ネットワークカメラ設定管理API標準GenICam

大野邦夫
株式会社安土
ohno@azuchi.net

1. はじめに

GenICamは、ネットワークを経由してカメラを制御する標準のプロトコルとインタフェースである。さらにその設定にはXMLを用いることに特徴がある。私はハードウエアとしてのカメラに関しては素人であるが、ネットワークやXMLに関しては多少の素養があるので、後者の観点から本稿ではGenICamについて紹介する。

ネットワークカメラの今後の活用には興味を持たざるを得ない。最近もオウムの指名手配容疑者逮捕に監視カメラの映像・画像が有効に使用されたが、今後は社会生活にネットワークカメラが広く導入されることは確実であろう。プライバシーの侵害などの問題はあるが、安全・安心な社会のための必要なシステムとしての認識と社会的な合意が形成されつつあると思われる。

GenICamは、そのようなシステム構築のための基本的なインタフェースとなる可能性が高い。本稿では、インタフェースの仕様書[1]とトランスポート層の仕様書[2]に基づき、その概要を紹介する。

2. インタフェース仕様

2.1 基本的考え方

インタフェース仕様は、GenICam Standard Generic Interface for Cameras Version 2.0に記載されている。ここではその概要を解説する。

最近のデジタルカメラは、単に画像の配信に留まらず多様な機能を提供する。画像処理、画像データストリームへの追加情報の付加、外部ハードウエアの制御、アプリケーションのリアルタイム実行などが、デジタルカメラの遠隔制御のための一般的な課題になっている。その結果、カメラへのインタフェースは高度なものになってきた。

GenICamのゴールは全てのカメラに対し汎用的なプログラミングインタフェースを提供することにある。インタフェース技術が何であれ、実装機能がどうあれ、多様なニーズに基づくサービスのためのアプリケーションに対してAPIは常に同一であるべきである。図1は、カメラ構成における階層を示す。ア

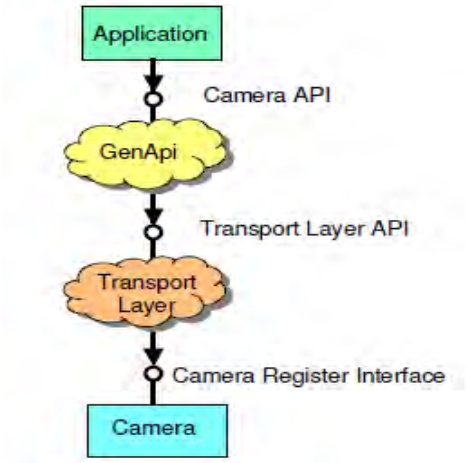

図1　カメラへのアクセスAPI

プリケーションはカメラAPIを要求し、それでカメラの機能を操作する。

GenICamにおいて、カメラの機能はフラットなレジスタの集合として扱われる。アプリケーションは、C言語的なコードで記述されることを想定している。例えば以下のようなコードを記述する。

```
Camera.Gain = 42;
```

GenApiモジュールは、トランスポート層APIで提供されるアクセス関数を呼び出して下記のようなレジスタ呼び出しの縦列命令に翻訳する。

```
TransportLayer.WriteRegister( 0x0815, 0x2A, 2 ); // address, data, length
```

最後に、トランスポーと層は、呼び出しをカメラインタフェースに伝達する。

GenICam規格は、XMLによるカメラ記述ファイルの仕様とトランスポート層の意味的対応を定義すると共に、機能についての標準的な名称（name）と型（type）を定めている。

2.2 カメラ記述ファイルの基本構成

カメラ機能は、XMLファイルで記述され、そのノードは、型とユニークな名称で記述される。ノードは互いにリンクとして関係付けられ、個々のリンクは特有のロールを持つ。図2は、簡単なグラフ記法（notation）の例である。ノードは、”type::name,”でラベル付けされた楕円で示され、関係はロール名でラベル付けされた矢印で示されている。

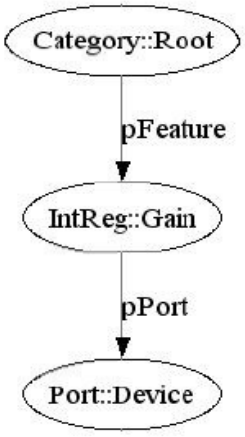

図2　簡単な記述ファイルのグラフトポロジー

2つの特別なノードが存在する。ノードグラフをたどる始点となるRootノードと、トランスポート層への接続を提供するDeviceノードである。図2のGainノードは、IntReg型に属し、レジスターからの整数（integer）の抽出を行う。Rootノードから見ると、それはカメラの機能として見える。従って、Rootノードは、Gainノードを参照するpFeatureという名前のリンクを包含する。Gainレジスターを読んだり書いたりするために、Gainノードはカメラポートにアクセスする必要があり、そのためにDeviceノードへのリンクを保持している。そのリンクはpPortと呼ばれ、Deviceノードを参照する。

図2の完全なカメラ記述ファイルは、図3のようにXMLで記述される。

<?xml> ノードは、XML宣言で、ファイルの符号化情報を与える処理要素で常に同じである。

<RegisterDescription>要素は、最も外側のブラケットで、カメラの全てのノード記述をカプセル化している。カメラはモード名（ModeName）とベンダー名（VendorName）属性で識別される（この場合は、”Test”ベンダーからのモデル”Example01”）。

<RegisterDescription>要素の内部では下位の要素としてのノード群がフラットに順番に置かれている。個々のノードはユニークな名前属性を持ち、他のノードの名前を包含するpRoleとしての名前の下位要素にリンクで関係付けられる。

個々のノードは、<ToolTip>要素（必須ではない）を持ち、そこで簡単に説明される。GainノードはRootに追加された要素で、IntReg型であり、レジスターの番地やデータ長を通知する機能を有する。

XMLファイルの構文は、XML schemaで定義され、それはschemaLocation属性により与えられる。このschemaは規格の一部となっている。

2.3 値の取得と設定

ユーザがノードの値を読み書きする時、当該ノードは、ノードグラフ上をカスケード的

```
<?xml version=”1.0” encoding=”utf−8”?>
<RegisterDescription
    ModelName=”Example01”
    VendorName=”Test”
    ToolTip=”Example 01 from the GenApi standard”
    StandardNameSpace=”None”
    SchemaMajorVersion=”1”
    SchemaMinorVersion=”1”
    SchemaSubMinorVersion=”0”
    MajorVersion=”1”
    MinorVersion=”0”
    SubMinorVersion=”0”
    ProductGuid=”1F3C6A72−7842−4edd−9130−E2E90A2058BA”
    VersionGuid=”7645D2A1−A41E−4ac6−B486−1531FB7BECE6”
    xmlns=”http://www.genicam.org/GenApi/Version_1_1”
    xmlns:xsi=”http://www.w3.org/2001/XMLSchema−instance”
    xsi:schemaLocation=”http://www.genicam.org/GenApi/Version_1_1
    http://www.genicam.org/GenApi/GenApiSchema_Version_1_1.xsd”>
    <Category Name=”Root”>
        <ToolTip>Entry for traversing the node graph</ToolTip>
        <pFeature>Gain</pFeature>
    </Category>
    <IntReg Name=”Gain”>
        <ToolTip>Access node for the camera’s Gain feature</ToolTip>
        <Address>0x0815</Address>
        <Length>2</Length>
        <AccessMode>RW</AccessMode>
        <pPort>Device</pPort>
        <Sign>Unsigned</Sign>
        <Endianess>BigEndian</Endianess>
    </IntReg>
    <Port Name=”Device”>
        <ToolTip> Port node giving access to the camera</ToolTip>
    </Port>
</RegisterDescription>
```

図3　カメラ記述ファイルの例

に逐次トリガーする。図4はGain機能について示している。

ユーザがGainノードの値を読むと、呼び出しはGainValueノードにディスパッチされる。それはさらにDeviceノードからのIPortインタフェースを用い正しいレジスターを照会する。

もしユーザがGainノードの値を設定する場合、実装は対応するGainMinとGainMaxノードを通じてMinとMaxの値を読み、範囲（range）をチェックする。もし値が許される範囲内であれば、GainノードはGainValueノードとカメラへのDeviceノードを通じてその値を書込む。対応するIntRegノードの

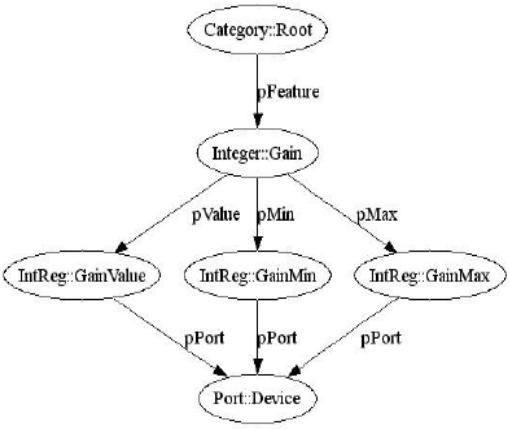

図4　値の読み書きの際の制御フロー

Cashable属性によってMinとMaxの値をキャッシュする場合もある。

3. トランスポート機能仕様

3.1 機能の概要

GenICam GenTL規格は、GenICamアプリケーションの相互運用性を確保するための汎用的なTransport Layer Interface (TLI)を定義することにある。このトランスポート技術とサードパーティソフトの間のソフトウエアインタフェースは、Cインタフェースに準拠し、一連の機能名称、意味的内容で定義される。これらの機能にアクセスするために、GenICam　GenApiモジュールが用いられる。

GenICam GenAPIモジュールは、前章で述べたとおりXML記述ファイルフォーマットを定める。標準的な機能名称の記述はXMLファイルを通じてこれらの機能の振舞いをニュートラルに定義する。

GenTLソフトウエアインタフェースは、通信確立の場合を除き遠隔デバイスのデバイス依存の機能を包含しない。GenTLはGenApiモジュールを経由して遠隔デバイス機能へのアクセスを許すポートを提供する。

これは、GenTLがデバイスとそのストリームデータを通信させる汎用のソフトウエアインタフェースを構築することを意味する。GenApiとGenTLとの組み合わせにより、カメラのようなデバイスにネットワーク経由でアクセスするための完全なソフトウエアアーキテクチャを提供する。

3.2 GenTLの基本構成

GenTLは、基本的には2種類の実体、プロデューサーとコンシューマーから構成される。GenTLプロデューサは、ソフトウエアドライバーでGenTLインタフェースを実装しアプリケーションやソフトウエアライブラリーにハードウエアへの汎用的なアクセスや構成を実現し、デバイスからの画像データを配信する。

GenTLコンシューマは、一つ以上のGenTLプロデューサを使用するソフトエアで定義されたGenTLインタフェースを経由する実体である。これらの関係は図5で示される。これはアプリケーションまたはソフトライブラリの一例である。

3.3 GenTLモジュール

GenTL標準は、GenTLインタフェースをライブラリ実装するために階層的な構造を定義する。個々の階層はモジュールとして定義される。モジュールは図6に示すようにシステムモジュールをベースとしてツリー構造として構成される。

3.3.1　システムモジュール

GenTLコンシューマにとって、ハイアラーキのルートであるシステムモジュールはGenTLプロデューサのソフトウエア・ドライバへのエントリーポイントである。それはGenTLライブラリの視点からのホスト側における全システムを代表する。

システムモジュールの主要なタスクは、実装がカバーする提供可能なインタフェースを枚挙しインスタンスとして生成することである。

システムモジュールは、さらにGenTLコンシューマへ通知機能とXML記述による構成を提供する。

複数のトランスポート層技術に関係する単一のGenTLプロデューサを異なるインタ

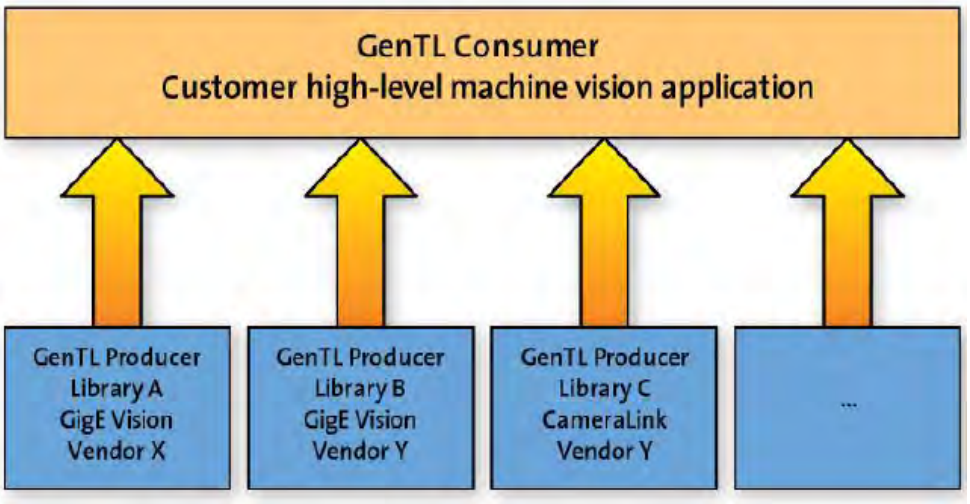

図5　GenTLコンシューマーとGenTLプロデューサ

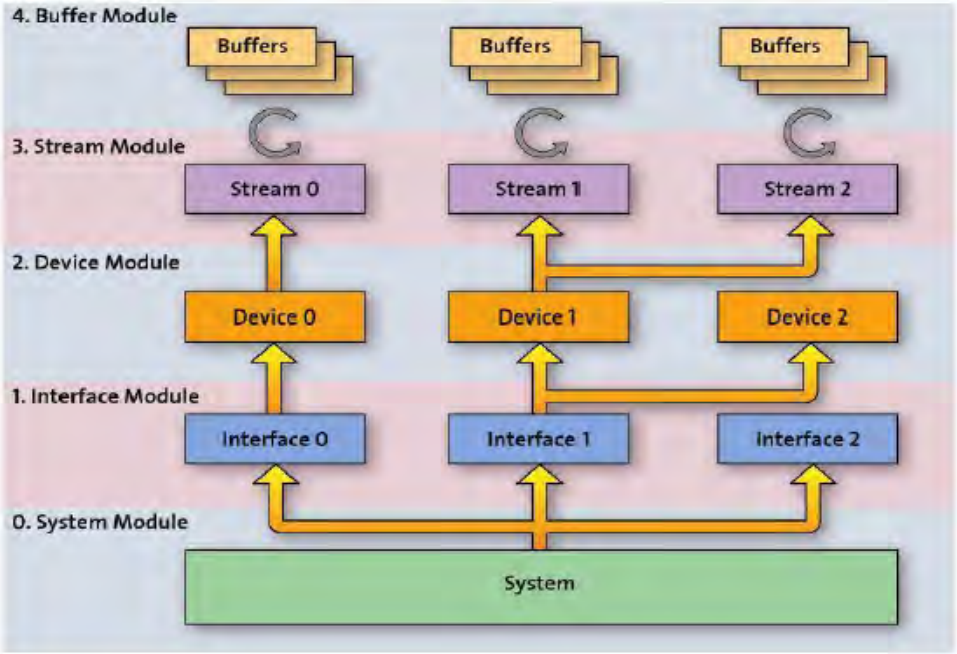

図6　GenTLモジュール階層

フェースモジュールとして公開することは可能である。この場合、システムモジュールのトランスポート層技術は、複合的であり子供のインタフェースモジュールは、それらの実際のトランスポート層の技術を実現する。

3.3.2　インタフェース・モジュール

インタフェースモジュールはシステムにおける物理インタフェースを代表する。Ethernetベースのトランスポート層技術にとっては、これはNICカードとなる。カメラリンクベースの実装では、これはフレーム獲得基盤である。このインタフェースにおける適合デバイスの枚挙とインスタンス生成がこのモジュールの主たる役割である。インタフェースモジュールは、さらにGenTLコンシューマのコンシューマへの非同期通知とモジュール構成機能を代表する。

3.3.3　デバイスモジュール

デバイスモジュールは、GenTLプロデューサにおける物理的な遠隔デバイスのプロキシを代表する。デバイスモジュールの責任は、遠隔デバイスの通信を可能にし、データストリームモジュールを枚挙しインスタンス生成することである。デバイスモジュールは、さらにGenTLコンシューマへのコンシューマへの非同期通知とモジュール構成機能を提示する。

3.3.4　データストリームモジュール

遠隔デバイスからの単一の（画像）データストリームは、データストリームモジュールで代表される。このモジュールの目的は、取得エンジンを提供し、内部バッファプールを維持することにある。それと共に、データストリームモジュールは、GenTLコンシューマにコンシューマへの非同期通知とモジュール構成機能を提供する。一つのデバイスは、ゼロ、１、または複数のデータストリームを包含する。デバイスが保有するストリーム数の論理的な制限は存在しない。これは完全にハードウエアによってのみ制約される。

3.3.5　バッファ・モジュール

バッファ・モジュールは、単一のメモリー・バッファをカプセル化する。その目的は、画像取得の標的として機能することにある。バッファのメモリーは、ユーザ又はGenTLプロデューサが割り付ける。後者はシステムメモリーとして事前に割り付けられていることもある。さらにバッファ・モジュールは、GenTLコンシューマに、非同期通知やモジュール構成機能を提供する。

3.4 GenTLモジュール相互運用部品

GenTLコンシューマとGenTLプロデューサのアクセスと互換性は、図7に示すように、CインタフェースとGenApi機能インタフェース（モジュール、コンシューマへの非同期通知、XML記述による構成、データ取得エンジンの振る舞いの記述など）の相互運用性を通じて保証される。

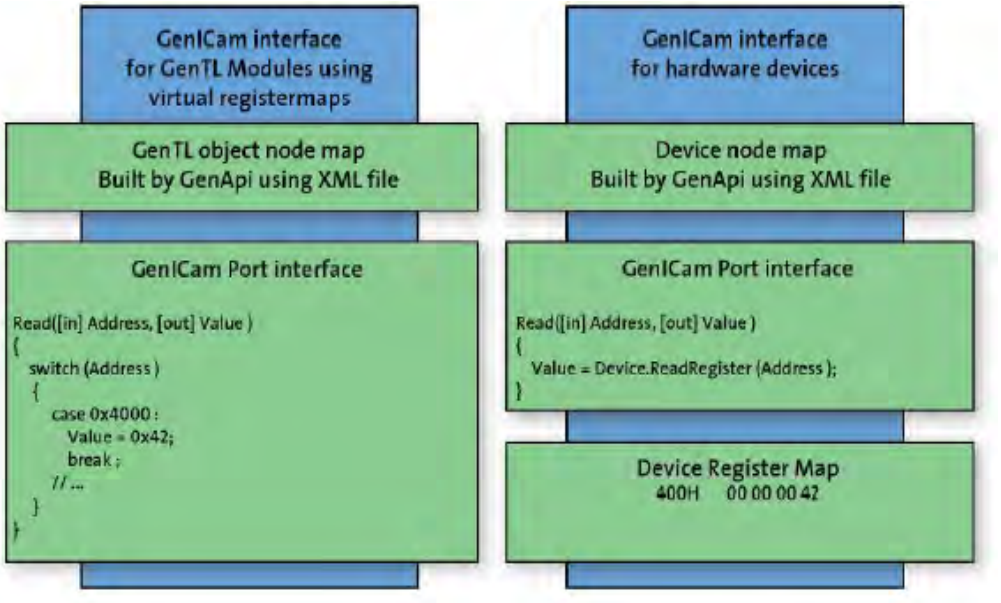

図7　GenICam GenTLインタフェース（CおよびGenApi機能インタフェース）

GenTLプロデューサドライバは、Cインタフェース、XML記述構成インタフェース、イベントインタフェース（コンシューマへの非同期通知）の3種類の論理的部品で構成さ

れる。インタフェースの詳細は下記のとおりである。

3.4.1 Cインタフェース

Cインタフェースは、GenTLプロデューサの実体への位置を提供する。それは、全てのモジュールインスタンスを枚挙して行く。それにはデータストリームモジュールにより操作される画像取得が含まれる。モジュールのコンシューマへの非同期通知・構成インタフェースもCインタフェースを経由してGenTLコンシューマにアクセスされる。このようにして、Cインタフェースを用いるだけで下位の技術を使用しないで画像を転送することが可能になる。さらにこのことは、GenTLプロデューサがデバイスを開放しそれからのデータの受信を保証することを意味する。Cインタフェースは、下記のような理由で選択されている。

・複数のクライアント言語サポート：Cインタフェース・ライブラリは、沢山のプログラム言語でサポートされる。基本型は容易に言語間やモジュール間（異なるheap、実装の詳細）でマーシャリング可能である。

・ライブラリの動的なローディング：C言語の関数を動的に呼び出すのは容易に可能である。このことは、実行時に一つ以上のGenTLコンシューマの実装を動的にロードすることを可能にする。

・アップグレード可能性：Cライブラリは、当初の段階ではバイナリ互換で設計される。そのため、GenTLコンシューマは改版されても再コンパイルを必要とはしない。

上記の理由でCインタフェースが選択されても、実際のGenTLプロデューサ実装はオブジェクト指向言語で実行することが可能である。グローバル関数以外は、インタフェース関数は、オブジェクトにマップされて操作される関数として機能する。

Cインタフェースのライブラリをエクスポート可能な任意のプログラム言語でGenTLプロデュサは実装可能である。

GenTLプロデューサとGenTLコンシューマの変換性（intercangeability）を保証するために、言語依存の機能は決められていないが、例外的に、ANSI CがGenTLプロデューサのインタフェースで用いられている。

3.4.2 モジュール階層へのアクセス

個々のモジュールはGenTLポート機能を提供するので、GenICam のGenApi（または同様の非参照実装）がモジュール階層のアクセスに用いられる。GenTLプロデューサ実装の基本操作は、特別なモジュール構成を用いないCインタフェースで行われる。より複合的または実装依存のアクセスは、フレキシブルなGenApi機能インタフェースで行われ、GenTLポート機能と提供されるGenApiのXML記述を用いる。

個々のモジュールは、このXML記述を提供し、それと共にモジュールのポートがモジュールの設定を読んだり変更したりする。そのために、個々のモジュールはその仮想レジスタマップを保有し、それはポート機能によりアクセスされる。このようにして、汎用的な遠隔デバイスへのアクセス方法がトランスポート層自身へ拡張される。

2.3.3 コンシューマへの非同期通知（イベント）

個々のモジュールはGenTLコンスシューマへのイベント通知の提供を可能にする。例えば、新しい画像情報データが遠隔デバイスから到着すると、New Bufferイベントが生じてコンシューマへの非同期通知される。特定のモジュールを支援するためのイベント数は、モジュールとその実装に依存する。

Cインタフェースは、GenTLコンシューマがレジスタにイベント登録することを可能にする。用いられるイベントオブジェクトは、プラットフォーム及び実装に依存するが、Cインタフェースにおいてはカプセル化されている。

4. まとめ及び考察

以上、GenICam Standard Generic Interface for Cameras Version 2.0と、GenICam GenTL Standard Version 1.2の２つの仕様書の概要を紹介した。ネットワークカメラの設定にXMLファイルを用いることが大きな特徴である。

カメラの機能は多種多様であり、それはW3CのXML Schemaで記述される。設定ファイルは、このXML Schemaのインスタンスである。このXMLインスタンスは、XMLの本質として木構造を持つが、この構造がカメラの状態遷移を表現するノード関係を定義し、設定するカメラのレジスタへのアクセスを記述している。

XMLファイルで定義された状態ノードは、型と名前を持つ。この型と名前は、C言語のAPIに対応し、APIに値を設定するような手順で、GenApiというモジュールがトランスポート層にデータを渡し、それがネットワーク経由で遠隔のカメラのレジスタのデータを設定する。設定するだけでなく、当然レジスタの値を読むこともできる。

以上のメカニズムから、多様な応用が考えられるであろう。いわゆるWebカメラのインタフェースが一元化されるので、多様なカメラに対して同じアプリケーションを使用することが可能になる。さらに、複数のカメラを連携させて画像処理を行わせることにより、移動体の追跡なども容易になるであろう。

トランスポート層については、構成だけで具体的な実装の議論が十分できなかったが、これは仕様書だけを見ても良く分からない。要するにトランスポート層以下を自由に構成するためのモジュールとインタフェースの規定だけが述べられているからである。だがWebサービスによるXMLデータばかり扱っていた者としては、C言語に割り切ったコンパクトなインタフェース設計は優れた手法であると思う。

Webの世界のプロトコルの主流はHTTPやHTTPSであり、最近はHTML5の流行で、Webソケットを用いる全二重化されたHTTPが話題になっているようだが、GenICamは、むしろ古典的なRPCやCORBAでも実現可能なアーキテクチャを実現している。その鍵は、C言語をコアとした実装だからである。

Webでは、HTML5でSVGを用いて、図形や画像を表示するが、その実態はデータを上回るタグの伝送と処理に通信路とコンピュータが使われており、工学系のセンスの乏しい技術の進展を苦々しく思っていたところであるが、GenICamはその点エレガントな技術的なアプローチを採っているように思う。

5. おわりに

カメラのハードウエアについては、レジスタの集合という以外は全く触れないで、ネットワークとXMLの世界からGenICamを取り上げたのであるが、WebやXMLの世界にどっぷり浸かりこんでいる技術者からすると目から鱗のような技術である。組込系のような、ハードウエア、ソフトウエア、ネットワークといった複合的な技術を推進するために必要なセンスを培うには、GenICamのような技術はきわめて興味深いと感じる。このような分野に関心を持つ技術者が今後育ってくれることを期待したい。この技術を知るきっかけを与えていただいた、（株）テクノスコープの白川 進氏に感謝します。

文献

[1] EMVA; ”GenICam GenTL Standard Version 1.2”, http://www.emva.org/cms/upload/Standards/GenICam_Downloads/GenICam_GenTL_1_2.pdf

[2] EVMA; “GenICam Standard Generic Interface for Cameras Version 2.0”, http://ebookbrowse.com/genicam−standard−v2−0−pdf−d68384593