供者に与えられている対象DFに関するキー数をバイナリで記録する。

$ KID

「バイト18」の1バイトにキーに関するIDをバイナリで記録する。

%キー長

「バイト19」の1バイトにキーの長さをバイナリで記録する。

&キー実体

「バイト20」からの可変長にキー実体をバイナリで記録する。ただし、キー実体の最大領域は32バイトとする。

2.4.5　アクセスコントロールテーブルファイル

(1)EF-ID

アクセスコントロールテーブルファイルに付与するShort　EF-IDは原則として任意とする。ただし、CAM標準ソフトウェアVersion2.0では、「0x04」に設定する。

(1)ファイル構成

アクセスコントロールファイルは、オペレーションカード所有者に与えられたデータカードのEF内のデータ項目に対するアクセス制御を行うための情報を、CAMが対象とするデータフォーマットに従い、表2.4.4に示すフォーマットで記録する。

表2.4.4　アクセスコントロールテーブルファイルの構成

タグ番号	バイト数	データ実体
0x00	3～5	CAMバージョン番号とEF領域サイズ
0x10	可変長	アクセスコントロールテーブル情報

(2)タグ「0x10」アクセスコントロールテーブル情報

アクセスコントロール情報をタグ「0x10」に表2.4.5に示すフォーマットで記述する。

表2.4.5　アクセスコントロールテーブル情報のフォーマット

先頭からバイト数	バイト数	データ形式	情報の種類
0	1	バイナリ	DF数
1	1	バイナリ	DF名長
2	16	バイナリ	DF名
18	1	バイナリ	EF数
19	1	バイナリ	EF-ID1
20	1～2	バイナリ	最終タグ番号
21	可変長	バイナリ	アクセスコントロールテーブル
	1	バイナリ	EF-ID2
	1～2		最終タグ番号
	可変長	バイナリ	アクセスコントロールテーブル
≀	≀	≀	≀

! DF数

「バイト0」の1バイトにデータカード内の対象DF数をバイナリで記録する。

" DF名長

「バイト1」の1バイトにデータカード内の対象DF名の長さをバイナリで記録する。

DF名

「バイト2」～「バイト17」の合計16バイトにデータカード内の対象DF名をバイナリで記録する。

$ EF数

「バイト18」の1バイトに対象DF内のEF数をバイナリで記録する。

% EF-ID

「バイト19」の1バイトに対象DF内のShortEF-IDをバイナリで記録する。

&最終タグ番号

「バイト20」の1バイトにアクセスコントロールテーブルに記述される対象EF内の最終タグ番号をバイナリで記録する。

'アクセスコントロールテーブル

「バイト21」からの可変長にコントロールアクセス権をビット列で記録する。各タグに対するアクセス権は、Read権限とWrite権限の順にタグ番号 0x10 から最終タグ番号までについて記録される。ただし、Read権限とWrite権限は「1:禁止」「0:許可」いずれかにより記述することとする。

また、データはバイト単位で表現し、ビット列の長さが8で割り切れない場合には、末尾を0埋めする。従って、データの長さは、

{（最終タグ番号－第1タグ番号＋1）×2} ÷8

を小数点以下切り上げたバイト数となる。

例として、図2.4.2に示すデータでは、タグ番号 0x11, 0x12, 0x14 は書込禁止、タグ番号 0x15, 0x17, 0x19 は読み込み禁止、タグ番号 0x16, 0x18 は読み込み書き込み禁止であることを示しており、データ全体の長さは 3バイトとなる。

バイト	第1バイト								第2バイト								第3バイト				
タグ番号（0x）	10		11		12		13		14		15		16		17		18		19		
ビット値	0	0	0	1	0	1	0	0	0	1	1	0	1	1	1	0	1	1	1	0	0埋め
Read／Write権限	R	W	R	W	R	W	R	W	R	W	R	W	R	W	R	W	R	W	R	W	

図2.4.2　アクセスコントロールテーブル例

2.4.6　その他留意事項

オペレーションカードを導入するに当たっては、オペレーションカード所有者に与えられているアクセス制御情報を、ホストコンピュータまたはリーダ・ライタ端末等のアプリケーションプログラムにおいて読み込み・書き込みするためのデータファイル等を準備することが必要不可欠である。データフォーマットはCAMにおけるデータフォーマットと同一とする。

ただし、データファイル名等は、アプリケーションプログラムや製造メーカ毎に異なることが予想されることから、ベンダー名、アプリケーションサービス名等を記録するタグを設定する等によって名称等を区別・認識するための対策を講じる必要がある。

2.5　新規機能

CAM　Version2.0では、CAM　Version1.0,1.1に既に組み込まれていた機能であるデータ動的変更機能、ファイルサイズ動的変更機能、さらにセキュリティ機能以外に、新たな要求仕様として提案されたカード所有者認証のためのPIN設定機能を付加する。以下、その概要を紹介する。

さらに、CAM Version2.0は、16社仕様準拠ICカードだけでなくJIS準拠ICカードに対応可能な機能仕様を規定することを踏まえ、CAM Version2.0において16社仕様準拠ICカードとJIS準拠ICカードの共存に向けた前提条件、及び実現方法について以下に紹介する。

2.5.1　カード所有者認証のためのPIN設定機能

(1)前提条件

CAMでは、CAM　Version1.0よりデータカード内のDF・EF・データ項目それぞれに対する操作者のアクセス権限の正当性を確認する手段としてオペレーションカードを導入していることは既述の通りである。

新たな要求仕様として提案されたカード所有者認証のためのPIN設定機能は、オペレーションカードと同様に「データカード内のDF・EF・データ項目それぞれに対する操作者のアクセス権限の正当性を確認する一手段」として位置づけることも可能であるが、アプリケーションサービスの提供形態等に依存する可能性が非常に高いと考えられる。

従って、ここでは、カード所有者認証のためのPIN設定機能を操作者のアクセス権限の正当性を確認するためにではなく、あくまでもカード所有者を認証するための一手段として位置づける。

(2)PINの所在

一般に、16社仕様ICカードでは、図2.5.1に示す通りPINをMF直下またはDF配下に設定することが可能であることに基づき、CAM Version2.0ではMF直下及びDF配下の両者への対応を図ることとする。

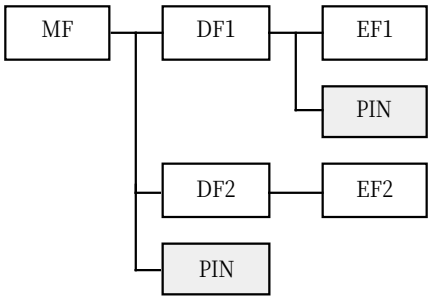


図2.5.1　PIN情報の所在例

ただし、PINをMF直下に設定する場合には以下に示す二点に留意する必要がある。

まず、JIS　X6306では、DFを選択した後にMFを選択することを規定していないため、MF直下に設定されたPINのCAMによる照合は、カードを活性化または再活性化した直後のみ実行可能である。

また、JIS X6306では、MF直下のPIN設定に係わらずDFを直接アクセスすることを可能とするため、カード所有者の本人確認を行うアプリケーションサービスに関するDFでは、DF配下にPINを設定することを必須条件とする。

(3)実現方法

CAM Version2.0では、カード所有者認証のためのPIN設定機能を実現する関数を新規に規定する。

2.5.2　JIS準拠ICカードと16社仕様準拠ICカードの共存への対応

(1)前提条件

将来、アプリケーションサービス次第では、従来より市場に流通している16社仕様準拠ICカードと新たに市場に流通するJIS準拠ICカードが共存する事態が想定される。CAMの基本概念が「カード世代間及びカード製造メーカ間の吸収」であることに鑑みるならば、CAM　Version2.0において16社仕様準拠ICカードとJIS準拠ICカードの共存を保証することは極めて重要であり、そのためには、以下に示す前提条件が整備されていなければならない。

表2.5.1　JIS準拠ICカードと16社仕様準拠ICカードの共存に向けた前提条件

項目	前提条件
DF名	16社仕様準拠ICカードとJIS準拠ICカードにおいて対象アプリケーションサービスに関するDF名が一対一に対応づけられている。
EF-ID	16社仕様準拠ICカードとJIS準拠ICカードにおいて対象アプリケーションサービスに関するEF-IDが一対一に対応づけられている。
タグ番号とデータ項目名等の関係	16社仕様準拠ICカードとJIS準拠ICカードにおいて対象アプリケーションサービスに関するタグ番号とデータ項目名等の関係が標準化されている。
サービス提供者・端末キー	16社仕様準拠ICカードとJIS準拠ICカードにおいて対象となるDF・EFに関するサービス提供者・端末キーが標準化されている。
データ項目に対するアクセス権限	16社仕様準拠ICカードとJIS準拠ICカードにおいてデータ項目に対するアクセス権限が標準化されている。

(2)実現方法

以上に示す前提条件が整備されている環境下では、ICカードとリーダ・ライタ端末間のAnswer To ResetによってJIS準拠ICカードと16社仕様準拠ICカードのカード種別を判定し、DF情報ファイルに記録されているCAMのバージョン番号を参照することにより対応可能である。

3.内容アクセスマネージャVersion2.0の関数仕様

3.1　概要

既に述べた通り、CAMはカードメーカ、電機メーカ各社等により製造される仕様が異なるICカードを、また16社仕様準拠ICカードやJIS準拠ICカード等のように世代が異なるICカードを相互に運用可能とすることを主たる目的として提案・開発された概念である。将来、電機メーカ、システムインテグレータ等が、CAMに規定された関数仕様に基づき、処理速度等の性能向上を目的としてCAM標準ソフトウェアを改変する、また新規に開発する場合に、CAMに規定された同一関数によるリターン値が異なる、また関数を実行する順序が異なる等の事態を是非とも回避しなければならないが、その反面実現方法等を制限しないことが極めて重要である。

以上を踏まえ、CAM　Version2.0は、図3.1.1に示す通り、その関数構造をアプリケーション・インタフェース関数、アプリケーション・インタフェース関数を実現するための内部関数、そしてデバイスドライバ共通インタフェース関数を制御するための内部関数の三階層から構成することにより、その上位に位置するアプリケーションプログラムとのインタフェース条件、及びその下位に位置するデバイスドライバ共通インタフェース関数とのインタフェース条件を規定する。

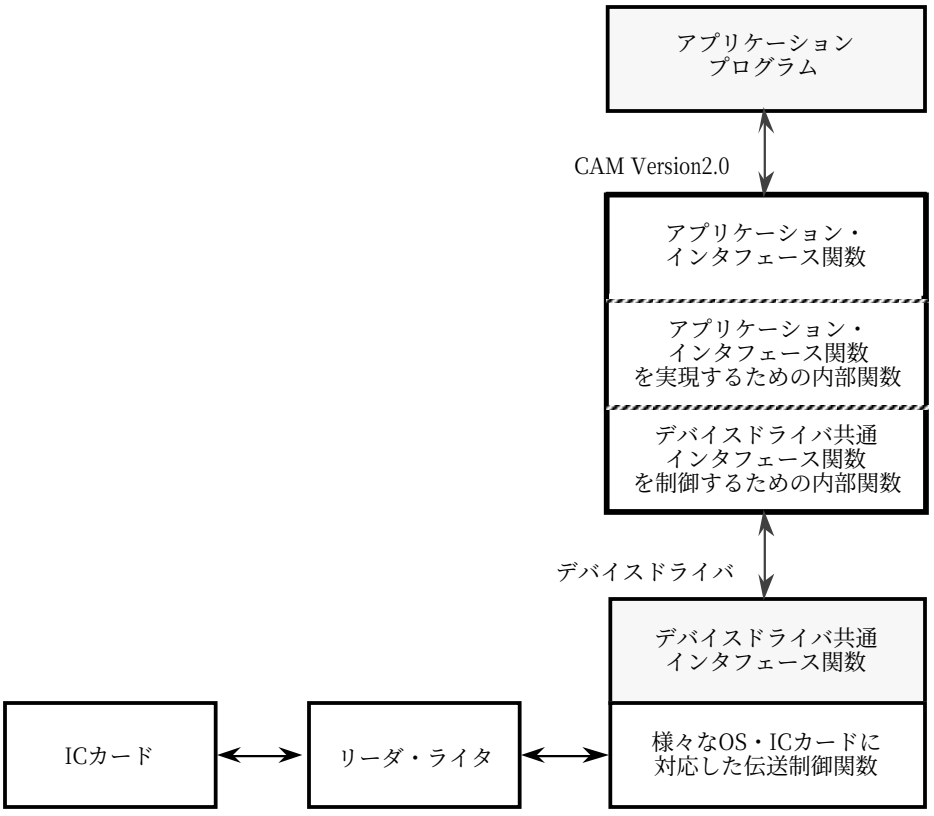


図3.1.1　CAM Version2.0の関数構成

3.1.1　アプリケーション・インタフェース関数

アプリケーション・インタフェース関数のインタフェース条件として、CAM Version1.0シリーズに倣い、呼出形式、引数、機能、実行条件、アクセス許可モード、リターン値等を規定する。

3.1.2　デバイスドライバ共通インタフェース関数

デバイスドライバ共通インタフェース関数は、CAM Version2.0をICカード仕様に依存する部分と依存しない部分を可能な限り切り分けることを目的とした関数であり、CAM Version 2.0において新たに規定した関数である。

本仕様に基づいて作成されるデバイスドライバ共通インタフェース関数プログラムでは、ICカードに関するコマンド及びリーダ・ライタに関するコマンドについて以下の通り対応することを原則とする。

ICカードに関するコマンドに関しては、16社標準仕様に基づくコマンド仕様とJIS　X6306に記載される必須コマンド仕様の両者または何れか一方に対応することを原則とする。また、リーダ・ライタに関するコマンドに関しては、ICカードのような標準的なコマンド仕様が存在しないことから、リーダ・ライタに具備することが必要不可欠であると想定される最小限の機能をコマンド関数として規定することを原則とする。

また、デバイスドライバ共通インタフェース関数仕様では、16社標準仕様に基づくコマンド仕様とJIS　X6306に記載される必須コマンド仕様のうち同等の機能を実現するコマンドを一つの関数により記述し、どちらに対応したコマンドであるかを関数におけるパラメータで指定することとする。ただし、本仕様は、今後各業界における標準化動向や市場からの要望により将来的に拡張される可能性がある。

表3.1.1　アプリケーション・インタフェース関数一覧（その1）

関数名	機能概要
icam_initial	CAMを初期化する。
icam_open_file	指定した基礎ファイルをオープンする。
icam_close_file	オープンした基礎ファイルをクローズする。
icam_card_in	カードをリーダ／ライタに挿入する。
icam_card_out	カードをリーダ／ライタから排出する。
icam_create_file	初期化された基礎ファイルをDF情報ファイルに登録する。
icam_create_rotation_file	初期化された循環型基礎ファイルをDF情報ファイルに登録する。
icam_write_item	基礎ファイルにデータを書き込む。
icam_read_item	基礎ファイルからデータを読み出す。
icam_read_item_sequential	基礎ファイルからデータを順に読み出す。
icam_reset_item_counter	ポインタを対象基礎ファイルの先頭データに移動する。
icam_get_file_size	基礎ファイルの領域サイズを調べる。
icam_erase_item	基礎ファイル内のデータを消去する。
icam_erase_file	基礎ファイルを消去することにより、CAMから解放する。
icam_list_file	基礎ファイルのリストを出力する。
icam_init_file	基礎ファイルをCAM用に初期化する。
icam_change_theoretical_id	循環型基礎ファイルの論理的なEF-IDを変更する。
icam_get_theoretical_id	実EF-IDから論理的なEF-IDを得る。
icam_get_actual_id	論理的なEF-IDから実EF-IDを得る。
icam_count_file_number	論理的なEF-IDが同一の循環型基礎ファイル数を得る。
icam_get_update_info	循環型基礎ファイルの更新日時を得る。
icam_get_group_item	読み込まれたデータのデータ実体数を判定する。
icam_set_group_item	読み込まれたデータのデータ実体数を指定する。
icam_pick_item_data	複数個のデータ実体から構成されるデータから任意の順番のデータ実体を読み出す。
icam_append_item_data	複数個のデータ実体から構成されるデータにデータ実体を追加する。
icam_remove_item_data	複数個のデータ実体から構成されるデータから任意の順番のデータ実体を削除する。
icam_count_item_data	複数個のデータ実体から構成されるデータのデータ実体数を得る。
icam_item_len	複数個のデータ実体から構成されるデータのデータ実体全長を得る。
icam_erase_all_item	オープンしているファイル内の管理情報を除く全データを削除する。
icam_get_error_info	エラー、コマンド、ステータス等に関するCAMの内部情報を得る。
icam_get_file_info	オープンしているファイルに関するCAMの内部情報を得る。
icam_get_df_info	DF情報に関するCAMの内部情報を得る。

表3.1.1　アプリケーション・インタフェース関数一覧（その2）

関数名	機能概要
icam_identify_operetion	オペレーションカードのデータをCAMにロードする。
icam_terminate_operation	オペレーションカードのデータをアンロードする。
icam_get_operation_info	オペレーションカードの所有者情報を得る。
icam_verify_pin	設定されたピンの照合を行う。

表3.1.2　デバイスドライバ共通インタフェース関数一覧

関数名	機能概要
std_com_open	ICカードドライバをオープンする。また、対応可能なICカードの種類を提示する。
std_com_close	ICカードドライバをクローズする。
std_card_in	ICカードを活性化する。初期応答情報の解析を行い、活性化したICカードの種類を判定する。
std_card_out	ICカードを非活性化し、リーダ・ライタからカードを排出させる。
std_card_reset	ICカードがリーダ・ライタに挿入されているときに、ICカードを再活性化する。
std_card_sense	ICカードがリーダ・ライタのスロット内に挿入されているか否かを確認する。
std_select_DF	SELECT DF（JIS X6306）またはSELECT ADF（S型ICカード）を実行する。
std_verify	VERIFY（JIS X6306）またはVERIFY KEY（S型ICカード）を実行する。
std_read_record	READ RECORD(S)（JIS X6306）またはREAD RECORD(S型ICカード）を実行する。
std_write_record	WRITE RECORD（JIS X6306）またはWRITE RECORD（S型ICカード）を実行する。
std_append_record	APPEND RECORD（JIS X6306）を実行する。
std_update_record	UPDATE RECORD（JIS X6306）を実行する。
std_erase_all_records	ERASE ALL RECORDS（JICSAP滝川市向け仕様）またはERASE ALL RECORDS（S型ICカード）を実行する。
std_other_command	上記以外のコマンドを実行させる場合に使用する。コマンドブロックのインフォメーション部の作成は、本関数の呼び出し側が行う。

3.2　アプリケーション・インタフェース関数仕様

3.2.1　icam_initial

(1)呼出形式

```
short    icam_initial( flag, iname )
```

(2)引き数

```
int flag;                /*   将来利用するための予備変数   */
char *iname;             /*   利用施設   */
```

(3)機能

CAMの初期化を行う。具体的には、各種内部変数、icam_verify_pinでメモリに読み込まれたPIN情報等を消去する。flagは将来の利用のためにリザーブする。inameには利用施設を指定する。

(4)リターン値一覧

リターン値	意味
0	正常

3.2.2　icam_open_file

(1)呼出形式

short　icam_open_file(fname, nlen, tid, mode, fd, opt, kroder)

(2)引き数

```
char *fname;          /*  DF名  */
int nlen;             /*  DF名の長さ  */
shrot tid;            /*  論理的なEF-ID  */
int mode;             /*  アクセス許可モード  */
int *fd;              /*  ファイルディスクリプタ（戻り値）  */
int opt;              /*  循環型基礎ファイル番号  */
int kroder;           /*  循環型基礎ファイル変更フラグ  */
```

(3)機能

icam_open_file関数は、fnameおよびidで指定される基礎ファイルを、modeの設定にしたがってオープンする。ただし、オープンされていない基礎ファイル内のデータへのアクセスは許可されない。fnameは16バイト以内のバイナリで表現されるDF名である。idはEF-IDを示す。modeは基礎ファイルのアクセス許可モードを示す。アクセス許可モードは、一般基礎ファイルを指定するためのReadOnly、WriteOnly、ReadWrite、循環型基礎ファイルを指定するためのRotation、OverWriteを設定する。

さらに、optとkroderはfname及びidで指定される基礎ファイルが循環型基礎ファイルである場合に利用するフラグである。optは複数存在する循環型基礎ファイルから対象ファイルを指定するための番号である。kroderは循環型基礎ファイルの更新日時変更に伴う循環型基礎ファイルの並び替えを制御するフラグであり、1である場合に並び替えを実行し、それ以外の場合には並び替えを実行しない。

icam_open_file関数では、指定されたアクセス許可モードに応じてアクセス権限の確認を行う。アクセス権限が満たされない場合にはリターン値にてその旨を通知する。

icam_open_file関数が正常に行われた場合、オープンされた基礎ファイルのファイルディスクリプタをfdに設定する。基礎ファイルをオープンした後、ファイルディスクリプタを用いてicam_read_item関数、icam_write_item関数、icam_erase_item関数、icam_read_item_sequential関数、icam_reset_item_counter関数によって基礎ファイル内のデータへのアクセスを可能にする。

CAMがロードできるDF情報の最大数およびCAMがオープンできる基礎ファイルの最

大数は任意とする。

(4)実行条件

■nlen≦16でなければならない。

■ICカード内にfnameで指定したDFが存在し、そのDF内にDF情報ファイルが存在しなければならない。

■DF情報ファイル内にidで指定した基礎ファイルのEF情報が存在しなければならない。

■少なくともDF情報ファイルのRead権及び指定した基礎ファイルのRead権を持っていなければならない。

■既にオープンされている基礎ファイルのオープンは無効とする。

■指定した基礎ファイルは一般基礎ファイルまたは循環型基礎ファイルでなければならない。

■使用するCAMがCAM　Version1.0である場合には、自治省が主導する地域カードシステムの循環型基礎ファイルのみをサポートする。ただし、記録日時情報がタグ番号0xFFのデータ実体に記録されている場合のみ有効とする。

■optはアクセス許可モードmodeにRotationが指定された場合のみ有効とする。

(5)アクセス許可モード

①ReadOnly

基礎ファイル内のデータのReadのみを可能にする。このモードでicam_open_file関数を実行するためには、ICカード内のDF情報ファイルおよび指定した基礎ファイルにアクセスするためのアクセス権としてDF情報ファイルのRead権、指定した基礎ファイルのRead権がなければならない。

②WriteOnly

基礎ファイル内のデータのWriteのみを可能にする。このモードでicam_open_file関数を実行するためには、ICカード内のDF情報ファイルおよび指定した基礎ファイルにアクセスするためのアクセス権としてDF情報ファイルのRead権およびWrite権、指定した基礎ファイルのRead権およびWrite権がなければならない。

③ReadWrite

基礎ファイル内のデータのReadおよびWriteを可能にする。このモードでicam_open_file関数を実行するためには、ICカード内のDF情報ファイルおよび指定した基礎ファイルにアクセスするためのアクセス権としてDF情報ファイルのRead権およびWrite権、指定した基礎ファイルのRead権およびWrite権がなければならない。

$ Rotation

循環型基礎ファイルをオープンすることを示す。アクセス許可モードがRotationである場合には、引数optによって更新日時の新旧による循環型基礎ファイル指定を可能とする。例えば、1993,1994,1995年に記録されたデータが格納されている場合には、1995年のファイルに0を、1994年のファイルに1を、1993年のファイルに2を指定する。ただし、アクセス許可モードとしてRotationが指定されていない場合には、更新日時が最新であるファイルがオープンされる。

% OverWrite

循環型基礎ファイルをオープンした際に、ファイルの内容をすべて消去することを示す。OverWriteは、例えば古い循環型基礎ファイルの内容をすべて最新の内容に更新する場合に使用する。ただし、アクセス許可モードとしてOverWriteが指定されていない場合には、オープンした際にファイルの内容は消去されない。

アクセス許可モードは、データのRead/Write権限を決めるコンテントアクセス権よりも優先される。例えば、アクセス許可モードWriteOnlyでオープンした基礎ファイル内の一データのコンテントアクセス権がRead可能であっても、このデータに対するReadは、許可されない。

(6)リターン値一覧

リターン値	意味
0	正常
100	ICカード内に指定したDFが存在しない。
101	ICカード内の指定したDFにDF情報ファイルが存在しない。
102	ICカード内の指定したDFに、指定した基礎ファイルが存在しない。
103	ICカードに対するアクセス権限を満たさない。
104	ICカードへのアクセス異常。
105	ICカードのコマンド異常。
106	ICカードのメモリ異常。
107	ファイルが閉塞している。
108	指定したレコードが存在しない。
151	リーダ／ライタへのアクセス異常。
200	これ以上ファイルをオープンできない。
201	指定した基礎ファイルは既にオープンされている。
204	DF情報ファイルが定義外である。
205	DF情報ファイル内に指定した基礎ファイルのEF情報が存在しない。
206	指定した基礎ファイルが定義外である。
210	循環型基礎ファイルでない。
211	指定した順番の循環型基礎ファイルは存在しない。
400	拡張ファイルアクセス異常。
600	CAMバージョンが一致しない。
601	CAMが用意しているバッファの容量不足。

注)伝送制御異常に関するリターン値は省略する。

3.2.3　icam_close_file

(1)呼出形式

short　icam_close_file(fd)

(2)引き数

int fd;　　/*　ファイルディスクリプタ　*/

(3)機能

オープンされている基礎ファイルをクローズする。fdにファイルディスクリプタを指定する。一度、オープンした基礎ファイルは、本関数を用いて必ずクローズしなければならず、クローズされていないファイルが存在する場合には、全てクローズしてからカードを排出する。基礎ファイルをクローズした後は、指定されたfdを解放する。クローズされた基礎ファイル内のデータに対するアクセスは許可されない。

ICカードへのアクセス回数を少なくすること、及びEEPROMの更新回数を少なくするために基礎ファイルをオープンしている間は、icam_write_item関数、icam_erase_item関数によって更新された基礎ファイル内のデータおよびDF情報ファイル内のデータを蓄積し、本関数実行時に蓄積した更新データをICカードに書き込むことを妨げない。

(4)実行条件

■icam_open_file関数によってオープンされた基礎ファイルのファイルディスクリプタを指定しなければならない。

(5)リターン値一覧

リターン値	意味
0	正常
100	ICカード内に指定したDFが存在しない。
101	ICカード内の指定したDFにDF情報ファイルが存在しない。
102	ICカード内の指定したDFに、指定した基礎ファイルが存在しない。
103	ICカードに対するアクセス権限を満たさない。
104	ICカードへのアクセス異常。
105	ICカードのコマンド異常。
106	ICカードのメモリ異常。
107	ファイルが閉塞している。
108	指定したレコードが存在しない。
151	リーダ／ライタへのアクセス異常。
202	指定したファイルディスクリプタは使用されていない。
203	指定したファイルディスクリプタは不正である。
400	拡張ファイルアクセス異常。
600	CAMバージョンが一致しない。
601	CAMが用意しているバッファの容量不足。

注)伝送制御異常に関するリターン値は省略する。

3.2.4　icam_card_in

(1)呼出形式

short　icam_card_in(sec)

(2)引き数

int sec;　　/*　時間（秒）　*/

(3)機能

リーダ／ライタにカードの挿入を許可する。secはカードの受付待ち時間を秒単位で指定する。

secにて指定される時間内にカードがリーダライタに挿入された場合（既に挿入済みの場合も含む）、カードの活性化が正常に行われた時点で本関数を終了する。secで指定される時間を越えてもカードがリーダライタに挿入されない場合、リーダライタのカード受付を禁止し本関数を終了する。本関数の終了要因はリターン値で通知する。

(4)リターン値一覧

リターン値	意味
0	正常
110	サポートされていない種類のカードが挿入された。
150	指定した時間内にICカードがリーダ／ライタに挿入されなかった。
151	リーダ／ライタへのアクセス異常。

注)伝送制御異常に関するリターン値は省略する。

3.2.5　icam_card_out

(1)呼出形式

short　icam_card_out(void)

(2)引き数

無し

(3)機能

リーダ／ライタからカードを排出する。オープンしている基礎ファイルが存在する場合には、全てクローズして排出する。

(4)リターン値一覧

リターン値	意味
0	正常
151	リーダ／ライタへのアクセス異常。

注)伝送制御異常に関するリターン値は省略する。

3.2.6　icam_create_file

(1)呼出形式

```
short    icam_create_file( fname, nlen, id )
```

(2)引き数

```
char *fname;          /*   DF名    */
int nlen;             /*   DF名の長さ    */
short id;             /*   EF-ID    */
```

(3)機能

fnameとidで指定される基礎ファイルをCAMで利用可能にするために、CAMによる制御に使用する情報をDF情報ファイルおよび該当基礎ファイルに書き込む。DF情報ファイルには指定された基礎ファイルのEF情報を追加登録する。ファイルタイプは基礎ファイル「01」とする。また、指定された基礎ファイルにはCAMバージョン番号とその領域サイズを記述するデータを登録する。

DF情報ファイルにDF情報が存在しないDFに対して本関数を実行した場合、DF情報を作成・登録する。ファイルタイプはDF情報ファイル「00」とする。

本関数を実行していない基礎ファイルに対してCAMに規定された他関数を実行することはできない。

(4)実行条件

■nlen≦16でなければならない。

■ICカード内にfnameで指定したDFが存在すると共に、そのDF内にDF情報ファイルが存在しなければならない。ただし、DF情報ファイル内におけるDF情報の存在有無は問わない。

■fnameとidで指定された基礎ファイルが存在しなければならない。

■fnameとidで指定された基礎ファイルは、予めicam_init_file関数によりCAM用に初期化されていなければならない。

■既に生成されている基礎ファイルに対する本関数は無効とする。

■DF情報ファイル及び指定した基礎ファイルのRead権とWrite権を持っていなければならない。

■fnameで指定したDFのDF情報ファイルは、既にDF情報が書き込まれているか、icam_init_file関数等によりCAM用に初期化されていなければならない。

(5)リターン値一覧

リターン値	意味
0	正常
100	ICカード内に指定したDFが存在しない。
101	ICカード内の指定したDFにDF情報ファイルが存在しない。
102	ICカード内の指定したDFに、指定したEFが存在しない。
103	ICカードに対するアクセス権限を満たさない。
104	ICカードへのアクセス異常。
105	ICカードのコマンド異常。
106	ICカードのメモリ異常。
107	ファイルが閉塞している。
108	指定したレコードが存在しない。
151	リーダ／ライタへのアクセス異常。
204	DF情報ファイルが定義外である。
207	指定した基礎ファイルは既にクリエイトされている。
208	DF情報ファイルあるいは指定した基礎ファイルにこれ以上データを追加できない。
400	拡張ファイルアクセス異常。
401	アクセス中のICカードは拡張ファイルをサポートしていない。
600	CAMバージョンが一致しない。
601	CAMが用意しているバッファの容量不足。

注)伝送制御異常に関するリターン値は省略する。

3.2.7　icam_create_rotation_file

(1)呼出形式

short　icam_create_rotation_file(fname, nlen, id, tid)

(2)引き数

```
char *fname;          /*  DF名  */
int nlen;             /*  DF名の長さ  */
short id;             /*  実EF-ID  */
short tid;            /*  論理的なEF-ID  */
```

(3)機能

循環型基礎ファイルを作成し、論理的なEF-IDを割り当てる。

fnameとidで指定される循環型基礎ファイルをCAMで利用可能にするために、CAMによる制御に使用する情報をDF情報ファイルおよび該当基礎ファイルに書き込む。DF情報ファイルには指定された循環型基礎ファイルのEF情報を追加登録する。ファイルタイプは基礎ファイル「01」とする。また、指定された循環型基礎ファイルにはCAMバージョン番号とその領域サイズを記述するデータを登録する。

DF情報ファイルにDF情報が存在しないDFに対して本関数を実行した場合、DF情報を作成・登録する。ファイルタイプはDF情報ファイル「00」とする。

本関数を実行していない循環型基礎ファイルに対してCAMに規定された他関数を実行することはできない。

また、既に存在する循環型基礎ファイルの論理的なEF-IDを変更する機能も併せ持つ。

(4)実行条件

■nlen≦16でなければならない。

■ICカード内にfnameで指定したDFが存在すると共に、そのDF内にDF情報ファイルが存在しなければならない。ただし、DF情報ファイル内におけるDF情報の存在有無は問わない。

■fnameとidで指定された基礎ファイルが存在しなければならない。

■fnameとidで指定された基礎ファイルは、予めicam_init_file関数によりCAM用に初期化されていなければならない。

■既に生成されている基礎ファイルに対する本関数は無効とする。

■DF情報ファイル及び指定した基礎ファイルのRead権とWrite権を持っていなければなら

ない。

■fnameで指定したDFのDF情報ファイルは、既にDF情報が書き込まれているか、icam_init_file関数等によりCAM用に初期化されていなければならない。

■CAM Version2.0で初期化済循環型基礎ファイルのみ実行可能とする。CAM Version2.0では、JIS準拠ICカードを対象としているため、実及び論理的なEF-IDは0x01から0x1dまでの値のみ使用可能である。

(5)リターン値一覧

リターン値	意味
101	ICカード内に指定したDFが存在しない。
204	DF情報ファイルが定義外である。
207	指定した基礎ファイルは既にクリエイトされている。
600	CAMバージョンが一致しない。

注)伝送制御異常に関するリターン値は省略する。

3.2.8　icam_write_item

(1)呼出形式

short　icam_write_item(fd, tag, data, len)

(2)引き数

```
int fd;                 /*  ファイルディスクリプタ  */
int tag;                /*  タグ  */
unsigned char *data;    /*  データ実体  */
int len;                /*  データ長  */
```

(3)機能

icam_open_file関数によって既にオープンされている基礎ファイルに対して、tagで指定するデータを書き込む。fdには該当する基礎ファイルに対応するファイルディスクリプタを設定する。tagには書き込むべきデータのタグを、lenにはそのデータ長を、dataにはそのデータを設定する。

CAMが該当する基礎ファイルのアクセスコントロールテーブルを持っている場合、アクセスコントロールテーブル内から指定されたタグのコンテントアクセス権を参照し、Writeが許可されている場合のみデータを書き込む。

CAMが該当するEFのアクセスコントロールテーブルを持っていない場合、またはアクセスコントロールテーブルを持っているが、アクセスコントロールテーブル内に該当するタグのコンテントアクセス権が設定されていない場合、指定されたタグはWriteが許可されているとみなし、データを書き込む。

ICカード内の基礎ファイルの内容を更新する場合、DF情報ファイル内の該当するEF情報も同時に更新する。ただし、本関数を実行する都度、ICカードへの書き込み処理を行うと処理速度の低下及びEEPROMの更新回数が増加するため、icam_close_file関数の実行時にまとめてICカードへの書き込み処理を行うようにしても良い。

複数個のデータ実体を持つデータを書き込む場合、data内のデータは実際にICカードに記録するフォーマットの形式で与える。

本関数で書き込む前に、予めicam_set_group_item関数を用いることによって、書き込むデータが複数個のデータ実体を持つか否かを指定する必要がある。

また、icam_append_item_data関数等を用いることによって、複数個のデータ実体を持つデータの各データ実体を変更・追加することができる。

(4)実行条件

■対象となる基礎ファイルは、icam_open_file関数によって既にオープンされていなければならない。

■対象となる基礎ファイルは、アクセス許可モードWriteOnlyまたはReadWriteでオープンされていなければならない。

■指定されたタグのコンテントアクセス権によってWriteが許可されていなければならない。

■指定されたタグはCAMによりリザーブされているタグ「0x00」～「0x0f」でない。

(5)リターン値一覧

リターン値	意味
0	正常
100	ICカード内に指定したDFが存在しない。
101	ICカード内の指定したDFにDF情報ファイルが存在しない。
102	ICカード内の指定したDFに、指定した基礎ファイルが存在しない。
103	ICカードに対するアクセス権限を満たさない。
104	ICカードへのアクセス異常。
105	ICカードのコマンド異常。
106	ICカードのメモリ異常。
107	ファイルが閉塞している。
108	指定したレコードが存在しない。
202	指定したファイルディスクリプタは使用されていない。
203	指定したファイルディスクリプタは不正である。
208	DF情報ファイルあるいは指定した基礎ファイルにこれ以上データを追加できない。
209	オープン時のアクセス許可モードの条件に合わない処理を実行しようとした。
300	データのタグ番号が不正である。
302	コンテントアクセス権を満たさない。
310	複数個のデータ実体から構成されるデータの記述形式が不正である。
400	拡張ファイルアクセス異常。
401	アクセス中のICカードは拡張ファイルをサポートしていない。
600	CAMバージョンが一致しない。
601	CAMが用意しているバッファの容量不足。

注)伝送制御異常に関するリターン値は省略する。

3.2.9　icam_read_item

(1)呼出形式

```
short    icam_read_item( fd, tag, data, dlen, len )
```

(2)引き数

```
int fd;                  /*  ファイルディスクリプタ  */
int tag;                 /*  タグ  */
unsigned char *data;     /*  データ実体  */
int dlen;                /*  データバッファの長さ  */
int *len;                /*  データ長  */
```

(3)機能

icam_open_file関数によって既にオープンされている基礎ファイルから、tagで指定するデータを読み出す。fdには該当する基礎ファイルに対応するファイルディスクリプタを設定する。tagには読み出すべきデータのタグ番号を、dlenにはデータバッファdataの長さをそれぞれ設定する。本関数を実行した後、dataには指定したタグのデータが、lenにはそのデータ長が設定される。

CAMが該当する基礎ファイルのアクセスコントロールテーブルを持っている場合、アクセスコントロールテーブル内から指定されたタグのコンテントアクセス権を参照し、Readが許可されている場合のみデータを読み出す。

CAMが該当する基礎ファイルのアクセスコントロールテーブルを持っていない場合、またはアクセスコントロールテーブルを持っているが、アクセスコントロールテーブル内に該当するタグのコンテントアクセス権が設定されていない場合、指定されたタグはReadが許可されているとみなし、データを読み出す。

複数個のデータ実体を持つデータを読み出す場合、data内のデータはICカードに記録されたフォーマットの形式で得られる。

本関数で読み出した後、icam_get_group_item関数を用いることによって、読み出されたデータが複数個のデータ実体を持つか否かを知ることができる。

また、icam_pick_item_data関数等を用いることによって、複数個のデータ実体を持つデータの各データ実体を読み出すことができる。

(4)実行条件

■対象となる基礎ファイルは、icam_open_file関数によって既にオープンされていなけれ

ばならい。

■対象となる基礎ファイルは、アクセス許可モードReadOnlyまたはReadWiteでオープンされていなければならない。

■指定されたタグはコンテントアクセス権によってReadが許可されていなければならない。

■指定されたタグはCAMによりリザーブされているタグ「0x00」～「0x0f」でない。

(5)リターン値一覧

リターン値	意味
0	正常
100	ICカード内に指定したDFが存在しない。
101	ICカード内の指定したDFにDF情報ファイルが存在しない。
102	ICカード内の指定したDFに、指定した基礎ファイルが存在しない。
103	ICカードに対するアクセス権限を満たさない。
104	ICカードへのアクセス異常。
105	ICカードのコマンド異常。
106	ICカードのメモリ異常。
107	ファイルが閉塞している。
108	指定したレコードが存在しない。
202	指定したファイルディスクリプタは使用されていない。
203	指定したファイルディスクリプタは不正である。
209	オープン時のアクセス許可モードの条件に合わない処理を実行しようとした。
300	データのタグ番号が不正である。
301	データのデータ長が不正である。
302	コンテントアクセス権を満たさない。
303	指定したデータバッファのメモリ不足。
304	指定したタグ番号のデータが存在しない。
310	複数個のデータ実体から構成されるデータの記述形式が不正である。
400	拡張ファイルアクセス異常。
600	CAMバージョンが一致しない。
601	CAMが用意しているバッファの容量不足。

注)伝送制御異常に関するリターン値は省略する。

3.2.10　icam_read_item_sequential

(1)呼出形式

short　icam_read_item_sequential(fd, tag, data, dlen, len)

(2)引き数

```
int fd;                 /*  ファイルディスクリプタ  */
int *tag;               /*  タグ  */
unsigned char *data;    /*  データ実体  */
int dlen;               /*  データバッファの長さ  */
int *len;               /*  データ長  */
```

(3)機能

icam_read_item_sequential関数は、icam_open_file関数によって既にオープンされている基礎ファイルから、リードポインタが指定するデータを読み出す。fdには該当する基礎ファイルに対応するファイルディスクリプタを設定する。dlenにはデータバッファdataの長さを設定する。本関数を実行した後、tagにはデータのタグ番号を、dataにはデータを、lenにはそのデータ長を設定する。

基礎ファイルのオープン直後、リードポインタは基礎ファイル内の先頭にあるデータを指定する。本関数は、データを読み出した後、リードポインタを次のデータに進める。icam_read_item関数は、リードポインタをicam_read_item関数で指定したデータに移動する。icam_write_item関数およびicam_erase_item関数は、リードポインタを対象となる基礎ファイルの先頭データに移動する。

CAMが該当する基礎ファイルのアクセスコントロールテーブルを持っている場合、アクセスコントロールテーブル内のコンテントアクセス権を参照し、Readが許可されている場合のみデータを読み出す。

CAMが該当する基礎ファイルのアクセスコントロールテーブルを持っていない場合、またはアクセスコントロールテーブルを持っているが、アクセスコントロールテーブル内に該当するタグのコンテントアクセス権が設定されていない場合、指定されたタグはReadが許可されているとみなし、データを読み出す。

複数個のデータ実体を持つデータを読み出す場合、data内のデータはICカードに記録されたフォーマットの形式で得られる。

本関数で読み出した後、icam_get_group_item関数を用いることによって、読み出されたデータが複数個のデータ実体を持つか否かを知ることができる。

また、icam_pick_item_data関数等を用いることによって、複数個のデータ実体を持つデータの各データ実体を読み出すことができる。

(4)実行条件

■対象となる基礎ファイルは、icam_open_file関数によって既にオープンされていなければならい。

■対象となる基礎ファイルは、アクセス許可モードReadOnlyまたはReadWiteでオープンされていなければならない。

■指定されたタグはコンテントアクセス権によってReadが許可されていなければならない。

■指定されたタグはCAMによりリザーブされているタグ「0x00」～「0x0f」でない。

(5)リターン値一覧

リターン値	意味
0	正常
100	ICカード内に指定したDFが存在しない。
101	ICカード内の指定したDFにDF情報ファイルが存在しない。
102	ICカード内の指定したDFに、指定した基礎ファイルが存在しない。
103	ICカードに対するアクセス権限を満たさない。
104	ICカードへのアクセス異常。
105	ICカードのコマンド異常。
106	ICカードのメモリ異常。
107	ファイルが閉塞している。
108	指定したレコードが存在しない。
202	指定したファイルディスクリプタは使用されていない。
203	指定したファイルディスクリプタは不正である。
209	オープン時のアクセス許可モードの条件に合わない処理を実行しようとした。
300	データのタグ番号が不正である。
301	データのデータ長が不正である。
302	コンテントアクセス権を満たさない。
303	指定したデータバッファのメモリ不足。
305	これ以上データが存在しない。
310	複数個のデータ実体から構成されるデータの記述形式が不正である。
400	拡張ファイルアクセス異常。
600	CAMバージョンが一致しない。
601	CAMが用意しているバッファの容量不足。

注)伝送制御異常に関するリターン値は省略する。

3.2.11　icam_reset_item_counter

(1)呼出形式

short　icam_reset_item_counter(fd)

(2)引き数

int fd;　　/*　ファイルディスクリプタ　*/

(3)機能

icam_reset_item_counter関数は、リードポインタを対象となる基礎ファイルの先頭データに移動する。fdには、対象となる基礎ファイルに対応するファイルディスクリプタを指定する。

(4)実行条件

■対象となる基礎ファイルは、icam_open_file関数によって既にオープンされていなければならい。

(5)リターン値一覧

リターン値	意味
0	正常
202	指定したファイルディスクリプタは使用されていない。
203	指定したファイルディスクリプタは不正である。

注)伝送制御異常に関するリターン値は省略する。

3.2.12　icam_get_file_size

(1)呼出形式

```
short    icam_get_file_size( fd, size )
```

(2)引き数

```
int fd;                  /*   ファイルディスクリプタ   */
int *size;               /*   ファイルサイズ   */
```

(3)機能

icam_get_file_size関数は、対象となる基礎ファイル内で使用されているデータ容量（単位：バイト）を取得する。fdには対象となる基礎ファイルに対応するファイルディスクリプタを指定する。本関数を実行した後、sizeには使用されているデータ容量を格納する。

使用されているデータ容量は、対象である基礎ファイルに記録されているデータ、すなわちタグ、データ長、データのバイト数を合計したものである。

また、対象である基礎ファイルが拡張ファイルを使用している場合は、拡張ファイルへの拡張を示すために基礎ファイル内に記録される最終データと拡張ファイル内にあるデータをさらに加える。ただし、拡張ファイル内にあるデータは、基礎ファイル内のデータのフォーマットにEF識別コードの長さを加える必要がある。

(4)実行条件

■対象となる基礎ファイルは、icam_open_file関数によって既にオープンされていなければならい。

■オープンされている基礎ファイルのアクセス許可モードが、ReadOnly、WriteOnly、ReadWriteの何れでも実行できなければならない。

(5)リターン値一覧

リターン値	意味
0	正常
202	指定したファイルディスクリプタは使用されていない。
203	指定したファイルディスクリプタは不正である。
400	拡張ファイルアクセス異常。
600	CAMバージョンが一致しない。
601	CAMが用意しているバッファの容量不足。

注)伝送制御異常に関するリターン値は省略する。

3.2.13　icam_erase_item

(1)呼出形式

short　icam_erase_item(fd, tag)

(2)引き数

int fd;　　/*　ファイルディスクリプタ　*/

int tag;　　/*　タグ　*/

(3)機能

icam_open_file関数によって既にオープンされている基礎ファイル内のtagにより指定されるデータを消去する。fdには該当する基礎ファイルに対応するファイルディスクリプタを設定する。tagには消去すべきデータのタグを設定する。

CAMが該当する基礎ファイルのアクセスコントロールテーブルを持っている場合、アクセスコントロールテーブル内から指定されたタグのコンテントアクセス権を参照し、Writeが許可されている場合のみデータを消去する。

CAMが該当する基礎ファイルのアクセスコントロールテーブルを持っていない場合、またはアクセスコントロールテーブルは持っているが、アクセスコントロールテーブル内に該当するタグのコンテントアクセス権が設定されていない場合、指定されたタグはWriteが許可されているとみなし、データを消去する。

ICカード内の基礎ファイルの内容を更新する場合、DF情報ファイル内の該当するEF情報も同時に更新する。ただし、本関数を実行する都度、ICカードへの書き込み処理を行うと処理速度の低下及びEEPROMの更新回数が増加するため、icam_close_file関数の実行時にまとめてICカードへの書き込み処理を行うようにしても良い。

(4)実行条件

■対象となる基礎ファイルは、icam_open_file関数によって既にオープンされていなければならい。

■対象となる基礎ファイルは、アクセス許可モードReadOnlyまたはReadWiteでオープンされていなければならない。

■指定されたタグはコンテントアクセス権によってWriteが許可されていなければならない。

■指定されたタグはCAMによりリザーブされているタグ「0x00」～「0x0f」でない。

(5)リターン値一覧

リターン値	意味
0	正常
100	ICカード内に指定したDFが存在しない。
101	ICカード内の指定したDFにDF情報ファイルが存在しない。
102	ICカード内の指定したDFに、指定した基礎ファイルが存在しない。
103	ICカードに対するアクセス権限を満たさない。
104	ICカードへのアクセス異常。
105	ICカードのコマンド異常。
106	ICカードのメモリ異常。
107	ファイルが閉塞している。
108	指定したレコードが存在しない。
202	指定したファイルディスクリプタは使用されていない。
203	指定したファイルディスクリプタは不正である。
209	オープン時のアクセス許可モードの条件に合わない処理を実行しようとした。
300	データのタグ番号が不正である。
301	データのデータ長が不正である。
302	コンテントアクセス権を満たさない。
304	指定したタグ番号のデータが存在しない。
400	拡張ファイルアクセス異常。
600	CAMバージョンが一致しない。
601	CAMが用意しているバッファの容量不足。

注)伝送制御異常に関するリターン値は省略する。

3.2.14　icam_erase_file

(1)呼出形式

short　icam_erase_file(fname, nlen, id)

(2)引き数

char *fname;　　/*　DF名　*/

int nlen;　　/*　DF名の長さ　*/

short id;　　/*　実EF-ID　*/

(3)機能

icam_erase_file関数は、 fname及びidで指定される基礎ファイル内の全データを消去する。fnameは16バイト以内のバイナリで表現されるDF名である。idは実EF-IDを示す。

対象である基礎ファイルが拡張ファイルを使用している場合、対象である基礎ファイルに属する拡張ファイル内の全データも消去する。

指定した基礎ファイル内にコンテントアクセス権によってReadまたはWriteが許可されていないデータが存在する場合には、該当するデータは消去せずに残す。なお、 DF情報に登録されたEF情報、及びタグ番号0x00の「CAMバージョン番号及びファイル領域サイズ」は消去しない。

(4)実行条件

■nlen≦16でなければならない。

■ICカード内にfnameで指定したDF及びidで指定した基礎ファイルが存在しなければならない。

■指定した基礎ファイルがicam_create_file関数によってCAMで利用可能でなければならない。

■指定したDFのDF情報ファイルのRead権及びWrite権が許可されていなければならない。

■指定した基礎ファイルのRead権及びWrite権が許可されていなければならない。

(5)リターン値一覧

リターン値	意味
0	正常
100	ICカード内に指定したDFが存在しない。
101	ICカード内の指定したDFにDF情報ファイルが存在しない。
102	ICカード内の指定したDFに、指定した基礎ファイルが存在しない。
103	ICカードに対するアクセス権限を満たさない。
104	ICカードへのアクセス異常。
105	ICカードのコマンド異常。
106	ICカードのメモリ異常。
107	ファイルが閉塞している。
108	指定したレコードが存在しない。
151	リーダ／ライタへのアクセス異常。
200	これ以上ファイルをオープンできない。
201	指定した基礎ファイルは既にオープンされている。
204	DF情報ファイルが定義外である。
205	DF情報ファイル内に指定した基礎ファイルのEF情報が存在しない。
206	指定した基礎ファイルが定義外である。
300	データのタグ番号が不正である。
301	データのデータ長が不正である。
302	コンテントアクセス権を満たさない。
303	指定したデータバッファのメモリ不足。
400	拡張ファイルアクセス異常。
600	CAMバージョンが一致しない。
601	CAMが用意しているバッファの容量不足。

注)伝送制御異常に関するリターン値は省略する。

3.2.15　icam_list_file

(1)呼出形式

```
short    icam_list_file( fname, nlen, list, n )
```

(2)引き数

```
char *fname;                /*   DF名    */
int nlen;                   /*   DF名の長さ    */
ICAM_FILE_LIST *list/*   n個のファイルリストを格納可能なサイズ    */
int *n;                     /*   基礎ファイル数    */
```

（注）ICAM_FILE_LISTのデータ構造

```
typedef struct ef_info        ICAM_FILE_LIST;
struct ef_info {
        unsigned char    ef;              /*   EF-ID    */
        unsigned char    type;            /*   EFのファイルタイプ    */
        struct datetype update;           /*   最新更新時間    */
        unsigned int       nb_rewrite;    /*   EFの更新回数    */
    };
struct datetype {
        short year;                       /*   最新更新時間：年    */
        char  month;                      /*                  月    */
        char    day;                      /*                  日    */
        char    hour;                     /*                  時    */
        char    mint;                     /*                  分    */
    };
```

(3)機能

icam_list_file関数は、16バイト以内のバイナリで表現されるfnameで指定したDFのDF情報ファイルから、そのDF内に存在する基礎ファイルであり、かつicam_create_file関数によってCAMで利用可能である全基礎ファイルのEF情報を取得し、ICAM_FILE_LISTで定義する構造体listに格納する。ポインタlistが指し示すデータ領域に取得するファイルリストの最大数を*nにファイル数単位で指定する。本関数は、実際に得たファイルリスト数をnが指し示すアドレスに返す。

(4)実行条件

■nlen≦16でなければならない。

■ICカード内にfnameで指定したDFが存在し、かつそのDF内にDF情報ファイルが存在しなければならない。

■指定したDFのDF情報ファイルのRead権が許可されていなければならない。

■listが指すデータ領域は、n個のファイルリストを格納することができるサイズが確保されていなければならない。

(5)リターン値一覧

リターン値	意味
0	正常
100	ICカード内に指定したDFが存在しない。
101	ICカード内の指定したDFにDF情報ファイルが存在しない。
102	ICカード内の指定したDFに、指定した基礎ファイルが存在しない。
103	ICカードに対するアクセス権限を満たさない。
104	ICカードへのアクセス異常。
105	ICカードのコマンド異常。
106	ICカードのメモリ異常。
107	ファイルが閉塞している。
108	指定したレコードが存在しない。
151	リーダ／ライタへのアクセス異常。
204	DF情報ファイルが定義外である。
600	CAMバージョンが一致しない。
601	CAMが用意しているバッファの容量不足。

注)伝送制御異常に関するリターン値は省略する。

3.2.16　icam_init_file

(1)呼出形式

short　icam_init_file(fname, nlen, id, recsize, recno, vs, dum, ext_flag)

(2)引き数

```
char *fname;          /*  DF名  */
int nlen;             /*  DF名の長さ  */
short id;             /*  実EF-ID  */
int recsize;          /*  レコードサイズ  */
int recno;            /*  レコード番号  */
int vs;               /*  CAMバージョン番号  */
int dum;              /*  利用せず  */
int ext_flag;         /*  拡張ファイルフラグ  */
```

(3)機能

基礎ファイルをCAM用に初期化する。fname と nlenで選択したDFを指定し、id で基礎ファイルを指定する。初期化する際の1レコードの大きさをrecsizeで、レコード数をrecnoで指定する。タグ0x00に書き込むバージョン番号をvsで指定する。拡張ファイルの場合には、ext_flagに1を、DF情報ファイルを含む一般基礎ファイルの場合には0を指定する。icam_create_file関数を実行する前に本関数でEFを初期化しておく必要がある。

(4)実行条件

■nlen≦16でなければならない。

■idで指定された基礎ファイルが、fnameで指定されたDFに存在していなければならない。

■fnameとidで指定された基礎ファイルにrecsizeで指定されたレコード長のレコードをrecnoで指定された数だけ書き込むことが可能でなければならない。

■拡張ファイルとして用いる基礎ファイルを初期化する際には、ext_flagに1を指定する。

(5)リターン値一覧

リターン値	意味
0	正常
100	ICカード内に指定したDFが存在しない。
102	ICカード内の指定したDFにDF情報ファイルが存在しない。
103	ICカードに対するアクセス権限を満たさない。
207	指定した基礎ファイルは既にクリエイトされている。
212	DF名またはDF名の長さが不正である。

注)伝送制御異常に関するリターン値は省略する。

3.2.17　icam_change_theoretical_id

(1)呼出形式

short　icam_change_theoretical_id(fname, nlen, id, tid)

(2)引き数

```
char *fname;        /*  DF名  */
int nlen;           /*  DF名の長さ  */
short id;           /*  実EF-ID  */
short tid;          /*  論理的なEF-ID  */
```

(3)機能

既に存在する循環型基礎ファイルの論理的なEF-IDを変更する。

(4)実行条件

■nlen≦16でなければならない。

■CAM　Version2.0で初期化された基礎ファイルのみ実行可能とする。CAM　Version2.0では、JIS準拠ICカードを対象としているため、実及び論理的なEF-IDは0x01から0x1dまでの値のみ使用可能である。

(5)リターン値一覧

リターン値	意味
0	正常
101	ICカード内に指定したDFが存在しない。
201	指定した基礎ファイルは既にオープンされている。
204	DF情報ファイルが定義外である。
205	DF情報ファイル内に指定した基礎ファイルのEF情報ファイルが存在しない。
600	CAMバージョンが一致しない。

注)伝送制御異常に関するリターン値は省略する。

3.2.18　icam_get_theoretical_id

(1)呼出形式

```
short　icam_get_theoretical_id( fname, nlen, id, tid )
```

(2)引き数

```
char *fname;          /*　DF名　*/
int nlen;             /*　DF名の長さ　*/
short id;             /*　実EF-ID　*/
short *tid;           /*　論理的なEF-ID　*/
```

(3)機能

実EF-IDから論理的なEF-IDを得る。

(4)実行条件

■nlen≦16でなければならない。

■CAM　Version2.0で初期化された基礎ファイルのみ実行可能とする。CAM　Version2.0では、JIS準拠ICカードを対象としているため、実及び論理的なEF-IDは0x01から0x1dまでの値のみ使用可能である。

(5)リターン値一覧

リターン値	意味
0	正常
101	ICカード内に指定したDFが存在しない。
204	DF情報ファイルが定義外である。
210	循環型基礎ファイルでない。
600	CAMバージョンが一致しない。

注)伝送制御異常に関するリターン値は省略する。

3.2.19　icam_get_actual_id

(1)呼出形式

short　icam_get_actual_id(fname, nlen, tid, odr, id)

(2)引き数

```
char *fname;          /*  DF名  */
int nlen;             /*  DF名の長さ  */
short tid;            /*  論理的なEF-ID  */
int odr;              /*  循環型基礎ファイル番号  */
short *id;            /*  実EF-ID（戻り値）  */
```

(3)機能

論理的なEF-IDから実EF-IDを得る。基礎ファイルは、論理的なEF-IDと更新時間の順序を示す循環型基礎ファイル番号で指定する。指定方法は、icam_open_file関数と同様である。

(4)実行条件

■nlen≦16でなければならない。

■CAM　Version2.0で初期化された基礎ファイルのみ実行可能とする。CAM　Version2.0では、JIS準拠ICカードを対象としているため、実及び論理的なEF-IDは0x01から0x1dまでの値のみ使用可能である。

(5)リターン値一覧

リターン値	意味
0	正常
101	ICカード内に指定したDFが存在しない。
204	DF情報ファイルが定義外である。
210	循環型基礎ファイルでない。
600	CAMバージョンが一致しない。

注)伝送制御異常に関するリターン値は省略する。

3.2.20　icam_count_file_number

(1)呼出形式

```
short    icam_count_file_number( fname, nlen, tid, nbr )
```

(2)引き数

```
char *fname;          /*  DF名  */
int nlen;             /*  DF名の長さ  */
short tid;            /*  論理的なEF-ID  */
int *nbr;             /*  ファイル数（戻り値）  */
```

(3)機能

論理的なEF-IDが同一の循環型基礎ファイル数を得る。

(4)実行条件

■nlen≦16でなければならない。

■CAM Version2.0で初期化された基礎ファイルのみ実行可能とする。

(5)リターン値一覧

リターン値	意味
0	正常
101	ICカード内に指定したDFが存在しない。
204	DF情報ファイルが定義外である。
210	循環型基礎ファイルでない。

注)伝送制御異常に関するリターン値は省略する。

3.2.21　icam_get_update_info

(1)呼出形式

```
short    icam_get_update_info( fname, nlen, id, tm )
```

(2)引き数

```
char *fname;              /*   DF名   */
int nlen;                 /*   DF名の長さ   */
unsigned char id;         /*   実EF-ID   */
struct datetype *tm;      /*   更新日時（戻り値）   */
```

(3)機能

循環型基礎ファイルの更新日時を得る。

(4)実行条件

■nlen≦16でなければならない。

■循環型基礎ファイルが存在していなければならない。

(5)リターン値一覧

リターン値	意味
0	正常
101	ICカード内に指定したDFが存在しない。
204	DF情報ファイルが定義外である。
205	DF情報ファイル内に指定した基礎ファイルのEF情報が存在しない。

注)伝送制御異常に関するリターン値は省略する。

3.2.22　icam_get_group_item

(1)呼出形式

```
short　icam_get_group_item( fd, tag, flag )
```

(2)引き数

```
int fd;                 /*　ファイルディスクリプタ　*/
int tag;                /*　タグ　*/
int *flag;              /*　読み込まれたデータのデータ実体数を判定するフラグ　*/
```

(3)機能

icam_read_item関数やicam_read_item_sequential関数の直後において本関数を用いることによって、読み込まれたデータが複数個のデータ実体を持つタグか否かを知ることが可能である。直前に実行する icam_read_item関数または icam_read_item_sequential関数で複数個のデータ実体から構成されるデータが読み出された場合には*flagとして1を返し、また唯一のデータ実体のデータである場合には*flagとして0を返す。

(4)実行条件

■ファイルディスクリプタfdとタグtagは、直前に読み込まれたファイルディスクリプタおよびタグと一致していなければならない。

(5)リターン値一覧

リターン値	意味
0	正常
202	指定したファイルディスクリプタは使用されていない。
203	指定したファイルディスクリプタは不正である。
300	データのタグ番号が不正である。
311	直前に読み込んだデータと異なるファイルディスクリプタまたはタグ番号を指定した。または、直前に読み込みは行われていない。

注)伝送制御異常に関するリターン値は省略する。

3.2.23　icam_set_group_item

(1)呼出形式

short　icam_set_group_item(fd, tag, flag)

(2)引き数

int fd;　　/*　ファイルディスクリプタ　*/

int tag;　　/*　タグ　*/

int flag;　　/*　読み込まれたデータのデータ実体数を判定するフラグ　*/

(3)機能

icam_write_item関数の直前において本関数を用いることによって、書き込むデータが複数個のデータ実体を持つタグか否かを指定することが可能である。複数個のデータ実体から構成されるデータを書き込む場合には *flag =1として本関数をコールする。また、唯一のデータ実体のデータを書き込む場合には *flag =0として本関数をコールするか、または本関数を用いない。

(4)実行条件

■ファイルディスクリプタfdとタグtagは、直後に書き込むデータのファイルディスクリプタおよびタグと一致していなければならない。

(5)リターン値一覧

リターン値	意味
0	正常
202	指定したファイルディスクリプタは使用されていない。
203	指定したファイルディスクリプタは不正である。
300	データのタグ番号が不正である。

注)伝送制御異常に関するリターン値は省略する。

3.2.24　icam_pick_item_data

(1)呼出形式

```
short    icam_pick_item_data( idat, data, dlen, len, cnt )
```

(2)引き数

```
unsigned char *idat;      /*  複数個のデータ実体を持つデータへのポインタ  */
unsigned char *data;      /*  データ  */
int dlen;                 /*  データバッファ data の長さ  */
int *len;                 /*  データバッファ data のデータ長  */
int cnt;                  /*  読み込むデータ実体の番号  */
```

(3)機能

icam_read_item関数またはicam_read_item_sequential関数で読み込まれた単数または複数個のデータ実体を持つデータidatを解析し、データ内で先頭からcnt番目のデータ実体をデータバッファdataに返す。

また、*lenには実際に読み込まれたデータ長がバイト単位で返される。

(4)実行条件

■複数個のデータ実体を持つデータへのポインタidatは、icam_read_item関数またはicam_read_item_sequential関数で読み出される複数個のデータ実体を持つデータ形式のアレイとする。

■データバッファdataは複数個のデータ実体を持つデータの各データ実体を格納するのに十分な長さdlenを持っていなければならない。

■複数個のデータ実体を持つデータは、データの最後尾に0x0が付加された形式とする。従って、複数個のデータ実体を持つデータを扱うデータバッファのサイズとして、少なくとも複数個のデータ実体を持つデータ自身のデータサイズ+1バイトが確保されている必要がある。

(5)リターン値一覧

リターン値	意味
0	正常
303	指定したデータバッファのメモリ不足。
310	複数個のデータ実体を持つデータの形式が不正である。
312	cntで指定した番号のデータが存在しない。

注)伝送制御異常に関するリターン値は省略する。

3.2.25　icam_append_item_data

(1)呼出形式

short　icam_append_item_data(idat, data, ilen, len)

(2)引き数

unsigned char *idat;　　/*　複数個のデータ実体を持つデータへのポインタ　*/

unsigned char *data;　　/*　データ　*/

int ilen;　　/*　データバッファidat の長さ　*/

int len;　　/*　データバッファ data のデータ長　*/

(3)機能

データdataを複数個のデータ実体を持つデータidatに追加し、icam_write_item関数により書き込むことが可能な形式に変換する。データバッファdataのデータ長 len にはデータバッファ　idata　に確保された領域の長さをバイト単位で指定する。データバッファidatのデータ長 ilen には複数個のデータ実体を持つデータ idat の長さを指定する。

(4)実行条件

■複数個のデータ実体を持つデータへのポインタidatは、icam_write_item関数で書き込まれる複数個のデータ実体を持つデータ形式のアレイとする。新たにidatを作成する場合には0埋めにより初期化しておくことが必要である。

■データバッファidatは複数個のデータ実体を持つデータに、本関数によりデータを追加された後の全データ実体を格納するのに十分な長さilenを持っていなければならない。

■複数個のデータ実体を持つデータは、データの最後尾に0x0が付加された形式とする。従って、複数個のデータ実体を持つデータを扱うデータバッファのサイズとして、少なくとも複数個のデータ実体を持つデータ自身のデータサイズ+1バイトが確保されている必要がある。

(5)リターン値一覧

リターン値	意味
0	正常
303	指定したデータバッファのメモリ不足。
310	複数個のデータ実体を持つデータの形式が不正である。

注)伝送制御異常に関するリターン値は省略する。

3.2.26　icam_remove_item_data

(1)呼出形式

```
short    icam_remove_item_data( idat, cnt )
```

(2)引き数

```
unsigned char *idat;        /*   複数個のデータ実体を持つデータへのポインタ   */
int cnt;                    /*   削除するデータ実体の番号   */
```

(3)機能

複数個のデータ実体を持つデータ idat 内の先頭から cnt 番目のデータ実体を削除する。

(4)実行条件

■複数個のデータ実体を持つデータへのポインタidatは、icam_write_item関数で書き込まれる複数個のデータ実体を持つデータ形式のアレイとする。

■データバッファidatは少なくとも cnt 個のデータ実体を持っていなければならない。

■複数個のデータ実体を持つデータは、データの最後尾に0x0が付加された形式とする。従って、複数個のデータ実体を持つデータを扱うデータバッファのサイズとして、少なくとも複数個のデータ実体を持つデータ自身のデータサイズ+1バイトが確保されている必要がある。

(5)リターン値一覧

リターン値	意味
0	正常
310	複数個のデータ実体を持つデータの形式が不正である。
312	cnt で指定した番号のデータが存在しない。

注)伝送制御異常に関するリターン値は省略する。

3.2.27　icam_count_item_data

(1)呼出形式

```
short    icam_count_item_data( idat, cnt )
```

(2)引き数

```
unsigned char *idat;        /*   複数個のデータ実体を持つデータへのポインタ   */
int *cnt;                   /*   データ内データ実体の総数   */
```

(3)機能

複数個のデータ実体を持つデータへのポインタidat内に記録されているデータ実体の個数を*cntに返す。

(4)実行条件

■複数個のデータ実体を持つデータへのポインタidatは、icam_write_item関数で書き込まれる複数個のデータ実体を持つデータ形式のアレイとする。

■複数個のデータ実体を持つデータは、データの最後尾に0x0が付加された形式とする。従って、複数個のデータ実体を持つデータを扱うデータバッファのサイズとして、少なくとも複数個のデータ実体を持つデータ自身のデータサイズ+1バイトが確保されている必要がある。

(5)リターン値一覧

リターン値	意味
0	正常
310	複数個のデータ実体を持つデータの形式が不正である。

注)伝送制御異常に関するリターン値は省略する。

3.2.28　icam_item_len

(1)呼出形式

short　icam_item_len(idat)

(2)引き数

unsigned char *idat;　　/*　複数個のデータ実体を持つデータへのポインタ　*/

(3)機能

複数個のデータ実体を持つデータへのポインタidatの全体長を返値として返す。

(4)実行条件

■複数個のデータ実体を持つデータへのポインタidatは、icam_write_item関数で書き込まれる複数個のデータ実体を持つデータ形式のアレイとする。

■複数個のデータ実体を持つデータは、データの最後尾に0x0が付加された形式とする。従って、複数個のデータ実体を持つデータを扱うデータバッファのサイズとして、少なくとも複数個のデータ実体を持つデータ自身のデータサイズ+1バイトが確保されている必要がある。

(5)リターン値一覧

リターン値	意味
0	正常
-1	複数個のデータ実体を持つデータの形式が不正である。

注)伝送制御異常に関するリターン値は省略する。

3.2.29　icam_erase_all_item

(1)呼出形式

int　icam_erase_all_item(fd)

(2)引き数

int fd;　　/*　ファイルディスクリプタ　*/

(3)機能

icam_erase_all_item関数は、icam_erase_file関数とほぼ同一の機能を有するが、fname及びidで基礎ファイルを指定するのではなく、オープンされている基礎ファイルに対してファイルディスクリプタfdで基礎ファイルを指定することにより、対象基礎ファイル内の全データを消去する。

対象である基礎ファイルが拡張ファイルを使用している場合、対象である基礎ファイルに属する拡張ファイル内の全データも消去する。指定した基礎ファイル内にコンテントアクセス権によってReadまたはWriteが許可されていないデータが存在する場合には、該当するデータは消去せずに残す。なお、DF情報に登録されたEF情報、及びタグ番号0x00の「CAMバージョン番号及びファイル領域サイズ」は消去しない。

(4)実行条件

■指定した基礎ファイルはicam_open_file関数によりオープンされていなければならない。

■指定した基礎ファイルがicam_create_file関数によってCAMで利用可能でなければならない。

■指定したDFのDF情報ファイルのRead権及びWrite権が許可されていなければならない。

■指定した基礎ファイルのRead権及びWrite権が許可されていなければならない。

(5)リターン値一覧

リターン値	意味
0	正常
300	データのタグ番号が不正である。
301	データのデータ長が不正である。
302	コンテントアクセス権を満たさない。
303	指定したデータバッファのメモリ不足。
400	拡張ファイルアクセス異常。
600	CAMバージョンが一致しない。
601	CAMが用意しているバッファの容量不足。

注)伝送制御異常に関するリターン値は省略する。

3.2.30　icam_get_error_info

(1)呼出形式

```
void　icam_get_error_info( loc_info, status_info, command_info )
```

(2)引き数

```
unsigned int *loc_info;　　/*　エラーの位置情報　*/
unsigned int *status_info;　/*　カード又はリーダ・ライタのステータス情報　*/
unsigned int *command_info; /*　カード又はリーダ・ライタに発行したコマンド情報　*/
```

(3)機能

CAM内部のエラー情報を取得する。

(4)リターン値一覧

無し

3.2.31　icam_get_file_info

(1)呼出形式

int　icam_get_file_info(fd, file_info)

(2)引き数

int fd;　　/*　ファイルディスクリプタ　*/

ICAM_FILE_INFO *file_info;　　/*　EF構造体番号　*/

(3)機能

　オープンしている基礎ファイルの情報を取得する。fdにより基礎ファイルを指定し、file_infoに基礎ファイルの情報が格納される。

(4)リターン値一覧

リターン値	意味
0	正常
202	指定したファイルディスクリプタは使用されていない。
203	指定したファイルディスクリプタは不正である。

注)伝送制御異常に関するリターン値は省略する。

3.2.32　icam_get_df_info

(1)呼出形式

int　icam_get_df_info (dfno, dfinfo)

(2)引き数

int dfno;　　/*　DFディスクリプタ　*/

ICAM_DF_INFO *dfinfo;　　/*　DF構造体番号　*/

(3)機能

　オープンしているDF情報ファイルの情報を取得する。DFディスクリプタにより指定し、dfinfoにDF情報ファイルの情報が格納される。

(4)リターン値一覧

リターン値	意味
0	正常
101	ICカード内の指定したDFにDF情報ファイルが存在しない。

注)伝送制御異常に関するリターン値は省略する。

3.2.33　icam_identify_operation

(1)呼出形式

```
int　icam_identify_operation( pins, n_pin, status )
```

(2)引き数

```
char *pins;                /*　キーデータ　*/
int n_pin;                 /*　キーの個数　*/
unsigned short *status     /*　リーダ／ライタまたはICカードのステータス　*/
```

(3)機能

icam_identify_operation関数は、オペレーションICカード内にある所有者情報ファイル、キーテーブルファイル、アクセスコントロールテーブルファイルから、それぞれ所有者情報、キーテーブル、アクセスコントロールテーブルを読み出し、以下の2つのアクセスコントロール機能を実現するための初期設定を行う。

■キーテーブル内のキーデータに基づくアクセスコントロール機能

ICカードが持つアクセスコントロール機能を利用して実現される。オペレーションカード毎にキーテーブル内のキーデータを変えることにより、同一データカード内の同一基礎ファイルであっても、それに対するアクセス権限を変えることが可能となる。

■コンテントアクセス権に基づくデータ単位のアクセスコントロール機能

アクセスコントロールテーブルに設定されるコンテントアクセス権に基づいて、データ単位のアクセス権限の設定が可能である。オペレーションカード毎にアクセスコントロールテーブルに設定するコンテントアクセス権を変えることにより、同一データカード内の同一基礎ファイルに存在する同一データであっても、そのアクセス権限を変えることが可能となる。

pinsにはキーデータを格納するバッファの先頭アドレスを設定するが、本関数を実行する前に、キーデータを格納するバッファにオペレーションカード内の各基礎ファイルにアクセスするための必要なキーまたはPINを以下のフォーマットで格納する必要がある。

また、n_pinにはそのバッファ内にあるキーの個数を指定する。statusには本関数を実行した結果得られるリーダ／ライタまたはICカードのレスポンスデータが格納される。

本関数は、オペレーションカード内のDF名「OP-DATA」に格納されているDFから所有者情報ファイル、キーテーブルファイル、アクセスコントロールテーブルファイルを検索する。これらのファイルおよびDF名「OP-DATA」のDF情報ファイルをアクセスするために必要なキーまたはPINはpins内に指定する。また、pins内に指定されるキーまた

はPIN以外に、CAM自身がいくつかのキーまたはPINを持つことを許容する。

本関数は、オペレーションICカードのリーダ／ライタへの挿入から排出までを行う。本関数でCAMが取得したオペレーションカードの所有者情報、キーテーブル、アクセスコントロールテーブルは、icam_terminate_operation関数が実行されるまで保持する。

表2.4.3　キーテーブル情報のフォーマット

先頭からバイト数	バイト数	データ形式	情報の種類
0	1	バイナリ	KID1
1	1	バイナリ	キー長
2	X	バイナリ	キー実体
2+X	1	バイナリ	KID2
3+X	1	バイナリ	キー長
4+X	Y	バイナリ	キー実体
〜	〜	〜	〜

(4)実行条件

■オペレーションカード内にDF名「OP-DATA」が存在し、かつDF情報ファイルが存在しなければならない。

■DF情報ファイルのRead権限を有さなければならない。

■DF名「OP-DATA」内に、所有者情報ファイル、キーテーブルファイル、アクセスコントロールテーブルファイルが存在する場合、それらのRead権限を有さなければならない。

■DF名「OP-DATA」内に、所有者情報ファイル、キーテーブルファイル、アクセスコントロールテーブルファイルの内の一部または全部が存在しなくても良い。

(5)リターン値一覧

リターン値	意味
0	正常
100	ICカード内に指定したDFが存在しない。
101	ICカード内の指定したDFにDF情報ファイルが存在しない。
102	ICカード内の指定したDFに、指定したEFが存在しない。
103	ICカードに対するアクセス権限を満たさない。
104	ICカードへのアクセス異常。
105	ICカードのコマンド異常。
106	ICカードのメモリ異常。
107	ファイルが閉塞している。
108	指定したレコードが存在しない。
150	指定した時間内にICカードがリーダ／ライタに挿入されなかった。
151	リーダ／ライタへのアクセス異常。
200	これ以上ファイルをオープンできない。
201	指定した基礎ファイルは既にオープンされている。
204	DF情報ファイルが定義外である。
205	DF情報ファイル内に指定した基礎ファイルのEF情報が存在しない。
206	指定した基礎ファイルが定義外である。
301	データのデータ長が不正である。
302	コンテントアクセス権を満たさない。
303	指定したデータバッファのメモリ不足。
400	拡張ファイルアクセス異常。
500	オペレーションカードの情報は既に読み込まれている。
502	オペレーションカードにアクセスするためのキー情報のパラメータ異常。
503	オペレーションカードのフォーマットエラー。
600	CAMバージョンが一致しない。
601	CAMが用意しているバッファの容量不足。

注)伝送制御異常に関するリターン値は省略する。

3.2.34　icam_terminate_operation

(1)呼出形式

int　icam_terminate_operation(void)

(2)引き数

無し

(3)機能

icam_terminate_operation関数は、icam_identify_operation関数で取得したオペレーションカードの所有者情報、キーテーブル、アクセスコントロールテーブルを消去する。

(4)リターン値一覧

リターン値	意味
0	正常
501	オペレーションカードの情報は、読み込まれていない。

注)伝送制御異常に関するリターン値は省略する。

3.2.35　icam_get_operation_info

(1)呼出形式

int　icam_operation_info(op_name, op_info)

(2)引き数

char *op_name;　　/*　オペレーションカード所有者名　*/

char *op_info;　　/*　オペレーションカード所有者情報　*/

(3)機能

使用しているオペレーションカード内の所有者情報を所得する。現在では、「0x10：所有者名」「0x11：所有者情報」としている。

(4)リターン値一覧

リターン値	意味
0	正常
-1	所有者情報が存在しない。

注)伝送制御異常に関するリターン値は省略する。

3.2.36　icam_verify_pin

(1)呼出形式

int　icam_verify_pin(fname, nlen, kid, kdata, klen)

(2)引き数

```
char *fname;          /*  DF名  */
int nlen;             /*  DF名の長さ  */
short kid;            /*  キーID  */
char *kdata;          /*  キーデータ  */
int klen;             /*  キーデータ長  */
```

(3)機能

DFを選択した後にPINを照合する場合には、nlenにfnameで指定する。nlenに0を指定した場合には、DFの選択を行わずにPINを照合する。照合するPINのID、内容、長さはそれぞれ kid, kdata, klen で指定する。

(4)実行条件

■nlen≦16でなければならない。

(5)リターン値一覧

リターン値	意味
0	正常
100	ICカード内に指定したDFが存在しない。

注)伝送制御異常に関するリターン値は省略する。

3.2.37　リターン値

(1)リターン値の分類

CAM　Version2.0のアプリケーション・インタフェース関数のリターン値を以下の通り分類する。

リターン値	意味
0	正常
1XX	ICカードおよびリーダ／ライタへのアクセス時の異常
2XX	ファイル制御に関する異常
3XX	アイテム制御に関する異常
4XX	拡張ファイル制御に関する異常
5XX	オペレーションカード制御に関する異常
6XX	上記以外の異常（CAMバージョンの相違、メモリ不足）
負数	ICカードおよびリーダ／ライタとの伝送制御に関する異常

注)リターン値は10進数表示とする。また、Xは0～9の任意の数を意味する。

(2)ICカードおよびリーダ／ライタからのステータスコード

CAM　Version2.0のアプリケーション・インタフェース関数の中には、ICカードあるいはリーダ／ライタとの間で、複数のコマンド／レスポンスを交換する関数があり、リターン値からではICカードあるいはリーダ／ライタへのどのコマンドで、どのような異常が生じたかを、上位のアプリケーションプログラムは知ることができない。

従って、CAM　Version2.0のアプリケーション・インタフェース関数では、2つの変数command(unsigned int)、status(unsigned int)を用意し、リターン値が異常である場合に、ICカードあるいはリーダ／ライタに対するどのコマンドにおいて異常が生じたかを上位のアプリケーションプログラムが把握できる手段を提供する。

command、statusには、それぞれ異常なリターン値（0以外のリターン値）の発生要因となったICカードあるいはリーダ／ライタへのコマンドコード、ステータスコードを格納する。アプリケーション・インタフェース関数がICカードあるいはリーダ／ライタにアクセスしない場合や、ICカードあるいはリーダ／ライタとの伝送制御上に異常があった場合には、command、statusには0を格納する。

(3)リターン値一覧

アプリケーション・インタフェース関数のリターン値一覧を以下に示す。

リターン値	意　味（上段：概要　下段：補足説明）
0	正常
100	ICカード内に指定したDFが存在しない。
	SELECT ADFでエラーステータス3000が返った場合。
101	ICカード内の指定したDFにDF情報ファイルが存在しない。
	DF情報ファイルへのREAD RECORD等でエラーステータス3000が返った場合。
102	ICカード内の指定したDFに、指定した基礎ファイルが存在しない。
	指定した基礎ファイルへのREAD RECORD等でエラーステータス3000が返った場合。
103	ICカードに対するアクセス権限を満たさない。
	ファイルオープン時のREAD権限およびWRITE権限の確認処理において、エラーステータス4000が検出された場合。その他、エラーステータス4000を検出した場合は、すべてこのリターン値を返す（拡張ファイルへのアクセス時も含む）。
104	ICカードへのアクセス異常。
	CAMが想定していないエラーステータスがICカードから返された場合にこのリターン値を返す。ICカードに重大な異常が生じたと考えられる。
105	ICカードのコマンド異常。
	上記のリターン値「104」を補足するために設定するリターン値であり、Lc/Leフィールドが間違っている、指定された論理チャネルを未サポートである、コマンドがファイル構造と矛盾する、カレントEFがない、等のパラメータ及びデータ設定に関するエラーステータスがICカードから返された場合にこのリターン値を返す。
106	ICカードのメモリ異常。
	上記のリターン値「104」を補足するために設定するリターン値であり、出力データに異常がある、ファイル制御情報に異常がある、ファイルへの書き込みが失敗等に関するエラーステータスがICカードから返された場合にこのリターン値を返す。
107	ファイルが閉塞している。
	上記のリターン値「104」を補足するために設定するリターン値であり、参照されたDFまたは基礎ファイルが閉塞している等に関するエラーステータスがICカードから返された場合にこのリターン値を返す。
108	指定したレコードが存在しない。
	上記のリターン値「104」を補足するために設定するリターン値であり、指定したレコードが存在しない等に関するエラーステータスがICカードから返された場合にこのリターン値を返す。
109	ベリファイに失敗した。
	ベリファイに失敗した。
110	サポートされていない種類のカードが挿入された。
	サポートされていない種類のカードが挿入された。

リターン値	意　味（上段：概要　下段：補足説明）
150	指定した時間内にICカードがリーダ／ライタに挿入されなかった。 ドライバ関数からのリターン値が-41（cn送信異常終結）の時、このリターン値に置き換える。
151	リーダ／ライタへのアクセス異常 リーダ／ライタへのコマンドに対してエラーステータス（0000以外）が返された。
200	これ以上ファイルをオープンできない。 オープンできるファイル数を越えてファイル数をオープンしようとした。オープン可能なファイル数は、CAMの製造者毎に異なっていても良い。
201	指定した基礎ファイルは既にオープンされている。 オープンされている基礎ファイルを二重にオープンすることはできない。
202	指定したファイルディスクリプタは使用されていない。 オープン処理によって、割り当てられていないファイルディスクリプタや既にクローズしているファイルディスクリプタを指定して、icam_read_item関数等を実行した場合に、このリターン値を返す。
203	指定したファイルディスクリプタは不正である。 オープン可能な基礎ファイルの最大設定数を越える値や負の値をファイルディスクリプタとして指定して、icam_read_item関数等を実行した場合に、このリターン値を返す。
204	DF情報ファイルが定義外である。 DF情報ファイルは存在するが、その中にレコードが存在しない場合やDF情報として定義されていないデータが検出された場合に、このリターン値を返す。
205	DF情報ファイル内に指定した基礎ファイルのEF情報が存在しない。 指定した基礎ファイルをオープンする時に、DF情報ファイル（EF－ID=0x01）内に指定した基礎ファイルのEF情報が存在しない場合に、このリターン値を返す。
206	指定した基礎ファイルが定義外である。 指定した基礎ファイルは存在するが、その中にレコードがない場合や定義されていないデータが検出された場合にこのリターン値を返す。
207	指定した基礎ファイルは既にクリエイトされている。 DF情報ファイル内に指定した基礎ファイルのEF情報が既に存在する。
208	DF情報ファイルあるいは指定した基礎ファイルにこれ以上データを追加できない。 拡張ファイルに対してもこれ以上データを追加できない場合に、このリターン値を返す。
209	オープン時のアクセス許可モードの条件に合わない処理を実行しようとした。 ReadOnlyでオープン時に、データのライト／イレーズを実行しようとした場合、およびWriteOnlyでオープン時に、データのリードを実行しようとした場合に、このリターン値を返す。

リターン値	意　味（上段：概要　下段：補足説明）
210	循環型基礎ファイルでない。 指定した基礎ファイルが循環型基礎ファイルでない。
211	指定した順番の循環型基礎ファイルは存在しない。 指定した順番の循環型基礎ファイルは存在しない。
212	DF名またはDF名の長さが不正である。 DF名がバイナリで記述されいない、またはDF名の長さが16バイトを越えている場合に、このリターン値を返す。
220	DF情報が不正である。 DF情報の記述形式、記録内容等が正しくない場合に、このリターン値を返す。
300	データのタグ番号が不正である。 リザーブされているタグ番号（0x00～0x0F）や定義外のタグ番号のデータをリード／ライト／イレーズする場合に、このリターン値を返す。
301	データのデータ長が不正である。 データ長が定義外のデータをリード／ライト／イレーズしようとした。
302	コンテントアクセス権を満たさない。 コンテントアクセス権を満たさないデータをリード／ライト／イレーズしようとした。
303	指定したデータバッファのメモリ不足。 データのリード系のアプリケーション・インタフェース関数において、指定したデータバッファよりもデータのサイズが大きい。
304	指定したタグ番号のデータが存在しない。 データのリードまたはイレーズを実行しようとした時に、指定したタグ番号が存在しない。
305	これ以上データが存在しない。 データの順次読み出しを実行しようとした時に、読み出すべきデータが存在しない。
310	複数個のデータ実体を持つデータの形式は不正である。 複数個のデータ実体を持つデータのフォーマットが正しくない。
311	直前に読み込んだデータと異なるファイルディスクリプタまたはタグ番号を指定した。または、直前に読み込みは行われていない。 直前に読み込んだデータと異なるファイルディスクリプタまたはタグ番号を指定した。または、直前に読み込みは行われていない。
312	cntで指定した番号のデータが存在しない。 cntで指定した番号のデータが存在しない。

リターン値	意　味（上段：概要　下段：補足説明）
400	拡張ファイルアクセス異常。
	以下のような場合に、このリターン値を返す。このリターン値は、拡張ファイル制御に関して重大な異常が生じたことを通知する。 ●DF情報ファイル内に拡張ファイルとして指定されているDFが存在しない。 ●DF情報ファイル内に拡張ファイルとして指定されているDFに基礎ファイルが存在しない。 ●拡張ファイル用のDF内にDF情報ファイルがない ●拡張ファイル用のDF情報ファイル内に定義外のデータがある。 ●拡張ファイル用のDF情報ファイル内で示される基礎ファイルが存在しない。 ●拡張ファイル用のDF情報ファイルで示される基礎ファイル内に定義外のデータがある。 注意：拡張ファイルのREAD権限あるいはWRITE権限を満たさないことを検知した場合は、リターン値104とする。
401	アクセス中のICカードは拡張ファイルをサポートしていない。
	通常のDF情報ファイルあるいは通常の基礎ファイルの容量が足りず、拡張ファイルを利用しようとした時に、アクセス中のICカードが拡張ファイルをサポートしていない場合に、このリターン値を返す。
500	オペレーションカードの情報は既に読み込まれている。
	オペレーションカードの所有者情報、アクセスコントロールテーブル、キーテーブルが既にロードされている。再ロードするためには、icam_terminate_operation関数を実行する必要がある。
501	オペレーションカードの情報は、読み込まれていない。
	オペレーションカードの情報が読み込まれていないにもかかわらず、解放しようとした。
502	オペレーションカードにアクセスするためのキー情報のパラメータ異常。
	icam_identify_operation関数で指定したキー情報が定義外である。
503	オペレーションカードのフォーマットエラー。
	オペレーションカード内のキーテーブル、アクセスコントローールファイルに定義外のフォーマットでデータが記憶されている。
600	CAMバージョンが一致しない。
	基礎ファイル内のタグ番号00に示されるCAMバージョンと動作中のCAMバージョンが異なる。
601	CAMが用意しているバッファの容量不足。
	CAM内部で利用しているバッファ容量が不足している。致命的エラー。
602	アロケーションエラーである。
	メモリ上でアロケーションエラーが発生した。

リターン値	意　味（上段：概要　下段：補足説明）
-1	複数個のデータ実体を持つデータの形式が不正である。
	複数個のデータ実体が記録されているデータのフォーマットが正しくないことを検出した。
-10	リーダ／ライタの電源がOFFである。または、ケーブルが接続されていない。
	リーダ／ライタや接続ケーブル等に起因するハード的な要因による異常を検出した。
-22	ブロックシーケンス番号異常。
	ICカードあるいはリーダ／ライタとの伝送制御に関するブロックの交換においてブロックシーケンス番号の交互性に異常を検出した。
-31	"en"受信異常終結。
	ICカードあるいはリーダ／ライタから、"en"ブロックを受信した。
-32	"cn"受信異常終結。
	ICカードあるいはリーダ／ライタから、"cn"ブロックを受信した。
-33	BWタイムアウト。
	ICカードあるいはリーダ／ライタから所定時間内にレスポンスブロックが返ってこない。
-40	"en"送信異常終結。
	ICカードあるいはリーダ／ライタに対し"en"ブロックを送信した。
-41	"cn"送信異常終結。
	ICカードあるいはリーダ／ライタに対し"cn"ブロックを送信した。
-51	[-]（チェーン無）ブロック受信シーケンス異常。
	ICカードとの通信においてブロック連鎖の処理中に、[-]（チェーン無）ブロックを受信した。
-100	CWがタイムアウトである。
	CWがタイムアウトである。
-200	その他のエラーが生じた。
	その他のエラーが生じた。

3.3　デバイスドライバ共通インタフェース関数仕様

3.3.1　std_com_open

(1)呼出形式

```
short    std_com_open( port, snad, slen, spec )
```

(2)引き数

```
unsigned char port;       /*   通信ポート指定   */
unsigned char snad;       /*   ホストノードアドレス（default:2）    */
unsigned short slen;      /*   情報フィールド長整数値（IFSD）    */
unsigned char *spec;      /*   対応ICカード提示   */
```

(3)機能仕様

RS-232C等の通信ポートの初期設定を行うと共に、ICカード及びICカードリーダ・ライタとの通信を行うための伝送制御パラメータの初期設定を行う。

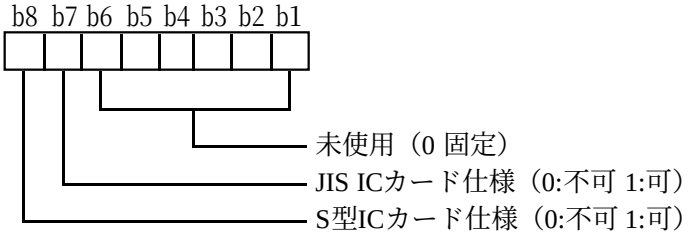
■パラメータ仕様

・指定可能な通信ポート番号（port）は、0～255とする。

・指定可能なホストノードアドレス（snad）は、0～7とする。

・指定可能な情報フィールド長（IFSD）は、1～256とする。

・対応ICカード提示バイト（spec）は以下に従う。



■対応ICカードの提示

・本ドライバが対応可能なICカードの種類を変数 spec に設定する。

■デバイス初期設定

・使用する機種に対応した通信ポート（RS-232C等）に関する初期設定を行う。

■伝送制御初期設定

・上位より指定されたICカードに関する伝送プロトコル上の初期設定等を行う。

(4)備考

■デバイスの初期設定の詳細は、使用する機種に依存するため、本関数仕様では規定し

ない。

■伝送制御上の初期設定は、ICカード及びICカードリーダ・ライタとの通信を制御するためのデバイスドライバ関数のプログラム構造や内部関数仕様等に依存するため、本関数仕様では規定しない。

(5)リターン値一覧

リターン値	意味
0	正常
-10	通信ポートのオープン異常。
-11	通信ポートは既にオープンされている。
-15	パラメータ設定エラー。

3.3.2　std_com_close

(1)呼出形式

short　std_com_close(port)

(2)引き数

unsigned char port;　　/*　通信ポート指定　*/

(3)機能仕様

RS-232C等の通信ポートをクローズする。

■パラメータ仕様

・指定可能な通信ポート番号（port）は、0～255とする。

■デバイスのクローズ

・使用する機種に対応した通信ポート（RS-232C等）に関するクローズ処理を行う。

(4)備考

■デバイスのクローズ処理の詳細は、使用する機種に依存するため、本関数仕様では規定しない。

(5)リターン値一覧

リターン値	意味
0	正常
-10	通信ポートのオープン異常。
-11	通信ポートは既にオープンされている。
-15	パラメータ設定エラー。

3.3.3　std_card_in

(1)呼出形式

short　std_card_in(port, wtime, spec, rlen, rbufr)

(2)引き数

unsigned char port;　　/*　通信ポート指定　*/

unsigned short wtime;　/*　カード挿入待ち時間（秒単位）　*/

unsigned char *spec;　　/*　カード仕様識別情報　*/

unsigned short *rlen;　　/*　初期応答情報の長さ　*/

unsigned char *rbufr;　　/*　初期応答情報の先頭アドレス　*/

(3)機能仕様

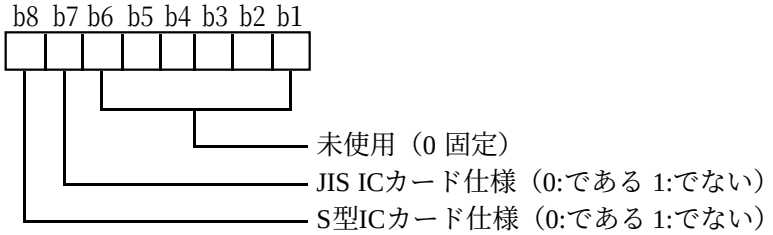
ICカードの挿入促進及び活性化を行うコマンドをリーダ・ライタに要求する。リーダ・ライタによってICカードの活性化が行われた場合、初期応答情報を取得、解析し、ICカードとの通信を行う上で必要な伝送制御パラメータを設定すると共に、活性化されたICカードの仕様を識別する。

■パラメータ仕様

・指定可能な通信ポート番号（port）は、0～255とする。

・指定可能なカード挿入待ち時間（wtime）は、1～600秒とし、0が指定された場合は、無限に待つものとする。

・カード仕様識別情報（spec）は以下に従う。



■ICカードの活性化

・ICカードの活性化処理は、リーダ・ライタが行う。本関数は、リーダ・ライタにICカードの活性化を要求する。

■初期応答情報の解析処理

・活性化によって得られた初期応答情報中に伝送プロトコルとして T=14 と指定されていれば、S型ICカードであると判断し、 T=1 が指定されていれば、JIS ICカードであると判断する。判断結果は、カード仕様識別情報（spec）に示す。S型ICカード、JIS ICカードの何れでもない場合、 spec=00h である。

(4)備考

■ICカードの活性化要求は、リーダ・ライタのコマンドによって実現される。しかし、このようなコマンドは、使用するリーダ・ライタの仕様に依存するため、ここでは、詳細を規定しない。

■初期応答情報の解析手順は、JIS X6304に従うものとする。

(5)リターン値一覧

リターン値	意味
0	正常
-10	通信ポートのオープン異常。
-12	通信ポートがオープンされていない。
-15	パラメータ設定エラー。
-20	ブロックシーケンス番号異常。
-30	プロトコル異常。
-31	プロトコル異常。（ブロック連鎖異常）
-40	BWタイムアウト。
-41	プロトコル異常、あるいは、指定した時間内にICカードがリーダ・ライタに挿入されなかった。
-60	初期応答情報を取得できない。（活性化失敗）
-61	対応できない初期応答情報が設定されている。

3.3.4　std_card_out

(1)呼出形式

short　std_card_out(port)

(2)引き数

unsigned char port;　　/*　通信ポート指定　*/

(3)機能仕様

ICカードを排出するコマンドをリーダ・ライタに要求する。

■パラメータ仕様

・指定可能な通信ポート番号（port）は、0～255とする。

■ICカードの非活性化

・ICカードを排出するコマンドをリーダ・ライタに要求する。

(4)備考

■ICカードの排出要求は、リーダ・ライタのコマンドによって実現される。しかし、このようなコマンドは、使用するリーダ・ライタの仕様に依存するため、ここでは、詳細を規定しない。

■初期応答情報の解析手順は、JIS X6304に従うものとする。

(5)リターン値一覧

リターン値	意味
0	正常
-10	通信ポートのオープン異常。
-12	通信ポートがオープンされていない。
-15	パラメータ設定エラー。
-20	ブロックシーケンス番号異常。
-30	プロトコル異常。
-31	プロトコル異常。（ブロック連鎖異常）
-40	BWタイムアウト。
-62	カードを排出できない。（カード詰まり）

3.3.5　std_card_reset

(1)呼出形式

short　std_card_reset(port, spec, rlen, rbufr)

(2)引き数

unsigned char port;　　/*　通信ポート指定　*/

unsigned char *spec;　　/*　カード仕様識別情報　*/

unsigned short *rlen;　　/*　初期応答情報の長さ　*/

unsigned char *rbufr;　　/*　初期応答情報の先頭アドレス　*/

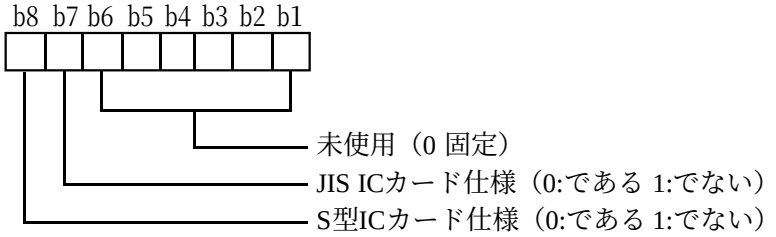
(3)機能仕様

ICカードがリーダ・ライタに挿入されているときに、ICカードの再活性化を行う。また、再活性化により得られた初期応答情報を、解析し、ICカードとの通信を行う上で必要な伝送制御パラメータを設定すると共に、再活性化されたICカードの仕様を識別する。

■パラメータ仕様

・指定可能な通信ポート番号（port）は、0～255とする。

・カード仕様識別情報（spec）は以下に従う。



■ICカードの再活性化

・ICカードの再活性化処理は、リーダ・ライタが行う。本関数は、リーダ・ライタにICカードの再活性化を要求する。

■初期応答情報の解析処理

・活性化によって得られた初期応答情報中に伝送プロトコルとして T=14 と指定されていれば、S型ICカードであると判断し、 T=1 が指定されていれば、JIS ICカードであると判断する。判断結果は、カード仕様識別情報（spec）に示す。S型ICカード、JIS ICカードの何れでもない場合、 spec=00h である。

(4)備考

■ICカードの再活性化要求は、リーダ・ライタのコマンドによって実現される。しかし、このようなコマンドは、使用するリーダ・ライタの仕様に依存するため、ここでは、

詳細を規定しない。

■初期応答情報の解析手順は、JIS X6304に従うものとする。

(5)リターン値一覧

リターン値	意味
0	正常
-10	通信ポートのオープン異常。
-12	通信ポートがオープンされていない。
-15	パラメータ設定エラー。
-20	ブロックシーケンス番号異常。
-30	プロトコル異常。
-31	プロトコル異常。（ブロック連鎖異常）
-40	BWタイムアウト。
-60	初期応答情報を取得できない。（活性化失敗）
-61	対応できない初期応答情報が設定されている。
-63	カードがリーダ・ライタに挿入されていない。

3.3.6　std_card_sense

(1)呼出形式

```
short    std_card_sense( port, flag )
```

(2)引き数

```
unsigned char port;        /*   通信ポート指定   */
unsigned char *flag;       /*   リーダ・ライタ内のカード所在情報   */
```

(3)機能仕様

ICカードがリーダ・ライタに挿入されているか否かを確認する。

■パラメータ仕様

・指定可能な通信ポート番号（port）は、0～255とする。

・カード所在情報（flag）は以下に従う。

0x00	カード挿入無し。
0x04	カードがリーダ・ライタのスロット内に存在する。 （半挿入状態を含む。カードは活性化されていない。）
上記以外	カードがリーダ・ライタのスロット内に存在する。 （カードが活性化されている。）

■ICカードの所在情報

・リーダ・ライタに対して、ICカードの所在情報を要求する。

(4)備考

■リーダ・ライタがICカードの所在確認機能をサポートしていない場合は、その旨をリターン値にて通知する。

(5)リターン値一覧

リターン値	意味
0	正常
-10	通信ポートのオープン異常。
-12	通信ポートがオープンされていない。
-15	パラメータ設定エラー。
-20	ブロックシーケンス番号異常。
-30	プロトコル異常。
-31	プロトコル異常。（ブロック連鎖異常）
-40	BWタイムアウト。
-60	初期応答情報を取得できない。（活性化失敗）
-61	対応できない初期応答情報が設定されている。
-64	リーダ・ライタがカードの所在確認機能をサポートしていない。

3.3.7　std_select_DF

(1)呼出形式

short　std_select_DF(port, typ, flen, file, slen, sbufr, rlen, rbufr)

(2)引き数

```
unsigned char port;      /*  通信ポート指定  */
unsigned char typ;       /*  0:SELECT FILE （DF名による選択）  */
                         /*  1:SELECT FILE （EF-IDによる選択　現在未使用）  */
                         /*  2:SELECT ADF （DF名による選択）  */
unsigned char *flen;     /*  DF名長  */
unsigned char *file;     /*  DF名のポインタ  */
unsigned short slen;     /*  送信:information部の長さ  */
unsigned char *sbufr;    /*  送信:information部の先頭アドレス  */
unsigned short *rlen;    /*  送信:information部の長さ  */
unsigned char *rbufr;    /*  送信:information部の先頭アドレス  */
```

(3)機能仕様

SELECT FILE（JIS X6306）、あるいはSELECT ADF（S型ICカード仕様）コマンドの何れかを実行する。

！ SELECT FILE が指定された場合

■コマンド実行

・SELECT FILE（JIS X6306）コマンドの information 部を作成、実行し、そのレスポンスを取得する。

・指定されたコマンドに対応していない場合、その旨のエラー通知を行う。

■パラメータ仕様

・指定可能な通信ポート番号（port）は、0～255とする。

・DF名長（flen）は、1～16とする。

・DF名のポインタ（file）は、バイナリコードとする。

・送信及び受信における information 部の長さは、1～65535バイトとする。

■レスポンスフォーマット（受信 information 部の構成）

SW1	SW2
（1）	（1）

（）内バイト数

" SELECT ADF が指定された場合

■コマンド実行

・SELECT ADF（S型ICカード）コマンドの information 部を作成、実行し、そのレスポンスを取得する。

・指定されたコマンドに対応していない場合、その旨のエラー通知を行う。

■パラメータ仕様

・指定可能な通信ポート番号（port）は、0～255とする。

・DF名長（flen）は、1～8とする。

・DF名のポインタ（file）は、バイナリコードとする。

・送信及び受信における information 部の長さは、1～65535バイトとする。

■レスポンスフォーマット（受信 information 部の構成）

制御コード	STS1	STS2
(2)	(1)	(1)

() 内バイト数

(4)備考

■本関数においてどのコマンドをサポートするかは、製造者が任意に決めて良い。

■引数 slen 及び sbufr は、デバイスドライバに通信用バッファの情報を渡すだけであり、送信 information 部の内容は任意である。

(5)リターン値一覧

リターン値	意味
0	正常
-10	通信ポートのオープン異常。
-12	通信ポートがオープンされていない。
-15	パラメータ設定エラー。
-16	指定されたコマンドはサポートされていない。
-20	ブロックシーケンス番号異常。
-30	プロトコル異常。
-31	プロトコル異常。（ブロック連鎖異常）
-40	BWタイムアウト。

3.3.8　std_verify

(1)呼出形式

short　std_verify(port, typ, efid, klen, ldata, , slen, sbufr, rlen, rbufr)

(2)引き数

```
unsigned char port;      /*  通信ポート指定  */
unsigned char typ;       /*  0:VERIFY（JIS X6306）  */
                         /*  1:VERIFY KEY（S型ICカード）  */
unsigned char efid;      /*  EF-ID  */
unsigned char klen;      /*  キーの長さ  */
unsigned char *kdata;    /*  キー情報へのポインタ  */
unsigned short slen;     /*  送信:information部の長さ  */
unsigned char *sbufr;    /*  送信:information部の先頭アドレス  */
unsigned short *rlen;    /*  送信:information部の長さ  */
unsigned char *rbufr;    /*  送信:information部の先頭アドレス  */
```

(3)機能仕様

VERIFY（JIS X6306）、あるいはVERIFY KEY（S型ICカード仕様）コマンドの何れかを実行する。

! VERIFY が指定された場合

■コマンド実行

・VERIFY（JIS X6306）コマンドの information 部を作成、実行し、そのレスポンスを取得する。

・指定されたコマンドに対応していない場合、その旨のエラー通知を行う。

■パラメータ仕様

・指定可能な通信ポート番号（port）は、0～255とする。

・EF-ID（efid）は、1～30とする。ただし、0はカレントEFの選択のため留保する。

・キーの長さ（klen）は、1～16とする。

・キー情報へのポインタ（kdata）は、バイナリコードとする。

・送信及び受信における information 部の長さは、1～65535バイトとする。

■レスポンスフォーマット（受信 information 部の構成）

SW1	SW2
(1)	(1)

() 内バイト数

" VERIFY KEY が指定された場合

■コマンド実行

・VERIFY KEY（S型ICカード）コマンドの information 部を作成、実行し、そのレスポンスを取得する。

・指定されたコマンドに対応していない場合、その旨のエラー通知を行う。

■パラメータ仕様

・指定可能な通信ポート番号（port）は、0～255とする。

・EF-ID（efid）は、00h～7fhとする。ただし、0はカレントEFの選択のため留保する。

・キーの長さ（klen）は、1～16とする。

・キー情報へのポインタ（kdata）は、バイナリコードとする。

・送信及び受信における information 部の長さは、1～65535バイトとする。

■レスポンスフォーマット（受信 information 部の構成）

制御コード	STS1	STS2
(2)	(1)	(1)

() 内バイト数

(4)備考

■本関数においてどのコマンドをサポートするかは、製造者が任意に決めて良い。

■引数 slen 及び sbufr は、デバイスドライバに通信用バッファの情報を渡すだけであり、送信 information 部の内容は任意である。

(5)リターン値一覧

リターン値	意味
0	正常
-10	通信ポートのオープン異常。
-12	通信ポートがオープンされていない。
-15	パラメータ設定エラー。
-16	指定されたコマンドはサポートされていない。
-20	ブロックシーケンス番号異常。
-30	プロトコル異常。
-31	プロトコル異常。（ブロック連鎖異常）
-40	BWタイムアウト。

3.3.9　std_read_record

(1)呼出形式

short　std_read_record(port, typ, efid, recno, slen, sbufr, rlen, rbufr)

(2)引き数

```
unsigned char port;      /*  通信ポート指定  */
unsigned char typ;       /*  0:READ RECORD(S)（JIS X6306）1レコード指定読み出し  */
                         /*  1:READ RECORD(S)（JIS X6306）複数レコード読み出し  */
                         /*  2:READ RECORD（S型ICカード）  */
unsigned char efid;      /*  EF-ID  */
unsigned char recno;     /*  レコード番号  */
unsigned short slen;     /*  送信:information部の長さ  */
unsigned char *sbufr;    /*  送信:information部の先頭アドレス  */
unsigned short *rlen;    /*  送信:information部の長さ  */
unsigned char *rbufr;    /*  送信:information部の先頭アドレス  */
```

(3)機能仕様

READ RECORD(S)（JIS X6306）、あるいはREAD RECORD（S型ICカード仕様）コマンドの何れかを実行する。

！ READ RECORD(S) が指定された場合

■コマンド実行

・READ RECORD(S)（JIS X6306）コマンドの information 部を作成、実行し、そのレスポンスを取得する。

・指定されたコマンドに対応していない場合、その旨のエラー通知を行う。

■パラメータ仕様

・指定可能な通信ポート番号（port）は、0～255とする。

・EF-ID（efid）は、1～30とする。ただし、0はカレントEFの選択のため留保する。

・レコード番号（recno）は、1～255とする。

・送信及び受信における information 部の長さは、1～65535バイトとする。

■レスポンスフォーマット（受信 information 部の構成）

データフィールド	SW1	SW2
（可変）	(1)	(1)

() 内バイト数

" READ RECORD が指定された場合

■コマンド実行

・READ RECORD（S型ICカード）コマンドの information 部を作成、実行し、そのレスポンスを取得する。

・指定されたコマンドに対応していない場合、その旨のエラー通知を行う。

■パラメータ仕様

・指定可能な通信ポート番号（port）は、0～255とする。

・EF-ID（efid）は、00h～7fhとする。ただし、0はカレントEFの選択のため留保する。

・レコード番号（recno）は、0～255とする。ただし、0はカレントレコード読み出しとする。

・送信及び受信における information 部の長さは、1～65535バイトとする。

■レスポンスフォーマット（受信 information 部の構成）

制御コード	STS1	STS2	長さ	レコード
(2)	(1)	(1)	(2)	(可変)

() 内バイト数

(4)備考

■本関数においてどのコマンドをサポートするかは、製造者が任意に決めて良い。

■引数 slen 及び sbufr は、デバイスドライバに通信用バッファの情報を渡すだけであり、送信 information 部の内容は任意である。

(5)リターン値一覧

リターン値	意味
0	正常
-10	通信ポートのオープン異常。
-12	通信ポートがオープンされていない。
-15	パラメータ設定エラー。
-16	指定されたコマンドはサポートされていない。
-20	ブロックシーケンス番号異常。
-30	プロトコル異常。
-31	プロトコル異常。（ブロック連鎖異常）
-40	BWタイムアウト。

3.3.10　std_write_record

(1)呼出形式

short　std_write_record(port, typ, efid, recno, dlen, dbufr, slen, sbufr, rlen, rbufr)

(2)引き数

```
unsigned char port;      /*  通信ポート指定  */
unsigned char typ;       /*  0:WRITE RECORD（JIS X6306）  */
                         /*  1:WRITE RECORD（S型ICカード）  */
unsigned char efid;      /*  EF-ID  */
unsigned char recno;     /*  レコード番号  */
unsigned short dlen;     /*  書き込みデータの長さ  */
unsigned char *dbufr;    /*  書き込みデータへのポインタ  */
unsigned short slen;     /*  送信:information部の長さ  */
unsigned char *sbufr;    /*  送信:information部の先頭アドレス  */
unsigned short *rlen;    /*  送信:information部の長さ  */
unsigned char *rbufr;    /*  送信:information部の先頭アドレス  */
```

(3)機能仕様

WRITE RECORD（JIS X6306）、あるいはWRITE RECORD（S型ICカード仕様）コマンドの何れかを実行する。

！ WRITE RECORD（JIS X6306）が指定された場合

■コマンド実行

・WRITE RECORD（JIS X6306）コマンドの information 部を作成、実行し、そのレスポンスを取得する。

・指定されたコマンドに対応していない場合、その旨のエラー通知を行う。

■パラメータ仕様

・指定可能な通信ポート番号（port）は、0～255とする。

・EF-ID（efid）は、1～30とする。ただし、0はカレントEFの選択のため留保する。

・レコード番号（recno）は、1～255とする。

・書き込みデータの長さ（dlen）は、1～65535とする。

・送信及び受信における information 部の長さは、1～65535バイトとする。

■レスポンスフォーマット（受信 information 部の構成）

SW1	SW2
(1)	(1)

() 内バイト数

" WRITE RECORD（S型ICカード）が指定された場合

■コマンド実行

・WRITE RECORD（S型ICカード）コマンドの information 部を作成、実行し、そのレスポンスを取得する。

・指定されたコマンドに対応していない場合、その旨のエラー通知を行う。

■パラメータ仕様

・指定可能な通信ポート番号（port）は、0～255とする。

・EF-ID（efid）は、00h～7fhとする。

・レコード番号（recno）は、0～255とする。ただし、0は追記書き込みとする。

・書き込みデータの長さ（dlen）は、1～65535とする。

・送信及び受信における information 部の長さは、1～65535バイトとする。

■レスポンスフォーマット（受信 information 部の構成）

制御コード	STS1	STS2
(2)	(1)	(1)

() 内バイト数

(4)備考

■本関数においてどのコマンドをサポートするかは、製造者が任意に決めて良い。

■引数 slen 及び sbufr は、デバイスドライバに通信用バッファの情報を渡すだけであり、送信 information 部の内容は任意である。

(5)リターン値一覧

リターン値	意味
0	正常
-10	通信ポートのオープン異常。
-12	通信ポートがオープンされていない。
-15	パラメータ設定エラー。
-16	指定されたコマンドはサポートされていない。
-20	ブロックシーケンス番号異常。
-30	プロトコル異常。
-31	プロトコル異常。（ブロック連鎖異常）
-40	BWタイムアウト。

3.3.11　std_append_record

(1)呼出形式

short　std_append_record(port, typ, efid, recno, dlen, dbufr, slen, sbufr, rlen, rbufr)

(2)引き数

```
unsigned char port;      /*  通信ポート指定  */
unsigned char typ;       /*  0:APPEND RECORD（JIS X6306）  */
                         /*  0以外は将来のために留保する  */
unsigned char efid;      /*  EF-ID  */
unsigned char recno;     /*  レコード番号（0固定）  */
unsigned short dlen;     /*  書き込みデータの長さ  */
unsigned char *dbufr;    /*  書き込みデータへのポインタ  */
unsigned short slen;     /*  送信:information部の長さ  */
unsigned char *sbufr;    /*  送信:information部の先頭アドレス  */
unsigned short *rlen;    /*  送信:information部の長さ  */
unsigned char *rbufr;    /*  送信:information部の先頭アドレス  */
```

(3)機能仕様

APPEND RECORD（JIS X6306）コマンドを実行する。

■コマンド実行

・APPEND RECORD（JIS X6306）コマンドの information 部を作成、実行し、そのレスポンスを取得する。

・指定されたコマンドに対応していない場合、その旨のエラー通知を行う。

■パラメータ仕様

・指定可能な通信ポート番号（port）は、0～255とする。

・EF-ID（efid）は、1～30とする。ただし、0はカレントEFの選択のため留保する。

・レコード番号（recno）は、1～255とする。

・書き込みデータの長さ（dlen）は、1～65535とする。

・送信及び受信における information 部の長さは、1～65535バイトとする。

■レスポンスフォーマット（受信 information 部の構成）

SW1	SW2
(1)	(1)

() 内バイト数

(4)備考

■本関数においてどのコマンドをサポートするかは、製造者が任意に決めて良い。

■引数 slen 及び sbufr は、デバイスドライバに通信用バッファの情報を渡すだけであり、送信 information 部の内容は任意である。

(5)リターン値一覧

リターン値	意味
0	正常
-10	通信ポートのオープン異常。
-12	通信ポートがオープンされていない。
-15	パラメータ設定エラー。
-16	指定されたコマンドはサポートされていない。
-20	ブロックシーケンス番号異常。
-30	プロトコル異常。
-31	プロトコル異常。（ブロック連鎖異常）
-40	BWタイムアウト。

3.3.12　std_update_record

(1)呼出形式

short　std_update_record(port, typ, efid, recno, dlen, dbufr, slen, sbufr, rlen, rbufr)

(2)引き数

```
unsigned char port;      /*  通信ポート指定  */
unsigned char typ;       /*  0:UPDATE RECORD（JIS X6306）  */
                         /*  0以外は将来のために留保する  */
unsigned char efid;      /*  EF-ID  */
unsigned char recno;     /*  レコード番号  */
unsigned short dlen;     /*  書き込みデータの長さ  */
unsigned char *dbufr;    /*  書き込みデータへのポインタ  */
unsigned short slen;     /*  送信:information部の長さ  */
unsigned char *sbufr;    /*  送信:information部の先頭アドレス  */
unsigned short *rlen;    /*  送信:information部の長さ  */
unsigned char *rbufr;    /*  送信:information部の先頭アドレス  */
```

(3)機能仕様

UPDATE RECORD（JIS X6306）コマンドを実行する。

■コマンド実行

・UPDATE RECORD（JIS X6306）コマンドの information 部を作成、実行し、そのレスポンスを取得する。

・指定されたコマンドに対応していない場合、その旨のエラー通知を行う。

■パラメータ仕様

・指定可能な通信ポート番号（port）は、0～255とする。

・EF-ID（efid）は、1～30とする。ただし、0はカレントEFの選択のため留保する。

・レコード番号（recno）は、1～255とする。ただし、0はカレントレコード指定を意味する。

・書き込みデータの長さ（dlen）は、1～65535とする。

・送信及び受信における information 部の長さは、1～65535バイトとする。

■レスポンスフォーマット（受信 information 部の構成）

SW1	SW2
(1)	(1)

（）内バイト数

(4)備考

■本関数においてどのコマンドをサポートするかは、製造者が任意に決めて良い。

■引数 slen 及び sbufr は、デバイスドライバに通信用バッファの情報を渡すだけであり、送信 information 部の内容は任意である。

(5)リターン値一覧

リターン値	意味
0	正常
-10	通信ポートのオープン異常。
-12	通信ポートがオープンされていない。
-15	パラメータ設定エラー。
-16	指定されたコマンドはサポートされていない。
-20	ブロックシーケンス番号異常。
-30	プロトコル異常。
-31	プロトコル異常。（ブロック連鎖異常）
-40	BWタイムアウト。

3.3.13　std_erase_all_records

(1)呼出形式

short　std_erase_all_records(port, typ, efid, slen, sbufr, rlen, rbufr)

(2)引き数

```
unsigned char port;       /*  通信ポート指定  */
unsigned char typ;        /*  0:ERASE ALL RECORDS（JICSAP滝川市向け仕様）  */
                          /*  1:ERASE ALL RECORDS（S型ICカード）  */
unsigned char efid;       /*  EF-ID  */
unsigned short slen;      /*  送信:information部の長さ  */
unsigned char *sbufr;     /*  送信:information部の先頭アドレス  */
unsigned short *rlen;     /*  送信:information部の長さ  */
unsigned char *rbufr;     /*  送信:information部の先頭アドレス  */
```

(3)機能仕様

ERASE ALL RECORDS（JICSAP滝川市向け仕様）、あるいはERASE ALL RECORDS（S型ICカード仕様）コマンドの何れかを実行する。

! ERASE ALL RECORDS（JICSAP滝川市向け仕様）が指定された場合

■コマンド実行

・ERASE ALL RECORDS（JICSAP滝川市向け仕様）コマンドの information 部を作成、実行し、そのレスポンスを取得する。

・指定されたコマンドに対応していない場合、その旨のエラー通知を行う。

■パラメータ仕様

・指定可能な通信ポート番号（port）は、0～255とする。

・EF-ID（efid）は、1～30とする。ただし、0はカレントEFの選択のため留保する。

・送信及び受信における information 部の長さは、1～65535バイトとする。

■レスポンスフォーマット（受信 information 部の構成）

SW1	SW2
（1）	（1）

（）内バイト数

" ERASE ALL RECORDS（S型ICカード）が指定された場合

■コマンド実行

・ERASE ALL RECORDS（S型ICカード）コマンドの information 部を作成、実行し、そのレスポンスを取得する。

・指定されたコマンドに対応していない場合、その旨のエラー通知を行う。

■パラメータ仕様

・指定可能な通信ポート番号（port）は、0～255とする。

・EF-ID（efid）は、00h～7fhとする。

・送信及び受信における information 部の長さは、1～65535バイトとする。

■レスポンスフォーマット（受信 information 部の構成）

制御コード	STS1	STS2
（2）	（1）	（1）

（）内バイト数

(4)備考

■本関数においてどのコマンドをサポートするかは、製造者が任意に決めて良い。

■引数 slen 及び sbufr は、デバイスドライバに通信用バッファの情報を渡すだけであり、送信 information 部の内容は任意である。

(5)リターン値一覧

リターン値	意味
0	正常
-10	通信ポートのオープン異常。
-12	通信ポートがオープンされていない。
-15	パラメータ設定エラー。
-16	指定されたコマンドはサポートされていない。
-20	ブロックシーケンス番号異常。
-30	プロトコル異常。
-31	プロトコル異常。（ブロック連鎖異常）
-40	BWタイムアウト。

3.3.14　std_other_command

(1)呼出形式

short　std_other_command(port, typ, slen, sbufr, rlen, rbufr)

(2)引き数

```
unsigned char port;      /*  通信ポート指定  */
unsigned char typ;       /*  0:T=1 伝送プロトコルを実行する  */
                         /*  1:T=14 伝送プロトコルを実行する  */
unsigned short slen;     /*  送信:information部の長さ  */
unsigned char *sbufr;    /*  送信:information部の先頭アドレス  */
unsigned short *rlen;    /*  送信:information部の長さ  */
unsigned char *rbufr;    /*  送信:information部の先頭アドレス  */
```

(3)機能仕様

本仕様において、明示的に示されていないICカードコマンドを実行するときに使用する。コマンドの information 部はこの関数の呼び出し側で作成する。

! T=1 が指定された場合

■コマンド実行

・この関数の呼び出し側によって設定された information 部に基づいて、コマンドブロックを構成し、ICカードに送信する。

■パラメータ仕様

・指定可能な通信ポート番号（port）は、0～255とする。

・送信及び受信における information 部の長さは、1～65535バイトとする。

■レスポンスフォーマット（受信 information 部の構成）

任意	SW1	SW2
（任意）	（1）	（1）

（）内バイト数

" T=14 が指定された場合

■コマンド実行

・この関数の呼び出し側によって設定された information 部に基づいて、コマンドブロックを構成し、ICカードに送信する。

■パラメータ仕様

・指定可能な通信ポート番号（port）は、0～255とする。

・送信及び受信における information 部の長さは、1～65535バイトとする。

■レスポンスフォーマット（受信 information 部の構成）

制御コード	STS1	STS2	任意
(2)	(1)	(1)	(任意)

() 内バイト数

(4)備考

■本関数を使用することにより、ICカード内のデータの互換性が確保されなくなる可能性が生じる。従って、本関数を使用するにあたっては、対象となるICカードの仕様及び対象となるICカードの利用範囲を十分に考慮する必要がある。

■引数 slen 及び sbufr は、デバイスドライバに通信用バッファの情報を渡すだけであり、送信 information 部の内容は任意である。

(5)リターン値一覧

リターン値	意味
0	正常
-10	通信ポートのオープン異常。
-12	通信ポートがオープンされていない。
-15	パラメータ設定エラー。
-16	指定されたコマンドはサポートされていない。
-20	ブロックシーケンス番号異常。
-30	プロトコル異常。
-31	プロトコル異常。（ブロック連鎖異常）
-40	BWタイムアウト。

3.3.15　リターン値

(1)リターン値の分類

CAM　Version2.0のデバイスドライバ共通インタフェース関数のリターン値を以下の通り分類する。

リターン値	意味
0	正常
-1X	通信ポート・コマンド制御等に関する異常
-2X	ブロックシーケンス等に関する異常
-3X	通信プロトコルに関する異常
-4X	リーダ・ライタ操作等に関する異常
-6X	初期応答情報に関する異常
その他	上記以外の異常

注)リターン値は10進数表示とする。また、Xは0～9の任意の数を意味する。

(2)リターン値一覧

CAM　Version2.0のデバイスドライバ共通インタフェース関数のリターン値一覧を以下に示す。

リターン値	意　味
0	正常
-10	通信ポートのオープン異常。
-11	通信ポートは既にオープンされている。
-12	通信ポートがオープンされていない。
-15	パラメータ設定エラー。
-16	指定されたコマンドはサポートされていない。
-20	ブロックシーケンス番号異常。
-30	プロトコル異常。
-31	プロトコル異常。（ブロック連鎖異常）
-40	BWタイムアウト
-41	指定した時間内にICカードがリーダ・ライタに挿入されなかった。
-60	初期応答情報を取得できない。（活性化失敗）
-61	対応できない初期応答情報が設定されている。
-62	カードを排出できない。（カード詰まり）
-63	カードがリーダ・ライタに挿入されていない。
-64	リーダ・ライタがカードの所在確認機能をサポートしていない。

4. 内容アクセスマネージャを効果的に利用するための環境整備

4.1　背景及び目的

将来の高度情報通信社会の到来に向けて、行政・保健・医療・福祉等の公共的な業種・分野を所轄する行政機関は、国民への情報提供の高度化、諸手続きの迅速化・簡略化等を始めとした公共分野サービスの質的向上をより一層推進するために、様々な公共的な業種・分野を対象に外部端子付ICカード、非接触ICカード等の高機能なカードメディアを導入していくと予想される。従来からの事務処理・手続き等を踏襲するならば、行政機関は各機関毎に利用者である国民にカードメディアを配布していくことになるが、この場合に国民は各機関毎に発行された複数枚のカードメディアを目的・用途毎に選択して使い分けなければならず、使い勝手の面で非常に煩雑になり、延いては高機能なカードメディアを導入する一つの重要な意義が失われる可能性がある。国民に対する配慮を最優先する観点に立つならば、行政機関は各種の公共分野サービスに用いるカードメディアのあり方・位置づけ等について十分に議論していくことが極めて重要であると考えられる。

従って、ここでは、カード所有者である国民が公共分野に限らず複数の様々なアプリケーションサービスを一枚のカードメディアによって広域的にかつ共通的に享受できる利用環境を健全にかつ円滑に実現していくために必要不可欠となるであろう基本方針、作業手順、整備項目等についてタグ標準化ワーキンググループにて議論された概要を以下に示す。

4.2　前提条件

4.2.1　対象カードメディア

ここでは、現在、既に実用化されているPETカード、磁気ストライプカード、外部端子付ICカード等を始めとした様々なカードメディアのうち、1チップ化されたCPU及び記憶メモリを搭載するカードメディアである外部端子付ICカード及び外部端子を持たない非接触型ICカード（以下、総称してCPU付ICカードという）を対象とする。

4.2.2　対象利用用途

CPU付ICカードは、その機能的な特長からセキュリティ機能を持つデータキャリアとしての利用用途とカード所有者及びアプリケーション端末に対する認証手段としての利用用途に

大別され、アプリケーションサービス提供者はその導入目的や利用形態等により何れかの利用用途を選択することが一般的である。

セキュリティ機能を持つデータキャリアとしての利用用途では、データを記録する形式として、CPU付ICカード内の記憶メモリに様々なデータ項目に対するデータ実体を書き込む形式とデータ実体が記録されているアドレス情報のみを書き込む形式が用いられる。前者は、主としてスタンドアローン形態により提供されるアプリケーションサービスを対象としたデータ記録形式である一方、後者は、ネットワーク形態により提供されるアプリケーションサービスを対象としたデータ記録形式である。前者は記録可能容量等の点でデータに関する対象・範囲が制限される懸念があるが、後者はこれらの制約条件等をある程度軽減することが可能であるといえる。

一方、後者としての利用用途では、後述するような点で現存するCPU付ICカードの認証機能が不十分である可能性を含んでおり、早急な対応の必要性が指摘されている状況下にある。

従来より「ICカード等多目的利用研究会」にて審議を進めてきたCPU付ICカードに関する基盤技術整備、並びに平成7年10月より施行されているJIS X6306等はいづれも前者としての利用用途を前提としていること等を勘案して、ここでは、セキュリティ機能を持つデータキャリアとしての利用用途を対象とする。

4.3 カード所有者等に対する認証手段に関する課題

一般に、アプリケーションサービスに係わる正当な情報システムが正当なカード所有者に対してサービスを提供するためには、情報システムやアプリケーション端末がカード所有者を認証すると共に、カード所有者が情報システムやアプリケーション端末を認証する相互認証を確立することが必要不可欠である。ただし、後者に関する認証は、スタンドアローン系とネットワーク系で手順が異なり、スタンドアローン系では、アプリケーション端末と情報システムが一体化されていることから、上記の一般概念をそのまま適用可能であるが、ネットワーク系では、アプリケーション端末と情報システムがネットワーク接続されることにより一体化されていないことから、上記一般概念を適用するために別途情報システムとアプリケーション端末間の相互認証を確立することが必要であることに留意しなければならない。

CPU付ICカードを認証手段として導入した場合には、CPU付ICカードの特長である内部・外部認証機能を用いることによって、前者に関する認証を確立することが可能であるが、JIS X6306に規定されているChallenge／Response, Internal Authetication コマンドでは、Challenge／Response実行後のカード機能やInternal Authetication エラー後の対応などを規定していないため、製造メーカ各社毎に個別の機能仕様になり、CPU付ICカードの互換性を確保できない可能性が極めて高い。

また、後者に関しては、現在製品化されているJIS X6306に準拠したCPU付ICカードにアプリケーション端末を認証する機能が組み込まれていない現状にある。

従って、CPU付ICカードを現在以上に発展・普及させていくためには、認証手段として十分な機能を発揮することが極めて重要であることから、上記二点の課題について速やかに対応策を講じていくことが必要である。

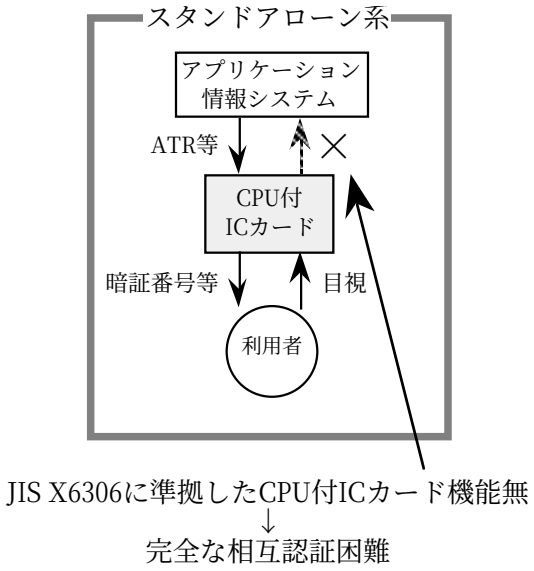


図4.3.1　スタンドアローン系における相互認証に関する一般概念

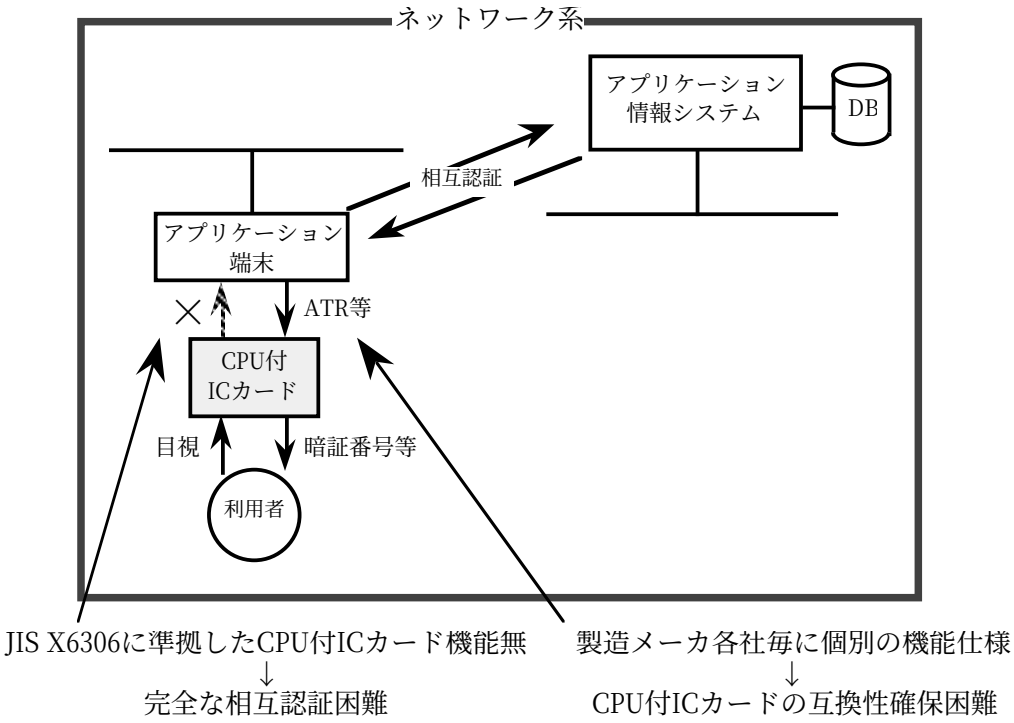


図4.3.2　ネットワーク系における相互認証に関する一般概念

4.4　基本方針

以上に示した目的、前提条件等を踏まえ、「1つのアプリケーションに関する情報を広域的に利用する尺度」と「複数のアプリケーションにまたがって情報を共通的に利用する尺度」という相互に独立な2つの尺度を設定することによって、CAMを効果的に利用するために、延いては多目的利用ICカードシステムを実用化していくために、多目的利用ICカードシステムを利用する形態を3つに分割し、それぞれの利用形態毎に整備すべき利用環境等を明確にすることを基本方針とする。

4.4.1　整備すべき利用環境

多目的利用ICカードシステムを利用する形態は、「1つのアプリケーションに関する情報を広域的に利用する尺度」と「複数のアプリケーションにまたがって情報を共通的に利用する尺度」に基づいて、図4.4.1に示す通り「第1利用形態」「第2利用形態」「第3利用形態」に3分割することが可能である。しかしながら、3つの利用形態は相互に独立な関係にあるのではなく、「第1利用形態」から「第2利用形態」へ、「第2利用形態」から「第3利用形態」へと順次移行していくことによって、「第2利用形態」さらには「第3利用形態」を実現することが可能である。

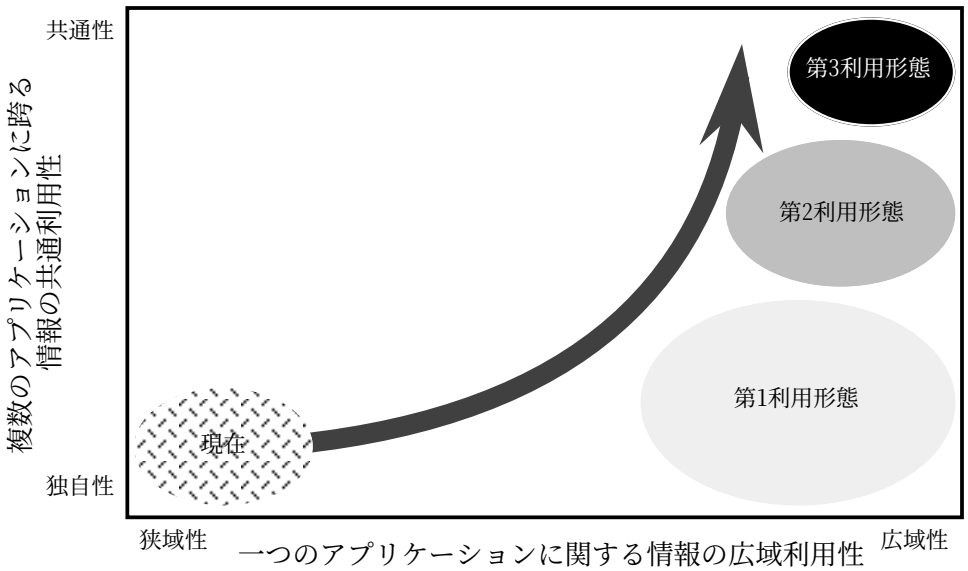


図4.4.1　基本方針

以下に、各利用形態において整備すべき利用環境を示す。

「第1利用形態」では、1つのアプリケーションにおいて提供される情報を広域的に利用するための環境整備に着手する。

また、「第2利用形態」では、「第1利用形態」により整備された環境下において、1つのアプリケーションに関する情報の1部分を他のアプリケーションに公開することにより複数のアプリケーションにまたがって情報を共通的に利用するための環境整備に着手する。

さらに、「第3利用形態」では、「第2利用形態」をより一層推進することにより、複数のアプリケーションにまたがって情報を共通的に利用するための環境整備を目指す。

現在、実用化が進められつつある保健・医療・福祉分野における保健医療カードと現在導入が検討されている運輸・交通分野における運転免許カードを例として、表4.4.1に示した各利用形態において整備すべき利用環境を以下に説明する。

表4.4.1　各利用形態において整備すべき利用環境例

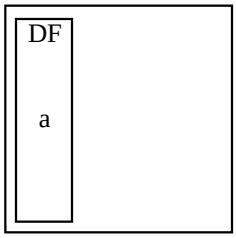
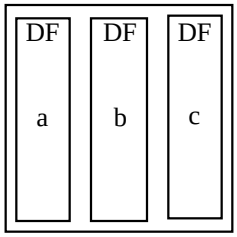
利用形態	概要
第1利用形態	■利用形態の概念 従来、個別に存在していた運転免許カード機能と保健医療カード機能を1枚のカードに集約する利用形態である。 ■整備すべき利用環境 第1利用形態では、運転免許カード機能に係わる領域と保健医療カード機能に係わる領域各々に記録される情報を各関係機関内において共通利用可能とするための環境整備に着手することが必要である。
第2利用形態	■利用形態の概念 「第1利用形態」が整備された環境下において、保健医療カード機能に係わる領域に記録される情報の一部分を運転免許カード機能に公開する、または運転免許カード機能に係わる領域に記録される情報の一部分を保健医療カード機能に公開することにより、両カード機能間で一部の情報を共通に参照可能とする形態である。 ■整備すべき利用環境 第2利用形態では、保健医療カード機能と運転免許カード機能間で共通参照する情報の調整等に着手することが必要である。
第3利用形態	■利用形態の概念 「第2利用形態」が整備された環境下において、保健医療カード機能と運転免許カード機能間で情報を相互に読み込み・書き込み可能とすることにより、保健医療カード機能と運転免許カード機能間で情報の相互運用を可能とする形態である。 ■整備すべき利用環境 第3利用形態では、保健医療カード機能と運転免許カード機能間で相互に読み込み・書き込み可能とする情報の整備に着手することが必要である。

4.4.2　各利用形態における課題・解決方針

カード発行者をA, B, Cまたそれぞれにより提供されるアプリケーションサービスをa, b, cとし、またカード発行者はカードに記録されているデータの所有者と同一であると仮定する

ならば、従来からの一般事例と第1利用形態以降それぞれにおける概念、課題、解決方針を以下の通り整理することができる。

表4.4.2　各利用形態における概念・課題・解決方針例

利用形態	概念	課題	解決方針
従来	カード発行者　A サービス提供者 A  A	■メーカ各社により製造された同一仕様のCPU付ICカードを相互利用する。 ■S型、JIS等の世代が異なるCPU付ICカードを相互利用する。	■JISに準拠したCPU付ICカードと内容アクセスマネージャにより解決する。 ■内容アクセスマネージャにより解決する。 等
第1利用形態	カード発行者　A サービス提供者 A,B,C  A	■アプリケーションサービスまたはアプリケーションサービス提供者を識別する。 ■アプリケーションサービスを広域的に共通利用する。	■ICカードアプリケーション識別子の付番制度を制定する。 ■アプリケーションサービス毎にCAMを利用するための環境を整備する。
第2利用形態	第2利用形態では、第1利用形態のファイル構造の下に、1つのアプリケーション情報の一部を他のアプリケーションに公開する。	■アプリケーション提供者間において公開すべき情報を標準化する。 ■アプリケーション提供者間において公開した情報に対する読み込み処理について運用・管理の仕組みを明確にする。	■カード発行者、アプリケーション提供者、さらにカード利用者が持つべき権限、責任等を明確にする。 ■カード本体及びアプリケーションに係わる情報の所有権の所在を明確にする。 ■アプリケーションサービス提供者間における情報の相互利用をカード所有者に提示する第三機関を設置する。 等
第3利用形態	第3利用形態では、第1利用形態のファイル構造の下に、複数のアプリケーションにまたがって情報を相互に運用・管理する。	■アプリケーション提供者間においてそれぞれが運用・管理すべき情報を規定する。 ■アプリケーション提供者間において相互に行われる読み込み・書き込み処理について運用・管理の仕組みを明確にする。	

各利用形態における課題、解決方針等に鑑みて、カード所有者が1枚のCPU付ICカードによって複数の様々なアプリケーションサービスを広域的にかつ共通的に享受できる利用環境を実現する足掛かりとなる「第1利用形態」における課題、解決方針について検討を行った結果を以下に示す。

4.5　ICカードアプリケーション識別子の付番制度について

現在、ISO/IEC　7816-5に従って（社）日本事務機械工業会が主体となり審議が進められているICカードアプリケーション識別子の付番制度案に基づき、内容アクセスマネージャの利用を想定したICカードアプリケーション識別子の付番制度に関する考え方を以下の通り整理する。

ISO/IEC 7816-5　ICカードアプリケーション識別子の付番制度

＜対象＞
■接点付ICカードだけでなく、非接触型ICカードを含めたIDカード全般

＜目的＞
■磁気ストライプカードとICカード等を用いて提供されるアプリケーションサービスの共存による社会的な混乱を回避する。
■一枚のICカードを唯一のアプリケーションサービスだけでなく、複数のアプリケーションサービスに適用可能とするための環境整備を促す。
■アプリケーションサービスの国際的または国内的な共通利用を図るための環境整備を促す。

＜登録様式＞
アプリケーション提供者識別子は以下の3つの構成のうちいずれかを持つ。

■タイプ1

登録されたアプリケーション提供者識別子RID（5バイト）	適当なアプリケーション識別拡張子PIX（11バイト）

■タイプ2

IIN ISO 7812で定義	FF	適当なアプリケーション識別拡張子PIX

■タイプ3

適当なアプリケーション提供者識別子（1～16バイト）

国内におけるICカードアプリケーション識別子の付番制度案

アプリケーション提供者識別子（RID）											アプリケーション識別拡張子（PIX）																			
D	3	9	2	X	X	Y	Y	Y	Y	Y	Y	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z	Z
(1)	(2)			(3)		(4)						(5)																			

固定　固定　国内独自の種別コード　分類コード

アプリケーションサービス提供者による識別

＜登録様式＞

分野・業種別の分類体系の付与

民間
(3)国内独自の種別コード「00：産業用」
(4)分類コード「JIS X403：産業分類」

公共
(3)国内独自の種別コード「01：行政用」
(4)分類コード「JIS X401：都道府県コード」「JIS X402：市区町村コード」

アプリケーションサービスによる識別

＜登録様式案＞

分野・業種別の分類体系の付与

広域利用
(3)国内独自の種別コード「02：広域地域サービス用」
(4)分類コード「JIS X403：産業分類」

特定地域利用
(3)国内独自の種別コード「03：特定地域サービス用」
(4)分類コード「JIS X401：都道府県コード」「JIS X402：市区町村コード」

ICカードアプリケーション識別子に関する留意事項

■複数のアプリケーションサービスが一枚のICカードに相乗りする事態を想定して、カード発行者の権限と責任範囲の明確化を順次進めていく必要がある。
■また、それと共にカードの所有権の所在を明確にすることが必要である。

内容アクセスマネージャを想定したICカードアプリケーション識別子の付番制度

＜対象＞
同左

＜目的＞
ICカードアプリケーション識別子にアプリケーションサービスの業種・業務を識別するための情報を記述することによって、アプリケーションサービスの広域的な利用を図る。
とりわけ、内容アクセスマネージャを導入する場合には、上記の効果・効能が最大限に発揮される。

＜登録様式案＞
アプリケーション提供者識別子に含まれるアプリケーション識別拡張子（PIX）に対象となるDFの内部構成や定義主体等を明示するための情報として「DF識別子」を登録する。

アプリケーション識別拡張子（PIX）						
Z	Z	Z	Z		・・・・・・・・・	
(5)						

DF識別子

■DF識別子
アプリケーション識別拡張子は「ICカードアプリケーション識別子の申請・登録（付番）に関する国内規則」において業種・分野及び機関・団体等で統一化された任意の識別コードとして規定されている。
従って、対象となるアプリケーションサービスを利用する業種・分野を管掌、調整する機関・団体毎に標準化される必要がある。

DF識別子に関する留意事項

■DF識別子を運用・管理する実施体制、仕組み等を明確にし、その普及啓蒙活動を推進していくことが重要である。

4.6　内容アクセスマネージャを対象とした利用環境の整備

CAMを導入することにより、カード事業実施・推進主体が国際または国内的な規模で共通利用可能であるCPU付ICカードシステムを構築する、または複数のアプリケーションサービスを相乗りできるCPU付ICカードシステムを構築するためには、カード所有者が持つ一般のICカードであるデータカードに関して以下に示す要素を順次整備していくことが必要不可欠である。[注1]

表4.6.1　データカードについて整備すべき要素

ファイル構成	整備すべき要素
DF（Dedicated File）	DF名称（ICカードアプリケーション識別子）
	業務提供者・端末キー
EF（Elementary File）	EF識別子
	業務提供者・端末キー
タグ番号	タグ番号に対応した標準項目名
データ実体	データ実体の記述形式

また、CAMでは、上記のデータカードの他に、それに対する読み込み・書き込み等の様々なオペレーションを制御するためのICカードとしてオペレーションカードを導入することから、データカードと併せて以下に示す要素を順次整備しなければならない。

表4.6.2　オペレーションカードについて整備すべき要素

ファイル構成	整備すべき要素
DF（Dedicated File）	DF名称
	業務提供者・端末キー
EF（Elementary File）	EF識別子
	業務提供者・端末キー
データ実体	データ実体に対するアクセス権限

ただし、カード事業実施・推進主体によっては、整備に関する手続き・手順等が大きく異なることから、カード事業実施・推進主体を行政機関、業界団体、さらにICカード利用業務提供者に区別して以下纏める。

[注1]CAMを用いず、国際・国内的な規模での共通利用を目指すアプリケーションサービスを対象としたCPU付ICカードシステムを構築するためには、これらの項目に加えて、導入するCPU付ICカード及びリーダ・ライタ等に関する機能・仕様を統一化する等の基盤技術を整備する必要がある。

4.6.1　データカードにおけるDF名称

データカードにおけるDF名称は、一枚のCPU付ICカードに複数のアプリケーションサービスが相乗りした場合に、それぞれのアプリケーションサービスに係わるファイルの最上位管理単位である。

国際的な相互利用を目的とするアプリケーションサービスに関するDF名称は、ISO/IEC 7816-5にて「ICカードアプリケーション識別子」としてISOの所定登録管理機構（KTAS）に申請・登録することが規定されている。また、国内に限定した相互利用を目的とするアプリケーションサービスに関するDF名称は、ISO/IEC 7816-5にて「ICカードアプリケーション識別子」として国内標準化機構に申請・登録することが規定されており、日本では、現在社団法人日本事務機械工業会が「ICカードアプリケーション識別子」としての付番制度及び申請・登録について審議を進めている状況にある。

従って、ここでは、社団法人日本事務機械工業会より平成7年12月25日付で提示された「ICカードアプリケーション識別子の申請・登録（付番）に関する国内規則（案）」に従った手続き・手順を示すこととする。

(1) 行政機関

! 関係行政機関を中心に学識経験者等の協力の下、対象業務に関連するICカードアプリケーション識別子の標準化作業を実施する。

" 「ICカードAID管理センター」へ登録申請する。「ICカードAID管理センター」とは、社団法人日本事務機械工業会内に設置された登録機関の名称である。

(2) 業界団体

! 関係業界団体を中心に民間企業等の協力の下、対象業務に関連するICカードアプリケーション識別子の標準化作業を実施する。

"「 ICカードAID管理センター」へ登録申請する。

(3) ICカード利用業務提供者

! ICカード利用業務提供者が対象業務を所轄する業界団体にその旨を申請する。

" 関係業界団体を中心に民間企業等の協力の下、対象業務に関連するICカードアプリケーション識別子の標準化作業を実施する。

ICカードAID管理センターへ登録申請する。

4.6.2　オペレーションカードにおけるDF名称

オペレーションカードにおけるDF名称は、データカードと同様、オペレーションカードに係わるファイルの最上位管理単位である。国内に限定した相互利用を目的とするアプリケーションサービスに関するデータカードのDF名称は「ICカードアプリケーション識別子」として国内標準化機構に申請・登録することが規定されているが、オペレーションカードにおけるDF名称はその対象に含まれない可能性が高いことから、データカードにおけるDF名称とは区別した形で以下に手続き・手順等を示す。

(1) 行政機関

! 関係行政機関を中心に学識経験者等の協力の下、オペレーションカードにおけるDF名称の標準化作業を実施する。

(2) 業界団体

! 関係業界団体を中心に民間企業等の協力の下、オペレーションカードにおけるDF名称の標準化作業を実施する。

(3) ICカード利用業務提供者

! ICカード利用業務提供者が対象業務を所轄する業界団体にその旨を申請する。

" 関係業界団体を中心に民間企業等の協力の下、オペレーションカードにおけるDF名称の標準化作業を実施する。

4.6.3　データカード及びオペレーションカードにおける業務提供者・端末キー

業務提供者・端末キーは、データカード及びオペレーションカードにおけるファイル最上位管理単位であるDFとその配下に割り付けられるEFへのアクセス制御を管理するためのセキュリティ機構である。

(1) 行政機関

! 関係行政機関を中心にメーカ各社の協力の下、データカード及びオペレーションカード内で対象となる各DF・EFに割り当てる業務提供者・端末キーを規定する。

(2) 業界団体

! 関係業界団体を中心にメーカ各社の協力の下、データカード及びオペレーションカー

ド内で対象となる各DF・EFに割り当てる業務提供者・端末キーを規定する。

(3) ICカード利用業務提供者

! ICカード利用業務提供者が対象業務を所轄する業界団体にその旨を申請する。

" 関係業界団体を中心にメーカ各社の協力の下、データカード及びオペレーションカード内で対象となる各DF・EFに割り当てる業務提供者・端末キーを規定する。

4.6.4 データカード及びオペレーションカードにおけるEF識別子

EF識別子は、既に述べたファイルの最上位管理単位であるDFの配下に割り付けられるEFそれぞれに付与する番号であり、JIS X6306では、EF識別子として「0x01」～「0x1e」の番号を許容している。

(1) 行政機関

! 関係行政機関を中心に学識経験者等の協力の下、データカードにおける対象DF内の各EF識別子に対応する業務名、及びオペレーションカードにおける対象DF内の所有者情報ファイル、キーテーブルファイル、アクセスコントロールテーブルファイルの各EF識別子の標準化作業を実施する。

(2) 業界団体

! 関係業界団体を中心に民間企業等の協力の下、データカードにおける対象DF内の各EF識別子に対応する業務名、及びオペレーションカードにおける対象DF内の所有者情報ファイル、キーテーブルファイル、アクセスコントロールテーブルファイルの各EF識別子の標準化作業を実施する。

(3) ICカード利用業務提供者

! ICカード利用業務提供者が対象業務を所轄する業界団体にその旨を申請する。

" 関係業界団体を中心に民間企業等の協力の下、データカードにおける対象DF内の各EF識別子に対応する業務名、及びオペレーションカードにおける対象DF内の所有者情報ファイル、キーテーブルファイル、アクセスコントロールテーブルファイルの各EF識別子の標準化作業を実施する。

4.6.5　データカードにおけるタグ番号に対応した標準項目名

CAMは、簡易符号化TLVデータ形式に準拠してJIS　X6306に規定されたレコードフォーマットのデータ実体にデータを記録することから、タグ番号に対応した標準項目名は、CAMが対象とするタグ番号に対応したデータ実体に記述するデータの種類等を明確にするための情報である。

(1) 行政機関

! 関係行政機関を中心に学識経験者等の協力の下、各タグ番号に対応する標準項目名の標準化作業を実施する。

(2) 業界団体

! 関係業界団体を中心に民間企業等の協力の下、各タグ番号に対応する標準項目名の標準化作業を実施する。

(3) ICカード利用業務提供者

! ICカード利用業務提供者が対象業務を所轄する業界団体にその旨を申請する。

" 関係業界団体を中心に民間企業等の協力の下、各タグ番号に対応する標準項目名の標準化作業を実施する。

4.6.6　データカードにおけるデータ実体の記述形式

データカードにおけるタグ番号に対応したデータ実体の記述形式として、数字バイナリ・符号付数字パック・英数カナ・漢字・バイナリ等の属性、桁数、バイト数等を規定する。

(1) 行政機関

! 関係行政機関を中心に学識経験者等の協力の下、標準項目名に対応するデータ実体のデータ記述形式の標準化作業を実施する。

" また、標準項目名は利用頻度の最も高い用語に過ぎないため、標準項目名の同義語を併せて整理する。

(2) 業界団体

! 関係業界団体を中心に民間企業等の協力の下、標準項目名に対応するデータ実体のデータ記述形式の標準化作業を実施する。

" また、標準項目名は利用頻度の最も高い用語に過ぎないため、標準項目名の同義語を併せて整理する。

(3) ICカード利用業務提供者

! ICカード利用業務提供者が対象業務を所轄する業界団体にその旨を申請する。

" 関係業界団体を中心に民間企業等の協力の下、標準項目名に対応するデータ実体のデータ記述形式の標準化作業を実施する。

また、標準項目名は利用頻度の最も高い用語に過ぎないため、標準項目名の同義語を併せて整理する。

4.6.7 オペレーションカードにおけるデータ実体に対するアクセス権限

オペレーションカードにおけるデータ実体に対するアクセス権限は、各EF内のデータ実体に対して操作者に与えられている職種・資格等の権限である。

(1) 行政機関

! 関係行政機関を中心に学識経験者等の協力の下、各データ実体に対するアクセス権限の標準化作業を実施する。

(2) 業界団体

! 関係業界団体を中心に民間企業等の協力の下、各データ実体に対するアクセス権限の標準化作業を実施する。

(3) ICカード利用業務提供者

! ICカード利用業務提供者が対象業務を所轄する業界団体にその旨を申請する。

" 関係業界団体を中心に民間企業等の協力の下、各データ実体に対するアクセス権限の標準化作業を実施する。

4.6.8 その他留意事項

(1) 普及啓蒙の観点から

カード事業実施・推進主体において、ICカードアプリケーション識別子、DF識別子、さらにEF識別子を運用・管理する実施体制、仕組み等を明確にし、それらの普及啓蒙活動を推進していくことが重要である。

(2) セキュリティの観点から

カード事業実施・推進主体において、業務提供者・端末キーに関する情報を開示する対象・範囲、契約事項等を明確にし、厳重な管理を行うことが必要である。

5.今後の課題及び総括

ここでは、本年度の調査研究内容を踏まえ、将来、外部端子付ICカードや非接触型ICカード等のCPU付ICカードを社会的なインフラストラクチャとして位置づけていくために、今後解決すべき技術的な課題を総括する。

5.1　解決すべき技術的な課題

CPU付ICカードが社会的なインフラストラクチャとして根ざしていくために解決すべき技術的な課題を、内容アクセスマネージャの機能向上に関する課題と「4.内容アクセスマネージャを効果的に利用するための環境整備」における「第3利用形態」を想定したCPU付ICカードの機能向上に関する課題に分類して、以下に示す通り纏める。

5.1.1　内容アクセスマネージャの機能向上に関する課題

(1) JIS X6306に規定されたレコード記録形式への対応

将来的なICカード内のメモリの有効利用、及びICカードに記録されているデータの検索等の簡易化を図ることを目的として、CAM Version2.0において規定しているレコード記録形式をJIS X6306に規定されている簡易符号化TLVデータ形式に統合可能にすることが望ましい。

ただし、CAMにおいて利用しているタグの拡張機能がJIS X6306に規定されていないため、ISO/IEC 7816-4に規定されている簡易符号化TLVデータ形式におけるタグの拡張機能との整合性を明確にする必要がある。

(2) ISO/IEC 7816-4に規定された透過ファイルへの対応

CAM Version2.0では、JIS X6306に規定されたファイル構造に従い、固定長順編成ファイル、可変長順編成ファイルを対象としているが、将来的なICカード内のメモリの有効利用を図るために、JIS X62306に規定されているファイル構造以外にISO/IEC 7816-4に規定されている透過ファイルへも対応可能にする必要がある。

(3) タグの拡張への対応

現在、CAM Version2.0は、1バイトでタグ番号を記述する場合に先頭ビットをタグ番号を拡張するための識別子として利用しているため、0x8000以上のタグ番号を利用不可とする機能仕様になっている。

従って、今後0x8000以上のタグ番号を利用可能とすると共に、ISO/IEC 7816-4に規定されているレコード識別子の拡張方法と整合性のあるタグ番号の拡張方法を確立することが必要である。

(4) オプションコマンドの拡張への対応

現在、CAM Version2.0はJIS X6306に規定されている基本コマンドのみを対象とした機能・関数仕様となっているが、今後より一層CPU付ICカードの柔軟性や拡張性を向上していくために、基本コマンド以外に存在する各種オプションコマンドへの対応を図っていくことが極めて重要である。

(5) CPU付ICカードの識別への対応

現時点では、カード製造メーカ各社により製造されるCPU付ICカードを登録する制度が整備されるか否か不明であるが、このような制度が整備されていくことを念頭に置きつつ、カード製造メーカ各社のCPU付ICカードを識別可能とする手段を講じていくことが肝要である。

5.1.2 CPU付ICカードの機能向上に関する課題

(1) システムコマンドの公開に向けた対応

将来的に「4.内容アクセスマネージャを効果的に利用するための環境整備」に示す「第1利用形態」から「第2利用形態」さらには「第3利用形態」へと1枚のCPU付ICカードを取り巻く利用形態が移行していく可能性が十分にあることを踏まえるならば、従来から現在までに顕在化してきた様々な弊害を順次解決していくことが極めて重要であると考えられる。

現在、その重大な一つの弊害として、カード製造メーカ各社がDF・EF等のファイル創成やそれらに関する鍵の設定等を制御するシステムコマンドを非公開としていることが指摘されていることから、システムコマンドに対して鍵を付与する等のシステムコマンドを公開していくための対応策を明確にし、実装していくことが重要である。

(2) 本人確認等の相互認証技術の確立

コンピュータシステムのダウンサイジング化、ネットワーク化、オープンシステム化等の電子情報技術に関する技術革新により、従来では想定され得なかった、商取引・決済等の経済活動を様々なコンピュータ・ネットワークを複合的に用いて実施するエレクトロニック・コマースが可能になってきている。

企業と個人、企業と企業、個人と個人等の様々な取引形態、かつ様々な利用頻度において取り行われるエレクトロニック・コマースを実現するためには、オープン・クローズを問わずコンピュータ・ネットワーク上で本人確認や相互認証を確立するためのセキュリティ技術を整備することが必要不可欠であり、マイクロプロセッサを搭載したメモリであるCPU付ICカードは、その機能的な特徴から一つの実用的ツールとして脚光を浴びており、積極的に導入されていくことが予想される。

現在では、CPU付ICカードにより本人確認を確立する方策として、「電子認証局や電子公証局を介して認証する方法」と「CPU付ICカード・アプリケーション端末・情報システム間において相互認証する方法」が挙げられているが、前者に関しては、行政機関等における電子認証局等の運用・管理に関する実施体制及び仕組み等を事前に明確にすることが必要であり、実用までに相当の期間を要すると予想される。

以上を踏まえ、後者に関する本人確認の方策を可及的速やかに実現していくことが必要であると考えられる。

5.2　総括

平成7年度の調査研究では、平成7年10月よりJIS　X6306が施行されたことを踏まえ、従来からの暫定的な国内標準として位置づけられた16社仕様準拠ICカードだけでなく、新たに制定されたJIS　X6306に準拠したICカードに内容アクセスマネージャを対応させるべく、既に公開されているCAM Version1.1に関する機能仕様書及び標準ソフトウェアのレビュー及びバージョンアップを図り、CAM Version2.0に関する機能仕様書及び標準ソフトウェアを取り纏めた。

また、それと共に、将来の高度情報通信社会の到来に備えて、カード所有者である国民が公共分野に限らず複数の様々なアプリケーションサービスを1枚のカードメディアにより広域的また共通的に享受できる利用環境を健全にかつ円滑に実現していくために必要不可欠となるであろう基本方針、作業手順、整備項目を整理し、その端緒として位置づけられるICカードアプリケーション識別子の付番制度及び内容アクセスマネージャを効果的に利用するための環境整備について議論を行った。

今後は、平成4年度より継続している本調査研究の成果である内容アクセスマネージャが「CPU付カードメディアの普及促進への貢献を実践するための技術的基盤」の一つとしてより広く認知され、かつより積極的に導入・実用化されていくことによって、次世代ICカードへのより高次な要求機能の実現、延いては次世代ICカードの実用化に向けた周辺環境整備を促進するための一端を担っていくことを期待する処である。

【参考資料】

1. CAM標準ソフトウェアVersion2.0の構成関数概要

CAM Version2.0は、関数構成をアプリケーション・インタフェース関数、アプリケーション・インタフェース関数を実現するための内部関数、そしてデバイスドライバ共通インタフェース関数を制御するための内部関数の三階層に分割し、アプリケーション・インタフェース関数及びデバイスドライバ共通インフェース関数のインタフェース条件のみを規定することによって、その実現方法を制限することなく、メーカ各社が製造する仕様が異なるICカードを、また世代が異なるICカードを相互に運用可能にすることを目指したミドルウェアである。

ICカード等多目的利用研究会では、上記の目的を検証・評価するだけでなく、CAM Version2.0機能仕様書に記述されない詳細を補足するべく、CAM標準ソフトウェア Version2.0を開発したので、以下にそれらの機能・仕様・構成等について紹介する。

また、ここではその他にCAM標準ソフトウェアVersion2.0を構成する最下位に位置するデバイスドライバ共通インタフェース関数を制御するための内部関数よりコールされる、デバイスドライバ共通インタフェース関数、OSに依存する部分である伝送制御関数についても紹介する。ただし、アプリケーション・インタフェース関数及びデバイスドライバ共通インタフェース関数は、【本編】に機能・仕様・構成等について既述した通りであることから、【参考資料】では詳述しないこととする。

CAM標準ソフトウェアVersion2.0を構成及び関連する関数は、表1-1-1,2に示す通り整理される。

表1-1-1　CAM標準ソフトウェアVersion2.0の構成関数分類

大分類	細分類
アプリケーション・インタフェース関数	同左
内部関数	DF情報管理に関する内部関数
	拡張ファイル管理に関する内部関数
	メモリ内のデータ管理に関する内部関数
	ユーティリティ内部関数
	循環型基礎ファイル管理に関する内部関数
	複数個のデータ実体から構成されるデータ管理に関する内部関数
	アクセス制御に関する内部関数
デバイスドライバ共通インタフェース関数を制御するための内部関数	同左

表1-1-2　CAM標準ソフトウェアVersion2.0の関連関数

大分類	細分類
デバイスドライバ共通インタフェース関数	同左
伝送制御関数	Windows3.1を対象としたJIS準拠ICカード対応デバイスドライバ関数を制御する内部関数

1.1　アプリケーション・インタフェース関数

【本編】を参照されたい。

1.2　内部関数

CAM標準ソフトウェアVersion2.0におけるアプリケーション・インタフェース関数を実現するための内部関数は以下の通りである。

1.2.1　DF情報管理に関する内部関数

表1-2-1　DF情報管理に関する内部関数一覧

関数名	機能概要
icut_open_dfi	指定したDF情報ファイルをオープンする。
icut_close_dfi	指定したDF情報ファイルをクローズする。
icut_load_DF_info	カード上のDF情報を読み込む。
icut_save_DF_info	ホスト上のDF情報をカードに書き込む。
icut_search_file	DF情報ファイル内でEF情報を検索する。
icut_register_file_info	DF情報ファイルにEF情報を追加する。
icut_update_file_info	DF情報ファイル内のEF情報の更新日時を変更する。
icut_get_time	システムが管理する日時を配列へ入れる。
icut_init_dfi	指定したDF情報ファイルを初期化する。
icut_close_dfi_if_unused	DF情報ファイルが利用されていない場合にDF情報ファイルをクローズする。

1.2.2　拡張ファイル管理に関する内部関数

表1-2-2　拡張ファイル管理に関する内部関数一覧

関数名	機能概要
icut_open_faf	拡張ファイルをオープンする。
icut_close_faf	拡張ファイルをクローズする。
icut_append_item_faf	拡張ファイルにデータを追加する。
icut_replace_item_faf	拡張ファイルにおいてデータを入れ換える。
icut_remove_item_faf	拡張ファイルからデータを削除する。
icut_get_item_faf	拡張ファイル内のデータを得る。
icut_get_item_faf_sequential	拡張ファイル内のデータを順次得る。
icut_load_faf	拡張ファイルのデータをバッファに読む。
icut_save_faf	拡張ファイルのデータをセーブする。
icut_define_faf	CAM用に拡張ファイルを初期化する。
icut_new_faf_code	拡張ファイル内の利用可能なコードを探す。
icut_close_faf_if_unused	拡張ファイルが利用されていない場合に拡張ファイルをクローズする。

1.2.3　メモリ内のデータ管理に関する内部関数

表1-2-3　メモリ内のデータ管理に関する内部関数一覧

関数名	機能概要
icut_get_item_sequential_mem	バッファ内のデータをファイル単位で順次得る。
icut_get_item_mem	バッファ内のデータを得る。
icut_append_item_mem	バッファにデータを追加する。
icut_replace_item_mem	バッファ内のデータを入れ換える。
icut_remove_item	バッファからデータを削除する。
icut_get_faf_item_sequential_mem	バッファ内の拡張ファイルのデータをファイル単位で順次得る。
icut_get_faf_item_mem	バッファ内の拡張ファイルのデータを得る。
icut_append_faf_item_mem	バッファに拡張ファイルのデータを追加する。
icut_replace_faf_item_mem	バッファ内の拡張ファイルのデータを入れ換える。
icut_remove_faf_item_mem	バッファから拡張ファイルのデータを削除する。

1.2.4　ユーティリティ内部関数

表1-2-4　ユーティリティ内部関数一覧

関数名	機能概要
icut_append_item	バッファにデータを追加する。
icut_replace_item	バッファ内のデータを入れ換える。
icut_remove_item	バッファからデータを削除する。
icut_get_item	バッファ内のデータを得る。
icut_get_item_sequential	バッファ内のデータをファイル単位で順次得る。
icut_load_ef_to_buffer	カード上の基礎ファイル内のデータをバッファに読む。
icut_write_ef	バッファ内のデータをカード上の基礎ファイルに書き込む
icut_read_ef	カード上の基礎ファイル内のデータをバッファ上に読み出す。
icut_erase_ef	カード上の基礎ファイルを削除する。
icut_clear_ef	基礎ファイル内を初期化する。
icut_define_file	カード上に基礎ファイルを創成する。
icut_init_ef	基礎ファイルの初期化を行う。
icut_get_length	データからデータ長を求める。
icut_set_length	データにデータ長を設定する。
icut_count_length	データ長領域のサイズを求める。
icut_get_tagcode	データからタグを求める。
icut_set_tagcode	データにタグを設定する。
icut_count_tagcode	タグのサイズを求める。
icut_file_writable	書き込み権限をチェックする。
icut_get_status	ステータスをチェックする。
icut_match_name	文字列を比較する。
icut_copy_name	文字列をコピーする。
icut_error_code	ステータスコードからエラーコードを求める。

1.2.5　循環型基礎ファイル管理に関する内部関数

表1-2-5　循環型基礎ファイル管理に関する内部関数一覧

関数名	機能概要
icut_is_rotation_file	オープンされているDF内の論理的なEF-IDのファイルが循環型基礎ファイルであるか否かを調べる。
icut_get_actual_efid	オープンされているDF内の論理EF-IDのファイルの実EF-IDを得る。
icut_get_order_v0100	CAM Version1.0で記録された、自治省が主導する地域カードシステムの循環型基礎ファイルの情報を得る。

1.2.6　複数個のデータ実体から構成されるデータ管理に関する内部関数

表1-2-6　循環型基礎ファイル管理に関する内部関数一覧

関数名	機能概要
icut_check_group_item_data	複数個のデータ実体を持つデータに関するフォーマットをチェックする。

1.2.7　アクセス制御に関する内部関数

表1-2-7　アクセス制御に関する内部関数一覧

関数名	機能概要
icut_verify_df	対象DFのキー情報をベリファイする。
icut_verify_item	対象基礎ファイル内のデータのアクセス権をベリファイする。
icut_init_operation_info	オペレーションカード内のキーテーブル等の情報を格納するための領域等を準備する。
icut_init_DF_access_info	オペレーションカード内のキーテーブル等の情報を格納するためのDF毎の領域等を準備する。
icut_get_DF_no_access_info	DF毎のアクセスに係わる情報の領域を指定するためのパラメータを検索する。
icut_clear_operation_info	バッファ内のアクセス権情報をクリアする。
icut_key_data_len	キーデータ列のバッファ長を計算する。
icut_set_key_info	キーテーブルをセットする。
icut_set_operation_card_key	オペレーションカード内のキーテーブルをバッファに設定する。
icut_set_operation_card_table	オペレーションカード内のアクセスコントロールテーブルをバッファに設定する。

1.3　デバイスドライバ共通インタフェース関数を制御するための内部関数

CAM標準ソフトウェアVersion2.0におけるデバイスドライバ共通インタフェース関数を制御するための内部関数は以下の通りである。

表1-3-1　デバイスドライバ共通インタフェース関数を制御するための内部関数一覧

関数名	機能概要
icd_std_open	デバイスドライバの初期化を行う。
icd_std_close	RS-232Cポートのクローズを行う。
icd_set_wait_time	カードの挿入待ち時間を設定する。
icd_card_info	挿入されているカードの種類を得る。
icd_read_record	基礎ファイルからレコードを読み出す。
icd_write_record	基礎ファイルにレコードを書き込む。
icd_append_record	基礎ファイルにレコードを追加する。
icd_update_record	基礎ファイル内のレコードを更新する。
icd_erase_all_record	基礎ファイル内の全レコードを消去する。
icd_select_DF	対象であるDFを選択する。
icd_in_card	カードを挿入する。
icd_out_card	カードを排出する。
icd_get_status	ステータスを調べる。
icd_show_status	ステータスを印刷する。
icd_verify	DFに関するキーを照合する。

1.4　デバイスドライバ共通インタフェース関数

【本編】を参照されたい。

1.5　伝送制御関数

CAM Version2.0では、UNIX，MS-DOS，Windows3.1等対象とするOSに依存して伝送制御関数を規定することが必要である。CAM標準ソフトウェアVersion2.0では、Windows3.1を対象に開発した伝送制御関数の一例を紹介する。

以下、Windows3.1に依存した伝送制御関数一覧を示す。

表1-5-1　Windows3.1を対象とした伝送制御関数例一覧

関数名	機能概要
icjd_open_device	ポートをオープンし、伝送パラメータをセットする。
icjd_set_comm_param	コミュニケーションパラメータをセットする。
icjd_close_device	ポートをクローズする。
icjd_send	コミュニケーションパラメータを送信する。
icjd_recieve	コミュニケーションパラメータを受信する。

2.アプリケーション・インタフェース関数

【本編】を参照されたい。

3. 内部関数

3.1　DF情報管理に関する内部関数

3.1.1　icut_open_dfi

(1)呼出形式

```
short    icut_open_dfi( fname, nlen, mode, dfd )
```

(2)引き数

```
unsigned char *fname     /*   DF名   */
int nlen                 /*   DF名長   */
int mode                 /*   ファイルオープンモード   */
int *dfd                 /*   DFディスクリプタ   */
```

(3)機能

ICカード上のDF情報ファイルをオープンする。ただし、「ic_def.h」に定義される5つファイルオープンモード「ReadOnly」「WriteOnly」「ReadWrite」「Rotation」「Overwrite」からファイルをオープンするモードを設定する必要がある。

(4)コール関数一覧

コール関数	
icut_match_name	
icut_load_DF_info	

3.1.2　icut_close_dfi

(1)呼出形式

```
short　icut_close_dfi( dfd )
```

(2)引き数

```
int dfd                    /*　DFディスクリプタ　*/
```

(3)機能

ICカード上のDF情報ファイルをクローズする。

(4)コール関数一覧

コール関数	
icut_save_DF_info	

3.1.3　icut_load_DF_info

(1)呼出形式

short　icut_load_DF_info(fname, nlen, mode, dfd)

(2)引き数

unsigned char *fname　/*　DF名　*/

int nlen　/*　DF名長　*/

int mode　/*　ファイルオープンモード　*/

int dfd　/*　DFディスクリプタ　*/

(3)機能

ICカード上のDF情報ファイルからデータをロードする。ただし、「ic_def.h」に定義される5つのファイルオープンモード「ReadOnly」「WriteOnly」「ReadWrite」「Rotation」「Overwrite」からファイルをオープンするモードを設定する必要がある。

(4)コール関数一覧

コール関数	
icut_read_ef	
icut_file_writable	
icut_get_item_mem	
icut_get_sequential_item_mem	
icut_open_faf	
icut_get_tagcode	
icut_get_faf_item_mem	

3.1.4　icut_save_DF_info

(1)呼出形式

```
short　icut_save_DF_info( dfno )
```

(2)引き数

```
int dfno                    /*　DF構造体番号　*/
```

(3)機能

ICカード上のDF情報ファイルにデータをセーブする。

(4)コール関数一覧

コール関数	
icut_read_ef	icut_define_faf
icut_get_time	icut_remove_faf_item_mem
icut_remove_item_mem	icut_get_item_mem
icut_append_item_mem	icut_get_length
icut_file_writable	icut_close_faf
icut_new_faf_code	icut_set_tagcode
icut_append_faf_item_mem	
icut_write_ef	
icut_open_faf	
icut_get_item_sequential_mem	

3.1.5　icut_search_file

(1)呼出形式

```
short    icut_search_file( id, dfno )
```

(2)引き数

```
unsigned char id          /*   実EF-ID   */
int dfno                  /*   DFディスクリプタ   */
```

(3)機能

DF情報ファイル中に対象となる基礎ファイルに関するEF情報が存在するか否かを調べる。基礎ファイルはDFディスクリプタと実EF-IDによって指定する。対象となる基礎ファイルが存在した場合には、実EF-IDを戻り値として返し、存在しない場合には戻り値として「-1」を返す。

(4)コール関数一覧

無し

3.1.6　icut_register_file_info

(1)呼出形式

short　icut_register_file_info(id, dfno, type, count)

(2)引き数

unsigned char id　　/*　実EF-ID　*/

int dfno　　/*　DFディスクリプタ　*/

int type　　/*　ファイル種類　*/

int count　　/*　再書き込みカウンタ　*/

(3)機能

DF情報ファイル内に対象となる基礎ファイルに関するEF情報を登録する。ただし、基礎ファイルを登録する場合には必ずファイルタイプの指定を行う。ファイルタイプの種類はヘッダファイル内で定義されており、以下の通りである。基礎ファイルはDFディスクリプタと実EF-IDによって指定する。

ファイル種類	ファイルタイプ
DF情報ファイル	DF_INFO_TYPE
基礎ファイル	FILE_TYPE
拡張ファイル	FAF_TYPE
キーテーブルファイル	KEY_FILE_TYPE
アクセスコントロールテーブルファイル	ACESS_TABLE_TYPE
所有者情報ファイル	OWNER_INFO_TYPE

(4)コール関数一覧

コール関数	
icut_search_file	
icut_get_time	

3.1.7　icut_update_file_info

(1)呼出形式

```
short    icut_update_file_info( id, dfno )
```

(2)引き数

```
short id                /*   実EF-ID   */
int dfno                /*   DFディスクリプタ   */
```

(3)機能

DF情報ファイル内に登録されている基礎ファイルに関するEF情報の更新日時を現在の日時に変更する。基礎ファイルはDFディスクリプタと実EF-IDによって指定する。

(4)コール関数一覧

コール関数	
icut_search_file	
icut_get_time	

3.1.8　icut_get_time

(1)呼出形式

```
void　icut_get_time( dt )
```

(2)引き数

```
struct *dt                    /*　データ構造体（構造体名 datetype ）　*/
```

(3)機能

日時の構造体内に現在の日時を入れる。

(4)コール関数一覧

無し

3.1.9　icut_init_dfi

(1)呼出形式

short　icut_init_dfi(fname, nlen)

(2)引き数

unsigned char *fname　/*　DF名　*/

int nlen　/*　DF名長　*/

(3)機能

DF情報ファイルを初期化する。

(4)コール関数一覧

コール関数	
icut_open_dfi	
icut_close_dfi	
icut_init_ef	
icut_get_time	
icut_save_DF_info	

3.1.10　icut_close_dfi_if_unused

(1)呼出形式

```
short    icut_close_dfi_if_unused( dfno, fd )
```

(2)引き数

```
int *dfno                /*   DFディスクリプタ   */
int fd                   /*   ファイルディスクリプタ   */
```

(3)機能

オープンされている基礎ファイルの内、ファイルディスクリプタで指定した基礎ファイル以外でDF情報ファイルが利用されていない場合に、DF情報ファイルをクローズする。

(4)コール関数一覧

コール関数	
icut_close_dfi	

3.2　拡張ファイル管理に関する内部関数

3.2.1　icut_open_faf

(1)呼出形式

```
short    icut_open_faf( dfno )
```

(2)引き数

```
int dfno                    /*   DFディスクリプタ   */
```

(3)機能

対象となるDFが使用する拡張ファイルをオープンする。拡張ファイルを使用するDFのDFディスクリプタよって指定する。

(4)コール関数一覧

コール関数	
icut_match_name	
icut_close_faf	
icut_open_dfi	
icut_load_faf	

3.2.2　icut_close_faf

(1)呼出形式

short　icut_close_faf(void)

(2)引き数

無し

(3)機能

拡張ファイルをクローズする。

(4)コール関数一覧

コール関数	
icut_save_faf	
icut_close_dfi	

3.2.3　icut_append_item_faf

(1)呼出形式

short　icut_append_item_faf(tag, data, len, fd)

(2)引き数

```
int tag;                    /*  タグ  */
unsigned char *data;        /*  データ実体  */
int len;                    /*  データ長  */
int fd;                     /*  ファイルディスクリプタ  */
```

(3)機能

対象となる基礎ファイルの拡張ファイル領域にデータを追加する。基礎ファイルはファイルディスクリプタによって指定する。データはタグ・データ長・データへのポインタによって指定する。

(4)コール関数一覧

コール関数	
icut_append_faf_item_mem	

3.2.4　icut_replace_item_faf

(1)呼出形式

short　icut_replace_item_faf(tag, data, len, fd)

(2)引き数

int tag;　　/*　タグ　*/

unsigned char *data;　　/*　データ実体　*/

int len;　　/*　データ長　*/

int fd;　　/*　ファイルディスクリプタ　*/

(3)機能

対象となる基礎ファイルの拡張ファイル領域のデータを入れ換える。基礎ファイルはファイルディスクリプタによって指定する。データはタグ・データ長・データへのポインタによって指定する。ただし、データを入れ換える場合には、存在する2つのデータ長が同一でなければならない。

(4)コール関数一覧

コール関数	
icut_replace_faf_item_mem	

3.2.5　icut_remove_item_faf

(1)呼出形式

```
short    icut_remove_item_faf( tag, fd )
```

(2)引き数

```
int tag;                /*   タグ   */
int fd;                 /*   ファイルディスクリプタ   */
```

(3)機能

対象となる基礎ファイルの拡張ファイル領域のデータを消去する。基礎ファイルはファイルディスクリプタによって指定する。データはタグによって指定する。

(4)コール関数一覧

コール関数	
icut_remove_faf_item_mem	

3.2.6　icut_get_item_faf

(1)呼出形式

```
short    icut_get_item_faf( tag, data, dlen, len, fd )
```

(2)引き数

```
int tag;                    /*  タグ  */
unsigned char *data;        /*  データ実体  */
int dlen;                   /*  データバッファの長さ  */
int *len;                   /*  データ長  */
int fd;                     /*  ファイルディスクリプタ  */
```

(3)機能

対象となる基礎ファイルの拡張ファイル領域のデータを読み込む。基礎ファイルはファイルディスクリプタによって指定する。データはタグによって指定する。読み出したデータが指定領域にコピーされ、データ長が返される。

(4)コール関数一覧

コール関数	
icut_get_faf_item_mem	

3.2.7　icut_get_item_faf_sequential

(1)呼出形式

short　icut_get_item_faf_sequential(tag, data, dlen, len, fd)

(2)引き数

```
int *tag;                  /*  タグ  */
unsigned char *data;       /*  データ実体  */
int dlen;                  /*  データバッファの長さ  */
int *len;                  /*  データ長  */
int fd;                    /*  ファイルディスクリプタ  */
```

(3)機能

対象となる基礎ファイルの拡張ファイル領域のデータを読み込む。基礎ファイルはファイルディスクリプタによって指定する。読み出したデータが指定領域にコピーされ、タグとデータ長が返される。

(4)コール関数一覧

コール関数	
icut_get_faf_item_sequential_mem	

3.2.8　icut_load_faf

(1)呼出形式

short　icut_load_faf(dfno)

(2)引き数

int dfno　　/*　DFディスクリプタ　*/

(3)機能

拡張ファイルの内容をバッファ内に読み出す。基礎ファイルはDFディスクリプタによって指定する。

(4)コール関数一覧

コール関数	
icut_read_ef	
icut_get_faf_item_mem	
icut_get_length	

3.2.9　icut_save_faf

(1)呼出形式

short　icut_save_faf(void)

(2)引き数

無し

(3)機能

ICカード上の拡張ファイル領域にデータをセーブする。

(4)コール関数一覧

コール関数	
icut_write_ef	
icut_update_file_info	

3.2.10　icut_define_faf

(1)呼出形式

short　icut_define_faf(dfno)

(2)引き数

int dfno　　/*　DFディスクリプタ　*/

(3)機能

CAMのために拡張ファイル領域を初期化する。DF情報ファイルにも拡張ファイルに関する情報を登録する。対象となる拡張ファイルはDFディスクリプタによって指定する。

(4)コール関数一覧

コール関数	
icut_open_dfi	
icut_init_dfi	
icut_search_file	
icut_init_ef	
icut_register_file_info	
icut_read_ef	
icut_get_faf_item_mem	
icut_append_faf_item_mem	
icut_write_ef	

3.2.11　icut_new_faf_code

(1)呼出形式

short　icut_new_faf_code(code)

(2)引き数

int *code　　/*　EF識別コード　*/

(3)機能

拡張ファイル領域において利用されていないEF識別コードを探索する。利用可能なコードが値として返される。

(4)コール関数一覧

コール関数	
icut_get_faf_item_sequential_mem	

3.2.12　icut_close_faf_if_unused

(1)呼出形式

short　icut_close_faf_if_unused(fd)

(2)引き数

int fd　　/*　ファイルディスクリプタ　*/

(3)機能

オープンされている基礎ファイルの内、ファイルディスクリプタで指定した基礎ファイル以外で拡張ファイルが利用されていない場合に、拡張ファイルをクローズする。

(4)コール関数一覧

コール関数	
icut_close_faf	

3.3　メモリ内のデータ管理に関する内部関数

3.3.1　icut_get_item_sequential_mem

(1)呼出形式

```
short    icut_get_item_sequential_mem( tag, data, dlen, len, bufptr, blen, stt, g_flg )
```

(2)引き数

```
int *tag;                /*  タグ  */
unsigned char *data;     /*  データ実体  */
int dlen;                /*  データバッファの長さ  */
int *len;                /*  データ長  */
unsigned char *bufptr;   /*  バッファポインタ  */
int blen;                /*  バッファ長  */
int *stt;                /*  スタートポイント  */
int *g_flg;              /*  グループアイテム用フラグ  */
```

(3)機能

バッファから対象ファイルのデータを項目単位で順次読み込む。バッファはバッファポインタ、バッファ長、スタートポイントによって指定する。読み出したデータが指定領域にコピーされ、タグ、データ長、データ実体が返される。また、スタートポイントには現在位置がバイト単位で返される。

(4)コール関数一覧

コール関数	
icut_get_tagcode	
icut_get_length	

3.3.2　icut_get_item_mem

(1)呼出形式

```
short    icut_get_item_mem( tag, data, dlen, len, bufptr, blen, stt, g_flg )
```

(2)引き数

```
int tag;                  /*  タグ  */
unsigned char *data;      /*  データ実体  */
int dlen;                 /*  データバッファの長さ  */
int *len;                 /*  データ長  */
unsigned char *bufptr;    /*  バッファポインタ  */
int blen;                 /*  バッファ長  */
int *stt;                 /*  スタートポイント  */
int *g_flg;               /*  グループアイテム用フラグ  */
```

(3)機能

バッファから対象ファイルの特定データを読み込む。バッファはバッファポインタ、バッファ長、スタートポイントによって指定する。読み出すデータはタグによって指定する。読み出したデータが指定領域にコピーされる。

(4)コール関数一覧

コール関数	
icut_get_item_sequential_mem	

3.3.3　icut_append_item_mem

(1)呼出形式

short　icut_append_item_mem(tag, data, len, alen, bufptr, blen, g_flg)

(2)引き数

```
int tag;                /*  タグ  */
unsigned char *data;    /*  データ実体  */
int len;                /*  データ長  */
int *alen;              /*  データ実長  */
unsigned char *bufptr;  /*  バッファポインタ  */
int blen;               /*  バッファ長  */
int g_flg;              /*  グループアイテム用フラグ  */
```

(3)機能

バッファに対象ファイル領域の新規データを追加する。バッファはバッファポインタ、バッファ長によって指定する。追加するデータはタグ、データ長、データへのポインタによって指定する。

(4)コール関数一覧

コール関数	
icut_get_item_mem	
icut_set_tagcode	
icut_set_length	

3.3.4　icut_replace_item_mem

(1)呼出形式

short　icut_replace_item_mem(tag, data, len, bufptr, blen, g_flg)

(2)引き数

```
int tag;                  /*  タグ  */
unsigned char *data;      /*  データ実体  */
int len;                  /*  データ長  */
unsigned char *bufptr;    /*  バッファポインタ  */
int blen;                 /*  バッファ長  */
int g_flg;                /*  グループアイテム用フラグ  */
```

(3)機能

バッファ内の対象ファイルのデータを入れ換える。バッファはバッファポインタ、バッファ長によって指定する。入れ換えるデータはタグ、データ長、データへのポインタによって指定する。ただし、データを入れ換える場合には、存在する２つのデータ長が同一でなければならない。

(4)コール関数一覧

コール関数	
icut_get_item_mem	
icut_count_tagcode	
icut_set_length	
icut_count_length	

3.3.5　icut_remove_item_mem

(1)呼出形式

```
short    icut_remove_item_mem( tag, alen, bufptr, blen )
```

(2)引き数

```
int tag;                    /*  タグ  */
int *alen;                  /*  データ実長  */
unsigned char *bufptr;      /*  バッファポインタ  */
int blen;                   /*  バッファ長  */
```

(3)機能

バッファ内の対象ファイルのデータを消去する。バッファはバッファポインタ、バッファ長によって指定する。データはタグによって指定する。削除されたデータの実長が返される。

(4)コール関数一覧

コール関数	
icut_get_item_mem	
icut_get_item_sequential_mem	

3.3.6　icut_get_faf_item_sequential_mem

(1)呼出形式

short　icut_get_faf_item_sequential_mem(efc, tag, data, dlen, len, bufptr, blen, stt, g_flg)

(2)引き数

```
int *efc;                /*  拡張ファイル用EF識別コード  */
int *tag;                /*  タグ  */
unsigned char *data;     /*  データ実体  */
int dlen;                /*  データバッファの長さ  */
int *len;                /*  データ長  */
unsigned char *bufptr;   /*  バッファポインタ  */
int blen;                /*  バッファ長  */
int *stt;                /*  スタートポイント  */
int *g_flg               /*  グループアイテム用フラグ  */
```

(3)機能

バッファから拡張ファイル形式のデータを順次読み出す。バッファはバッファポインタ、バッファ長、スタートポイントによって指定する。読み出したデータが指定領域にコピーされ、拡張ファイル用EF識別コード、タグ、データ長、データ実体が返される。

(4)コール関数一覧

コール関数	
icut_get_tagcode	
icut_get_length	

3.3.7　icut_get_faf_item_mem

(1)呼出形式

short　icut_get_faf_item_mem(efc, tag, data, dlen, len, bufptr, blen, stt, g_flg)

(2)引き数

```
int efc;                /*  拡張ファイル用EF識別コード  */
int tag;                /*  タグ  */
unsigned char *data;    /*  データ実体  */
int dlen;               /*  データバッファの長さ  */
int *len;               /*  データ長  */
unsigned char *bufptr;  /*  バッファポインタ  */
int blen;               /*  バッファ長  */
int *stt;               /*  スタートポイント  */
int *g_flg;             /*  グループアイテム用フラグ  */
```

(3)機能

バッファから拡張ファイル形式のデータを読み出す。バッファはバッファポインタ、バッファ長、スタートポイントによって指定する。データは拡張ファイル用EF識別コードとタグによって指定する。読み出したデータが指定領域にコピーされる。

(4)コール関数一覧

コール関数	
icut_get_faf_item_sequential_mem	

3.3.8　icut_append_faf_item_mem

(1)呼出形式

short　icut_append_faf_item_mem(efc, tag, data, len, alen, bufptr, blen, g_flg)

(2)引き数

```
int efc;                /*  拡張ファイル用EF識別コード  */
int tag;                /*  タグ  */
unsigned char *data;    /*  データ実体  */
int len;                /*  データ長  */
int *alen;              /*  データ実長  */
unsigned char *bufptr;  /*  バッファポインタ  */
int blen;               /*  バッファ長  */
int g_flg;              /*  グループアイテム用フラグ  */
```

(3)機能

バッファに拡張ファイルのデータを追加する。バッファはバッファポインタ、バッファ長によって指定する。データは拡張ファイル用EF識別コード、タグ、データ長、データへのポインタによって指定する。

(4)コール関数一覧

コール関数	
icut_get_faf_item_mem	
icut_set_tagcode	
icut_set_length	

3.3.9　icut_replace_faf_item_mem

(1)呼出形式

short　icut_replace_faf_item_mem(efc, tag, data, len, bufptr, blen, g_flg)

(2)引き数

```
int efc;                /*  拡張ファイル用EF識別コード  */
int tag;                /*  タグ  */
unsigned char *data;    /*  データ実体  */
int len;                /*  データ長  */
unsigned char *bufptr;  /*  バッファポインタ  */
int blen;               /*  バッファ長  */
int g_flg;              /*  グループアイテム用フラグ  */
```

(3)機能

バッファにおいて拡張ファイルのデータを入れ換える。バッファはバッファポインタ、バッファ長によって指定する。データは拡張ファイル用EF識別コード、タグ、データ長、データへのポインタによって指定する。

(4)コール関数一覧

コール関数	
icut_get_faf_item_mem	
icut_count_tagcode	
icut_count_length	
icut_set_length	

3.3.10　icut_remove_faf_item_mem

(1)呼出形式

```
short    icut_remove_faf_item_mem( efc, tag, len, bufptr, blen )
```

(2)引き数

```
int efc;                  /*  拡張ファイル用EF識別コード  */
int tag;                  /*  タグ  */
int *len;                 /*  データ長  */
unsigned char *bufptr;    /*  バッファポインタ  */
int blen;                 /*  バッファ長  */
```

(3)機能

バッファにおいて拡張ファイルのデータを削除する。バッファはバッファポインタ、バッファ長によって指定する。データは拡張ファイル用EF識別コード、タグ、データ長によって指定する。

(4)コール関数一覧

コール関数	
icut_get_faf_item_mem	
icut_get_faf_item_sequential_mem	

3.4　ユーティリティ内部関数

3.4.1　icut_append_item

(1)呼出形式

```
short    icut_append_item( tag, data, len, fd )
```

(2)引き数

```
int tag;                    /*   タグ   */
unsigned char *data;        /*   データ実体   */
int len;                    /*   データ長   */
int fd;                     /*   ファイルディスクリプタ   */
```

(3)機能

通常のファイル領域にデータを追加する。ファイルはファイルディスクリプタによって指定する。データはタグ、データ長、データへのポインタによって指定する。

(4)コール関数一覧

コール関数	
icut_append_item_mem	

3.4.2　icut_replace_item

(1)呼出形式

```
short    icut_replace_item( tag, data, len, fd )
```

(2)引き数

```
int tag;                    /*  タグ  */
unsigned char *data;        /*  データ実体  */
int len;                    /*  データ長  */
int fd;                     /*  ファイルディスクリプタ  */
```

(3)機能

通常のファイル領域のデータを入れ換える。ファイルはファイルディスクリプタによって指定する。データはタグ、データ長、データへのポインタによって指定する。ただし、データを入れ換える場合には、存在する2つのデータ長が同一でなければならない。

(4)コール関数一覧

コール関数	
icut_replace_item_mem	

3.4.3　icut_remove_item

(1)呼出形式

short　icut_remove_item(tag, fd)

(2)引き数

int tag;　　/*　タグ　*/

int fd;　　/*　ファイルディスクリプタ　*/

(3)機能

通常のファイル領域のデータを消去する。ファイルはファイルディスクリプタによって指定する。データはタグによって指定する。

(4)コール関数一覧

コール関数	
icut_remove_item_mem	

3.4.4　icut_get_item

(1)呼出形式

```
short    icut_get_item( tag, data, dlen, len, fd )
```

(2)引き数

```
int tag;                    /*  タグ  */
unsigned char *data;        /*  データ実体  */
int dlen;                   /*  データバッファの長さ  */
int *len;                   /*  データ長  */
int fd;                     /*  ファイルディスクリプタ  */
```

(3)機能

通常のファイル領域のデータを読み込む。ファイルはファイルディスクリプタによって指定する。読み出すデータはタグによって指定する。読み出したデータが指定領域にコピーされる。

(4)コール関数一覧

コール関数	
icut_get_item_mem	

3.4.5　icut_get_item_sequential

(1)呼出形式

```
short    icut_get_item_sequential( tag, data, dlen, len, fd )
```

(2)引き数

```
int *tag;                  /*  タグ  */
unsigned char *data;       /*  データ実体  */
int dlen;                  /*  データバッファの長さ  */
int *len;                  /*  データ長  */
int fd;                    /*  ファイルディスクリプタ  */
```

(3)機能

通常のファイル領域のデータをファイル単位で順次読み込む。ファイルはファイルディスクリプタによって指定する。読み出したデータが指定領域にコピーされ、タグ、データ長、データ実体が返される。

(4)コール関数一覧

コール関数	
icut_get_item_sequential_mem	

3.4.6　icut_load_ef_to_buffer

(1)呼出形式

short　icut_load_ef_to_buffer(fd)

(2)引き数

int fd;　　/*　ファイルディスクリプタ　*/

(3)機能

ファイルの内容をバッファ内に読み出す。ただし、拡張ファイルにデータが存在する場合には拡張ファイル内のデータもロードする。ファイルはファイルディスクリプタによって指定する。

(4)コール関数一覧

コール関数	
icut_read_ef	
icut_get_item_sequential_mem	

3.4.7　icut_write_ef

(1)呼出形式

short　icut_write_ef(id, fname, nlen, bufptr, len, rec_len)

(2)引き数

```
short id;                  /*  実EF-ID  */
char *fname;               /*  DF名  */
int nlen                   /*  DF名長  */
unsigned char *bufptr;     /*  バッファポインタ  */
int len;                   /*  データ長  */
int rec_len;               /*  記録レコード長  */
```

(3)機能

バッファ内のデータを基礎ファイルに書き込む。対象となる基礎ファイルはDF名と実EF-IDによって指定する。データはバッファへのポインタとデータ長によって指定する。

(4)コール関数一覧

コール関数	
icd_select_DF	
icd_update_record	
icut_match_name	
icut_verify_df	
icut_error_code	

3.4.8　icut_read_ef

(1)呼出形式

```
short    icut_read_ef( id, fname, nlen, bufptr, blen, len, rec_len )
```

(2)引き数

```
short id;                  /*  実EF-ID   */
char *fname;               /*  DF名   */
int nlen                   /*  DF名長   */
unsigned char *bufptr;     /*  バッファポインタ   */
int blen;                  /*  バッファ長   */
int *len;                  /*  データ長   */
int *rec_len;              /*  記録レコード長   */
```

(3)機能

バッファへ格納するデータを基礎ファイルより読み出す。対象となる基礎ファイルはDF名と実EF-IDによって指定する。データはバッファポインタによって指定した領域にコピーされ、コピーしたデータのデータ長、レコード長がバイト数で返される。

(4)コール関数一覧

コール関数	
icd_select_DF	
icd_read_record	
icut_match_name	
icut_verify_df	
icut_error_code	

3.4.9　icut_erase_ef

(1)呼出形式

short　icut_erase_ef(id, fname, nlen)

(2)引き数

short id;　　/*　実EF-ID　*/

char *fname;　　/*　DF名　*/

int nlen　　/*　DF名長　*/

(3)機能

基礎ファイル内の内容を全て消去する。対象となる基礎ファイルはDF名と実EF-IDによって指定する。

(4)コール関数一覧

コール関数	
icd_select_DF	
icd_erase_all_record	
icut_match_name	
icut_verify_df	
icut_error_code	

3.4.10　icut_clear_ef

(1)呼出形式

short　icut_clear_ef(id, fname, nlen, rec_len)

(2)引き数

short id;　　/*　実EF-ID　*/

char *fname;　　/*　DF名　*/

int nlen　　/*　DF名長　*/

int rec_len;　　/*　記録レコード長　*/

(3)機能

基礎ファイルにおいて「CAMバージョン番号と基礎ファイルサイズ」以外の全セクタを初期化する。対象となる基礎ファイルはDF名と実EF-IDによって指定する。

(4)コール関数一覧

コール関数	
icd_select_DF	
icut_match_name	
icut_verify_df	
icd_update_record	
icut_error_code	

3.4.11　icut_define_file

(1)呼出形式

short　icut_define_file(id, fname, nlen)

(2)引き数

short id;　　/*　実EF-ID　*/

char *fname;　　/*　DF名　*/

int nlen　　/*　DF名長　*/

(3)機能

CAMのために基礎ファイルを初期化する。DF情報ファイルにも対象となる基礎ファイルを登録する。対象となる基礎ファイルはDF名と実EF-IDによって指定する。

(4)コール関数一覧

コール関数	
icut_read_ef	
icut_get_item_mem	
icut_append_item_mem	
icut_write_ef	
icut_init_ef	
icut_register_file_info	
icut_open_dfi	

3.4.12　icut_init_ef

(1)呼出形式

short　icut_init_ef(id, fname, nlen, type, nbrw)

(2)引き数

short id;　　/*　実EF-ID　*/

char *fname;　　/*　DF名　*/

int nlen　　/*　DF名長　*/

int type;　　/*　ファイル種類　*/

int *nbrw;　　/*　書き込み可能回数　*/

(3)機能

対象となる基礎ファイルがCAM用に初期化されているか否かチェックし、初期化されていなければ初期化する。DF情報ファイルにも基礎ファイルを登録する。ファイル種類はヘッダファイル内で定義されており、以下の通りである。なお、基礎ファイルはDF名と実EF-IDによって指定する。

ファイル種類	ファイルタイプ
DF情報ファイル	DF_INFO_TYPE
基礎ファイル	FILE_TYPE
拡張ファイル	FAF_TYPE
キーテーブルファイル	KEY_FILE_TYPE
アクセスコントロールテーブルファイル	ACESS_TABLE_TYPE
所有者情報ファイル	OWNER_INFO_TYPE

(4)コール関数一覧

コール関数	
icut_read_ef	
icut_error_code	
icut_get_item_mem	
icut_get_faf_item_mem	

3.4.13　icut_get_length

(1)呼出形式

```
short    icut_get_length( ptr, len, dlen, g_flg )
```

(2)引き数

```
unsigned char *ptr;        /*   データのポインタ   */
int *len;                  /*   データ長   */
int *dlen;                 /*   データバッファの長さ   */
int *g_flg;                /*   グループアイテム用フラグ   */
```

(3)機能

データレコードからデータ長を求める。

(4)コール関数一覧

無し

3.4.14　icut_set_length

(1)呼出形式

```
short    icut_set_length( ptr, len, dlen, g_flg )
```

(2)引き数

```
unsigned char *ptr;        /*   データのポインタ   */
int len;                   /*   データ長   */
int *dlen;                 /*   データバッファの長さ   */
int g_flg;                 /*   グループアイテム用フラグ   */
```

(3)機能

データレコードにデータ長を設定する。

(4)コール関数一覧

無し

3.4.15　icut_get_tagcode

(1)呼出形式

```
short    icut_get_tagcode( ptr, tag, dlen )
```

(2)引き数

```
unsigned char *ptr;        /*   データのポインタ   */
int *tag;                  /*   タグ   */
int *dlen;                 /*   データバッファの長さ   */
```

(3)機能

データレコードからタグ番号を求める。

(4)コール関数一覧

無し

3.4.16　icut_set_tagcode

(1)呼出形式

```
short    icut_set_tagcode( ptr, tag, dlen )
```

(2)引き数

```
unsigned char *ptr;        /*   データのポインタ   */
int tag;                   /*   タグ   */
int *dlen;                 /*   データバッファの長さ   */
```

(3)機能

データレコードにタグ番号を設定する。

(4)コール関数一覧

無し

3.4.17　icut_count_length

(1)呼出形式

```
short    icut_count_length( len, dlen, g_flg )
```

(2)引き数

```
int len;                /*   データ長   */
int *dlen;              /*   データバッファの長さ   */
int g_flg;              /*   グループアイテム用フラグ   */
```

(3)機能

データ長領域のサイズを求める。

(4)コール関数一覧

無し

3.4.18　icut_count_tagcode

(1)呼出形式

```
short    icut_count_tagcode( tag, dlen )
```

(2)引き数

```
int tag;                    /*   タグ   */
int *dlen;                  /*   データバッファの長さ   */
```

(3)機能

タグ番号の領域サイズを求める。

(4)コール関数一覧

無し

3.4.19　icut_file_writable

(1)呼出形式

short　icut_file_writable(id, fname, nlen)

(2)引き数

short id;　　/*　実EF-ID　*/

char *fname;　　/*　DF名　*/

int nlen;　　/*　DF名長　*/

(3)機能

基礎ファイルへの書き込みに関する権限チェックを先頭の一レコードによって行う。また、通常ファイルが拡張ファイルを使用している場合に限り、ファイルオープン時に拡張ファイルのアクセス権チェックを行う。通常ファイルに対して「Write」権限があっても、拡張ファイルに対して「Write」権限がないと「WriteOnly」「ReadWrite」モードでのオープンはエラーとなる。拡張ファイルを使用していないファイルについては、ファイルオープン時のチェックを行わない。

(4)コール関数一覧

コール関数	
icd_select_DF	
icut_match_name	
icut_verify_df	
icd_read_record	
icd_update_record	
icut_error_code	

3.4.20　icut_get_status

(1)呼出形式

short　icut_get_status(void)

(2)引き数

無し

(3)機能

ステータスをチェックする。

(4)コール関数一覧

コール関数	
icd_get_status	

3.4.21　icut_match_name

(1)呼出形式

```
short    icut_match_name( n1, len1, n2, len2 )
```

(2)引き数

```
char *n1;              /*   第1番目の文字列    */
int len1;              /*   第1番目の文字列の長さ    */
char *n2;              /*   第2番目の文字列    */
int len2;              /*   第2番目の文字列の長さ    */
```

(3)機能

2つの文字列の比較を行う。

(4)コール関数一覧

無し

3.4.22　icut_copy_name

(1)呼出形式

```
short    icut_copy_name( n1, n2, n )
```

(2)引き数

```
char *n1;                /*   第1番目の文字列   */
char *n2;                /*   第2番目の文字列   */
int n;                   /*   コピーする文字列   */
```

(3)機能

一つの文字列から別の文字列へ文字列のコピーを行う。

(4)コール関数一覧

無し

3.4.23　icut_error_code

(1)呼出形式

short　icut_error_code(er)

(2)引き数

short er;　　/*　エラーコード　*/

(3)機能

ステータスコードからエラーコードを得る。

(4)コール関数一覧

コール関数	
icut_get_status	

3.5　循環型基礎ファイル管理に関する内部関数

3.5.1　icut_is_rotation_file

(1)呼出形式

```
short    icut_is_otation_file( dfno, tid )
```

(2)引き数

```
int dfno;                /*   DFディスクリプタ   */
int tid;                 /*   論理的なEF-ID   */
```

(3)機能

icut_open_dfi関数でオープンされているDF内のtidで指定された論理的なEF-IDを持つ基礎ファイルが循環型基礎ファイルであるか否かを調べる。dfnoにDFディスクリプタを指定する。ここで、循環型基礎ファイルとは、同一の論理的なEF-IDを持つ基礎ファイルが2つ以上存在するファイルを意味する。

同一の論理的なEF-IDを持つ基礎ファイルが2つ以上存在する場合、本関数はリターン値としてTRUE（非0）を返す。指定されたDF内に論理的なEF-IDがtidである基礎ファイルが唯一の場合、または論理的なEF-IDがtidである基礎ファイルが登録されていない場合、本関数はリターン値としてFALSE（0）を返す。

(4)コール関数一覧

無し

3.5.2　icut_get_actual_efid

(1)呼出形式

```
short    icut_get_actual_efid( dfno, tid, odr, id )
```

(2)引き数

```
int dfno;              /*   DFディスクリプタ   */
short tid;             /*   論理的なEF-ID   */
int odr;               /*   循環型基礎ファイル番号   */
short *id;             /*   実EF-ID   */
```

(3)機能

icut_open_df関数でオープンされているDF内のtidで指定された論理的なEF-IDを持つ基礎ファイルの実EF-IDを得る。dfnoにDFディスクリプタを指定する。odrに循環型基礎ファイル番号を指定する。

(4)コール関数一覧

無し

3.5.3　icut_get_order_v0100

(1)呼出形式

short　icut_get_order_v0100(fd, opt, odr, dfno, mul)

(2)引き数

```
int fd;          /*  ファイルディスクリプタ  */
int opt;         /*  入力としての循環型基礎ファイル番号  */
int *odr;        /*  出力としての循環型基礎ファイル番号  */
int *nbr;        /*  基礎ファイル数  */
int *mul;        /*  エレメント内のデータ数  */
```

(3)機能

CAM　Version1.0で記録された、自治省が主導する地域カードシステムの循環型基礎ファイルの情報を得る。ファイルはfd及び循環型基礎ファイル番号optで指定する。指定されたopt番目に新しい循環型基礎ファイルが何番目に記録されているかを示す番号odr、循環型基礎ファイル数nbr、各エレメント内のデータ数mulを返す。

(4)コール関数一覧

コール関数	
icut_get_item_mem	
icut_get_faf_item_mem	

3.6　複数個のデータ実体から構成されるデータ管理に関する内部関数

3.6.1　icut_check_group_item_data

(1)呼出形式

```
short    icut_check_group_item_data( idat, glen )
```

(2)引き数

```
unsigned char *idat;        /*    複数個のデータ実体を持つデータへのポインタ    */
int glen;                   /*    複数個のデータ実体を持つデータ全長    */
```

(3)機能

内部利用するための複数個のデータ実体を持つデータに関するフォーマットをチェックする。

(4)コール関数一覧

コール関数	
icut_get_length	

3.7　アクセス制御に関する内部関数

3.7.1　icut_verify_df

(1)呼出形式

```
int    icut_verify_df( fname, nlen, status )
```

(2)引き数

```
char *fname;              /*   DF名   */
int nlen;                 /*   DF名長   */
unsigned short *status;   /*   リーダ・ライタまたはICカードのステータス   */
```

(3)機能

対象DFのキー情報をベリファイする。

(4)コール関数一覧

コール関数	
icut_get_DF_no_access_info	
icd_verify	
icut_error_code	

3.7.2　icut_verify_item

(1)呼出形式

int　icut_verify_item(fname, nlen, id, tag, rw)

(2)引き数

```
char *fname;            /*  DF名  */
int nlen;               /*  DF名長  */
unsigned short id;      /*  実EF-ID  */
int tag;                /*  タグ  */
int *rw;                /*  アクセス権情報  */
```

(3)機能

対象基礎ファイル内のデータのアクセス権をベリファイする。ベリファイするデータはDF名、実EF-ID、タグによって指定する。

(4)コール関数一覧

コール関数	
icut_get_DF_no_access_info	
icut_match_name	

3.7.3　icut_init_operation_info

(1)呼出形式

int　icut_init_operation_info(n_df)

(2)引き数

int n_df;　　/*　バッファで取り扱うDFの最大数　*/

(3)機能

　オペレーションカードから読み込まれ、ICカードへのアクセス時に使用されるアクセスコントロールテーブル、所有者情報、キーテーブルを格納するための領域及びパラメータを準備する。

(4)コール関数一覧

無し

3.7.4　icut_init_DF_access_info

(1)呼出形式

```
int    icut_init_DF_access_info( fname, nlen, n_ef, n_df )
```

(2)引き数

```
char *fname;            /*   DF名    */
int nlen;               /*   DF名長   */
int n_ef;               /*   バッファで取り扱うEFの最大数   */
int *n_df;              /*   バッファで取り扱うDFの最大数   */
```

(3)機能

　オペレーションカードから読み込まれ、ICカードへのアクセス時に使用されるアクセスコントロールテーブル、所有者情報、キーテーブルを格納するためのDF毎の領域及びパラメータを準備する。

(4)コール関数一覧

無し

3.7.5　icut_get_DF_no_access_info

(1)呼出形式

int　icut_get_DF_no_access_info(fname, nlen, n_ef, n_df)

(2)引き数

char *fname;　　/*　DF名　*/

int nlen;　　/*　DF名長　*/

int *n_df;　　/*　バッファで取り扱うDFの最大数　*/

(3)機能

icut_init_DF_access_info関数で確保されたDF毎のアクセスコントロールテーブル、所有者情報、キーテーブルを格納した領域を指定するためのパラメータであるn_dfを検索する。

(4)コール関数一覧

コール関数	
icut_match_name	

3.7.6　icut_clear_operation_info

(1)呼出形式

```
int　icut_clear_operation_info( void )
```

(2)引き数

無し

(3)機能

バッファ内のアクセス権情報をクリアする。

(4)コール関数一覧

無し

3.7.7　icut_key_data_len

(1)呼出形式

```
int    icut_key_data_len( kdt, n_key )
```

(2)引き数

```
char *kdt;                  /*   キーデータ列   */
int n_key;                  /*   キー数   */
```

(3)機能

キーデータ列のバッファ長を計算する。

(4)コール関数一覧

無し

3.7.8　icut_set_key_info

(1)呼出形式

```
int    icut_set_key_info( n_df, kdt, n_key )
```

(2)引き数

```
int *n_df;              /*　バッファで取り扱うDFの最大数　*/
char *kdt;              /*　キーデータ列　*/
int n_key;              /*　キー数　*/
```

(3)機能

キーテーブルをセットする。

(4)コール関数一覧

コール関数	
icut_key_data_len	

3.7.9　icut_set_operation_card_key

(1)呼出形式

int　icut_set_operation_card_key(void)

(2)引き数

無し

(3)機能

オペレーションカード内のキーテーブルをバッファに設定する。

(4)コール関数一覧

無し

3.7.10　icut_set_operation_card_table

(1)呼出形式

```
int　icut_set_operation_card_table( n_df )
```

(2)引き数

```
int n_df;                    /*　バッファで取り扱うDFの最大数　*/
```

(3)機能

オペレーションカード内のアクセスコントロールテーブルをバッファに設定する。

(4)コール関数一覧

無し

3.7.11　icut_set_additional_key

(1)呼出形式

```
int　icut_set_additional_key( fname, nlen, keyid, keydata, klen )
```

(2)引き数

```
char *fname;　　/*　DF名　*/
int nlen;　　/*　DF名長　*/
int keyid;　　/*　キーID　*/
int *keydata;　　/*　キーデータ　*/
int klen;　　/*　キーデータ長　*/
```

(3)機能

キーテーブルに一時的な鍵を追加する。

(4)コール関数一覧

無し

3.7.12　icut_clear_additional_key

(1)呼出形式

int　icut_clear_additional_key(void)

(2)引き数

無し

(3)機能

設定された一時的な鍵を消去する。

(4)コール関数一覧

無し

4. デバイスドライバ共通インタフェース関数を制御するための内部関数

4.1 概要

デバイスドライバ共通インタフェース関数を制御するための内部関数は、CAM標準ソフトウェアVersion2.0を構成する最下層に位置する内部関数である。

従って、デバイスドライバ共通インタフェース関数を制御するための内部関数内でコールされている表4-1-1に示す関数は、CAMVersion2.0において規定したデバイスドライバ共通インタフェース関数であり、各関数に関する機能・仕様等は【本編】に詳述する。

表4-1-1　デバイスドライバ共通インタフェース関数一覧

関数名	機能概要
std_com_open	ICカードドライバをオープンする。また、対応可能なICカードの種類を提示する。
std_com_close	ICカードドライバをクローズする。
std_card_in	ICカードを活性化する。初期応答情報の解析を行い、活性化したICカードの種類を判定する。
std_card_out	ICカードを非活性化し、リーダ・ライタからカードを排出させる。
std_card_reset	ICカードがリーダ・ライタに挿入されているときに、ICカードを再活性化する。
std_card_sense	ICカードがリーダ・ライタのスロット内に挿入されているか否かを確認する。
std_select_DF	SELECT DF（JIS X6306）またはSELECT ADF（S型ICカード）を実行する。
std_verify	VERIFY（JIS X6306）またはVERIFY KEY（S型ICカード）を実行する。
std_read_record	READ RECORD(S)（JIS X6306）またはREAD RECORD(S型ICカード）を実行する。
std_write_record	WRITE RECORD（JIS X6306）またはWRITE RECORD（S型ICカード）を実行する。
std_append_record	APPEND RECORD（JIS X6306）を実行する。
std_update_record	UPDATE RECORD（JIS X6306）を実行する。
std_erase_all_records	ERASE ALL RECORDS（JICSAP滝川市向け仕様）またはERASE ALL RECORDS（S型ICカード）を実行する。
std_other_command	上記以外のコマンドを実行させる場合に使用する。コマンドブロックのインフォメーション部の作成は、本関数の呼び出し側が行う。

4.2　関数詳細

4.2.1　icd_std_open

(1)呼出形式

short　icd_std_open(void)

(2)引き数

無し

(3)機能

デバイスドライバの初期化を行う。

(4)コール関数一覧

コール関数	
std_com_open	

4.2.2　icd_std_close

(1)呼出形式

short　icd_std_close(void)

(2)引き数

無し

(3)機能

RS-232Cポートのクローズを行う。

(4)コール関数一覧

コール関数	
std_com_close	

4.2.3　icd_set_wait_time

(1)呼出形式

```
short    icd_set_wait_time( sec )
```

(2)引き数

```
short sec;                    /*   時間（秒）    */
```

(3)機能

カード挿入待ち時間を秒単位で設定する。

(4)コール関数一覧

無し

4.2.4　icd_card_info

(1)呼出形式

short　icd_card_info(card_type)

(2)引き数

unsigned char *card_type;

(3)機能

挿入されているカードの種類をcard_typeに返す。引数card_typeの形式は、デバイスドライバ共通インタフェース関数 std_card_in() における引数カード仕様識別情報 spec と同一である。

(4)コール関数一覧

無し

4.2.5　icd_read_record

(1)呼出形式

short　icd_read_record(id, recno, data, len)

(2)引き数

short id;　　/*　実EF-ID　*/

unsigned int recno;　　/*　レコード番号　*/

unsigned char *data;　　/*　データ実体　*/

unsigned short *len;　　/*　データ長　*/

(3)機能

基礎ファイルからレコードを読み出す。ただし、DFはic_select_DF関数により、また基礎ファイルは実EF-IDにより、さらにレコードはレコード番号によりそれぞれ指定する。レコード番号は「1」から開始し、「0」を指定した場合には先頭レコードからの順次読み出しとなる。読み出されたデータは指定した領域にコピーされ、転送データ量を返値とする。

(4)コール関数一覧

コール関数	
std_read_record	

4.2.6　icd_write_record

(1)呼出形式

short　icd_write_record(id, recno, data, len)

(2)引き数

```
short id;                /*  実EF-ID  */
unsigned int recno;      /*  レコード番号  */
unsigned char *data;     /*  データ実体  */
unsigned short len;      /*  データ長  */
```

(3)機能

基礎ファイルにレコードを書き込む。ただし、DFはic_select_DF関数により、また基礎ファイルは実EF-IDにより、さらにレコードはレコード番号によりそれぞれ指定する。レコード番号は「1」から開始し、「0」を指定した場合には先頭レコードからの順次書き込みとなる。書き込むデータはデータ領域へのポインタとデータ長によって指定する。

(4)コール関数一覧

コール関数	
std_write_record	

4.2.7　icd_append_record

(1)呼出形式

```
short    icd_append_record( id, data, len )
```

(2)引き数

```
short id;                /*   実EF-ID   */
unsigned char *data;     /*   データ実体   */
unsigned short len;      /*   データ長   */
```

(3)機能

基礎ファイルにレコードを追加する。ただし、DFはic_select_DF関数により、また基礎ファイルは実EF-IDによりそれぞれ指定する。書き込むデータはデータ領域へのポインタとデータ長によって指定する。

(4)コール関数一覧

コール関数	
std_append_record	

4.2.8　icd_update_record

(1)呼出形式

short　icd_update_record(id, recno, data, len)

(2)引き数

short id;　　/*　実EF-ID　*/

unsigned int recno;　　/*　レコード番号　*/

unsigned char *data;　　/*　データ実体　*/

unsigned short len;　　/*　データ長　*/

(3)機能

基礎ファイル内のレコードを更新する。ただし、DFはic_select_DF関数により、また基礎ファイルは実EF-IDにより、さらにレコードはレコード番号によりそれぞれ指定する。更新するデータはデータ領域へのポインタとデータ長によって指定する。

(4)コール関数一覧

コール関数	
std_update_record	

4.2.9　icd_erase_all_record

(1)呼出形式

short　icd_erase_all_record(id)

(2)引き数

short id;　　/*　実EF-ID　*/

(3)機能

基礎ファイル内の全レコードを消去する。ただし、DFはic_select_DF関数により指定する。

(4)コール関数一覧

コール関数	
std_erase_all_record	

4.2.10　icd_select_DF

(1)呼出形式

```
short    icd_select_DF( fname, nlen )
```

(2)引き数

```
char *fname;              /*   DF名   */
int nlen;                 /*   DF名長   */
```

(3)機能

対象であるDFを選択する。ただし、DFに関するキー情報のベリファイも併せて行う。

(4)コール関数一覧

コール関数	
std_select_DF	

4.2.11　icd_verify

(1)呼出形式

```
int    icd_verify( kdt, n_key )
```

(2)引き数

```
char *kdt;              /*   キーデータ列   */
int n_key;              /*   キー数   */
```

(3)機能

DFに関するキーを照合する。

(4)コール関数一覧

コール関数	
std_verify	

4.2.12　icd_in_card

(1)呼出形式

short　icd_in_card(void)

(2)引き数

無し

(3)機能

カードを挿入する。

(4)コール関数一覧

コール関数	
std_card_in	
std_card_reset	
std_card_sense	

4.2.13　icd_out_card

(1)呼出形式

short　icd_out_card(void)

(2)引き数

無し

(3)機能

カードを排出する。

(4)コール関数一覧

コール関数	
std_card_out	
std_card_sense	

4.2.14　icd_get_status

(1)呼出形式

int　icd_get_status(status)

(2)引き数

unsigned short *status　/*　リーダ・ライタまたはICカードのステータス　*/

(3)機能

ステータスを調べる。

(4)コール関数一覧

無し

4.2.15　icd_show_status

(1)呼出形式

int　icd_show_status(void)

(2)引き数

無し

(3)機能

ステータスを印刷する。

(4)コール関数一覧

無し

5. デバイスドライバ共通インタフェース関数

【本編】を参照されたい。

6. 伝送制御関数

CAM標準ソフトウェアVersion2.0において開発したWiodows3.1を対象としたJIS準拠ICカード対応デバイスドライバ関数に係わる伝送制御関数を以下に示す。

6.1　Windows3.1を対象としたJIS準拠ICカード

Windows3.1を対象とした伝送制御関数では、Windows3.1により提供されているデバイスドライバを制御する関数をコールしているが、これらの各関数に関する機能・仕様等の詳細はMicrosoft Windows3.1プログラマーズリファレンス等を始めとする関連マニュアルを参照されたい。

6.1.1　icjd_open_device

(1)呼出形式

int　icjd_open_device(cwt, bwt)

(2)引き数

int cwt;　　/*　コマンドリトライ時間（秒）　*/

int bwt;　　/*　BWタイマー値（秒）　*/

(3)機能

通信ポートCOM1をオープンし、通信パラメータをセットする。オープンされた回線のチャンネル番号を返す。

(4)コール関数一覧

コール関数	
OpenComm	
BuildCommDCB	
icjd_set_comm_param	

6.1.2　icjd_set_comm_param

(1)呼出形式

```
int    icjd_set_comm_param( ich, cwt, bwt )
```

(2)引き数

```
int ich;                /*   オープンされた回線のチャンネル番号   */
int cwt;                /*   コマンドリトライ時間（秒）    */
int bwt;                /*   BWタイマー値（秒）    */
```

(3)機能

ICカードと通信するための通信パラメータをセットする。

(4)コール関数一覧

コール関数	
SetCommState	

6.1.3　icjd_close_device

(1)呼出形式

int　icjd_close_device(ich)

(2)引き数

int ich;　　/*　オープンされた回線のチャンネル番号　*/

(3)機能

icjd_open_device関数でオープンされたCOM1をクローズする。

(4)コール関数一覧

コール関数	
CloseComm	

6.1.4　icjd_send

(1)呼出形式

```
int    icjd_send( ich, nad, pcb, len, inf, buf )
```

(2)引き数

```
int ich;                /*   オープンされた回線のチャンネル番号   */
unsigned char nad;      /*   ノードアドレス（相手局<<4｜自局）   */
unsigned char pcb       /*   パラメータコントロールブロック   */
unsigned char len       /*   送信データの長さ   */
unsigned char *inf      /*   送信データ   */
unsigned char *buf      /*   通信用バッファのポインタ   */
```

(3)機能

通信ポートを介してデータをICカードまたはリーダ・ライタに送信する。

(4)コール関数一覧

コール関数	
WriteComm	
GetCommError	

6.1.5　icjd_recieve

(1)呼出形式

int　icjd_recieve(ich, buf)

(2)引き数

int ich;　　/*　オープンされた回線のチャンネル番号　*/

unsigned char *buf　　/*　通信用バッファのポインタ　*/

(3)機能

ICカードまたはリーダ・ライタから送られてきたデータを受信する。

(4)コール関数一覧

コール関数	
ReadComm	
GetCommError	

平成7年度

電子メディアに関する調査研究

内容アクセスマネージャの標準化に関する調査研究 報告書

平成8年3月

発行所　財団法人ニューメディア開発協会

〒107　東京都港区三田1丁目4番28号
三田国際ビルヂング23階

TEL　03-3457-0671