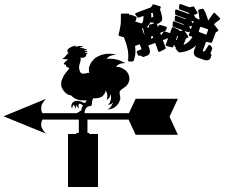


組込みシステム構成の類型

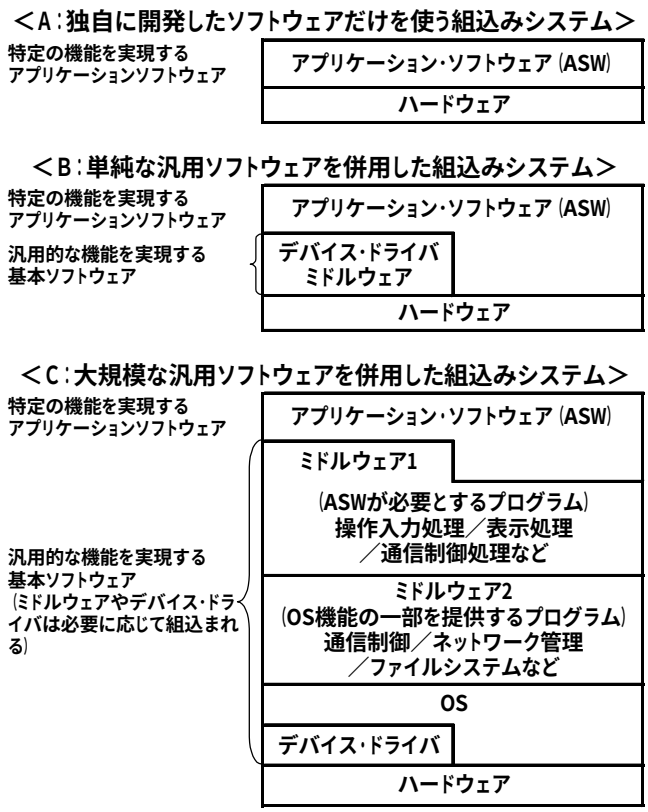
これまで、一般化した組込みシステムのハードウェア／ソフトウェア構成例を見てきた。実際にはシステムごとに含まれるレイヤーや要素、そして要素間の関係が変わってくるが、代表的な構成は下図のように３つに類型化できる。

Ａ：独自に開発した専用のアプリケーションソフトウェアだけを実装したもの

Ｂ：小規模な基本ソフトウェアと独自に開発した専用のアプリケーションソフトウェアを実装したもの

Ｃ：大規模な汎用ソフトウェアと独自に開発した専用のソフトウェアを実装したもの

組込みシステムで実現しようとする機能が高度化すると、一般にＡ→Ｂ→Ｃの順番でプログラム規模は大規模化し、複雑化する。ここでいう基本ソフトウェアとは、具体的に、デバイスドライバ、モニタ表示や通信制御、ネットワーク管理、ファイルシステムなどといった「汎用的な機能を実現するプログラム」であり、独自に開発した専用のソフトウェアとは、モーター制御やブレーキ制御、センサー監視、加熱制御などといった「特定の機能を実現するプログラム」のことをいう。ＯＳ などの汎用的なプログラムは開発効率をあげるために既存プログラムが再利用される傾向が高く、図表２のＢやＣで示したデバイスドライバ、ＯＳ、ミドルウェアなどの汎用プログラムは既存のプログラムが再利用されることが多い。



図表：組込みシステムの構成例　（作成：安田賢憲）

従来、リアルタイム性が低いものの高機能が志向される製品に搭載される組込みシステムは、図表の Ｃ のような構成となる場合が多い。一方で、高いリアルタイム性を要求する自動車のような製品は図表２の**Ａ**や**Ｂ**のようにシンプルな構成になる場合が多い。ただし、近年、厳しい競争の中、高機能化が急速に起こっていおり、大規模な組込みシステムが搭載される傾向にあるという。

（東京富士大学　准教授　安田賢憲）

以下、図表5-3で掲げた組込みシステムのハードウェアおよびソフトウェアの各構成要素について概要を述べる（図表5-5、図表5-6）。

図表5-5　組込みシステム・ハードウェアの構成要素

ハードウェア	
マイクロプロセッサ	メモリに読み込まれたプログラムに従ってコントローラの制御や整数・浮動小数点の演算、メモリの管理などを行うLSI。プログラムに応じて多目的の処理を行える。パソコンで使われているものと仕組みは同じだが、組み込み用に特化した仕様のものもある。組込みシステムでは、制御用に特化したマイクロプロセッサをマイコン（マイクロコントローラ）と呼ぶ。マイクロプロセッサは周囲のハードウェア・コンポーネントとバスと呼ばれる回路を通じて接続される。
DSP等プロセッサ	DSPはDigital Signal Processor。電気信号や音声などの信号処理を高速に行うためのLSI。DSP以外にも、暗号処理や画像処理など特定領域の処理に適したプロセッサが必要に応じて用いられる。
メモリ	記憶装置。大きくRAM（Random Access Memory）とROM（Read-Only Memory）に分かれる。前者にはDRAM（Dynamic RAM）やSRAM（Static RAM）が主に用いられるが、これらは電源を切ると記憶が消えてしまう。パソコンの場合は、ソフトウェアはハードディスクに記録されているものをRAMに読み込んだ上でマイクロプロセッサが実行するものであるが、組込みソフトウェアの多くは製品出荷時にROMに書き込まれ、マイクロプロセッサはそれをROMから直接読み出しながら実行する。ROMは予め書き込んでおいた内容を何度も読み出すために使われるメモリで、主に回路基板上で記憶の消去・書き換えが可能なEEPROM（Electronically Erasable and Programmable ROM）やフラッシュメモリが用いられる。EEPROMやフラッシュメモリは電源を切っても記憶が消えない書き換え可能なメモリという意味で不揮発性メモリとも呼ばれる。
HMI	Human-Machine Interface。LED、ディスプレイ、ボタン、キーボード、音響装置など、人間と機器との間の情報のやり取りの接点となる装置。
外部I/O	外部Input/Output。ハードディスクドライブやUSB機器など、組込みシステムのボード本体と接続される外部入出力装置。
通信モジュール	回線、無線、ネットワークなどを通じた通信を制御するコンポーネントの総称。
その他コンポーネント	組込みシステムでよく見られるその他のコンポーネントとして次のいくつかを挙げる。センサー、アクチュエータを用いて電子的に制御される機構（ロボットアームやDVDのピックアップ、電動のラッチなど・・・機構の電子制御技術を日本ではメカトロニクスという）、赤外線送／受信モジュール（リモコン用）など。

図表 5-6　組込みソフトウェアの構成要素

ソフトウェア	
デバイスドライバ	DSP、HMI、外部 I/O などのコンポーネント／装置の基本的な操作するためのプログラムで、OS、ミドルウェアやアプリケーションから利用される。OS を利用する場合は、デバイスドライバは OS の管理下に置かれ、アプリケーションは必ずしもデバイスドライバを直接呼び出す必要はなく、OS の API を通じて間接的に利用される。
OS	Operating System。オペレーティング･システム。次の複数の役割を担うソフトウェア。 1. マイクロプロセッサやその他のハードウェア・コンポーネントあるいは装置の状態や利用を管理したり、アプリケーションに対するこうしたリソースの配分を管理したりする。 2. アプリケーションが API（Application Programming Interface）を通じてコンピュータ･システムを「ファイル」「コンソール」「ストリーム」「プロセス」「スレッド」などといった（ハードウェア上に必ずしも存在しない）抽象概念によって利用・操作できるようにする。 3. こうしたことにより、ハードウェア構成の細かな違いに大きく影響されない（共通仕様で）移植性の高いアプリケーションを開発可能とする。 規模の小さい組込みシステムやミッションクリティカルな組込みシステムでは OS を用いず、アプリケーションが直接ハードウェアを操作する場合も多い。これは OS が間に入ることでメモリ使用量が増えたり性能が落ちることと、OS がブラックボックスであり開発者が思ったようなハードウェア操作ができないことがあるためである。規模が大きく複雑な組込みシステムの場合などは、上記機能によるソフトウェア開発生産性向上やリソース管理の最適化、アプリケーションの移植性の高さといった利点が欠点を上回り、OS が採用されやすくなる。
ミドルウェア	MW とも書く。ミドルウェアは、システムの機能実現のために必要となる（例えば特定規格に基づく動画データや音声データなどの）処理アルゴリズムや、OS に含まれていない管理機能などを実装したソフトウェア。
アプリケーション	ユーザによるシステムの利用目的に応じた機能を実現するソフトウェア。組込みシステムでは、ほとんどの場合アプリケーションは C または C++言語によって記述される。また、基本的に ROM に予め書き込まれた状態で製品ごと出荷したものがそのまま使われる。しかし、出荷後のバグ修正や新しい通信規格などに対応するために、ネットワークを通じて更新できるようになっている場合もある。
バーチャルマシン	VM とも書く。OS は既にハードウェアをある程度抽象化しているが、バーチャルマシンはさらにマイクロプロセッサや OS の違い、メモリ管理までも抽象化してアプリケーションから見えなくしてしまう。これによってアプリケーションはコンパクトになり、その開

	発生産性がさらに上がるだけでなく、ハードウェア構成がある程度異なるシステム間で同じアプリケーションをそのまま動作させることが可能になる。ただし VM を使用するにはその分多くのメモリとマイクロプロセッサの処理能力を必要とするため、これを搭載した組込みシステムは一部に留まる。現在では多くの携帯電話端末に Java VM（アプリケーションは Java 言語で記述）が搭載されているが、その主な用途は製品の購入後にユーザが任意でダウンロードして利用できる、いわゆるケータイアプリの実行である[17]。

5.3.2 マイクロプロセッサと IP コア

マイクロプロセッサとはメモリとの間でのプログラムおよびデータの読み書き、I/O（Input/Output）を通じた外部機器との間のデータの入出力、演算などを行う単体の LSI のことである。一般には世界初のマイクロプロセッサは 1971 年に米 Intel から登場した 4-bit CPU「4004」だと言われている。それ以前には上記機能は複数の IC や半導体部品を組み合わせて作られるのみであり小型化・省電力化には不利であったが、マイクロプロセッサの登場により、小型の電気製品にこれを組込み、メモリに搭載するプログラム次第で製品のさまざまな機能を実現できる（今で言う）組込みシステムの概念が誕生したのである[18]。

現在のマイクロプロセッサの主流は 32-bit であり[19]、組込みシステムでは後述の RISC と呼ばれるアーキテクチャが多く用いられる。パソコン用で主流の米 Intel や　米 AMD の「x86」と総称されるマイクロプロセッサとは同じ 32-bit であっても CISC アーキテクチャである点が異なる。現在主流であるパソコン用マイクロプロセッサは処理能力は高いが消費電力も多く、発熱も多いため、組込みシステムで用いられることは少ない。ただし、8-bit および 16-bit の CISC アーキテクチャのマイクロプロセッサは現在でも組込みシステムでしばしば用いられている。

以下で代表的な組込みシステム用マイクロプロセッサ製品を分類する。

① RISC アーキテクチャ

RISC（Reduced Instruction Set Computer）アーキテクチャは CISC プロセッサと比べ少ない命令セットによって小型化、高性能化を図ったプロセッサ・アーキテクチャである。反面、少なくなった命令セットを多数の命令の実行で補うため、必要とするメモリ容量が CISC よりも大きい。そのため、かつては搭載可能なメモリに制限のある組込みシステムでの適用には向かないとされていたが、近年では組込みシステムであっても大きなメモリ容

[17] 一般にケータイアプリのような、製品の購入後に任意にダウンロードしてその製品上で利用するソフトウェアは組込みソフトウェアではない。なぜなら、そのようなソフトウェアはあってもなくても製品（携帯電話端末）自体が提供する機能（たとえば通話機能やメール機能）には影響しないからである。しかし最近は、たとえば「おサイフケータイ」（電子マネーを利用した小額決済機能）のような製品機能を有効化するために、ユーザによるこの機能の利用形態に合わせた携帯アプリを後から選び、ダウンロードして使用しなければならない例もある。このことから、今後この種のアプリケーションのどれが組込みソフトウェアでどれが非組込みソフトウェアなのかの境界線が曖昧になっていくことが予見される。

[18] よく知られているように、この「4004」は日本の電卓メーカーであるビジコン社の依頼を受けて当時の弱小半導体メーカーのインテルが開発したものだった。結局インテルはこの 4-bit マイクロプロセッサに関する権利を買い取り、後の 8-bit CPU「8008」や 16-bit CPU「8086」などへと発展させていった。

[19] 一回の命令実行につき 32 ビット分のデータ（日本語文字 2 文字分相当）を処理できることを表す。

量を持つことができるようになり、組込みシステムにおいてもっともポピュラーなプロセッサ・アーキテクチャとなっている。代表的な製品としては以下が挙げられる。

- R シリーズ（MIPS、SGI）
- StrongARM（Intel）
  - ARM コアのライセンスを受けた DEC が製造したプロセッサ。その後 Intel の所有となり、XScale に引き継がれている。
- XScale（Intel）
- H8、SuperH シリーズ（ルネサステクノロジー）
  - 日立製作所より移管
- PowerPC（Freescale）
- V850 シリーズ（NEC）
- FR シリーズ（富士通）
- TX RISC（東芝）
  - MIPS アーキテクチャをベースにしている。

② CISC アーキテクチャ

CISC（Complex Instruction Set Computer）は大きな命令セットを持つプロセッサ・アーキテクチャの総称である。基本的な処理命令のほかに、多数の高機能な命令を持つため、少ない命令の組み合わせで多くの処理を実行でき、かつ少ないメモリでも搭載・実行できるプログラムを実現しやすいとされる。RISC アーキテクチャの普及で主流からは退いた感があるが、現在でも適材適所で用いられている。上記の特徴に加え、過去のプログラム資産を継承しやすいことも、CISC アーキテクチャを息の長いものにしている。代表的な製品としては以下が挙げられる。

- M16C、M32C（ルネサステクノロジー）
  - 三菱電機より移管
- Z80（Zilog）
- x86 系（Intel、AMD）
- 68000 シリーズ（Motorola）
  - Freescale 社により組込みシステム用に ColdFire シリーズとして継承され、開発が続いている。

組込みシステム用マイクロプロセッサは、完成品として製造・販売されるだけでなく、その設計資産をメーカーにライセンスすることでも利用される。この設計資産を IP コアと呼ぶ。IP とは Intellectual Property（知的財産）のことである。

IP コアの実態はハードウェア設計のための一種のライブラリである。これを購入することにより一から論理設計するのではなく、例えばマイクロプロセッサといった大きなモジュールを構成する

論理回路のブロックを再利用しながら設計できる。

ソフト IP プロセッサは IP コアの組み合わせで組込みシステム用マイクロプロセッサを設計・実装するための製品。ほかに DSP 用、USB 等の I/O 機能などのための IP コアが製品化されている。

上記の MIPS も IP コア販売のビジネスモデルである。また、英 ARM Limited 社自体は ARM コアの IP を販売することで売り上げを上げる会社である。ARM のアーキテクチャに基づくプロセッサは少数のトランジスタの利用で消費電力も非常に低いにもかかわらず高性能であり、携帯電話、ポータブルゲーム機（ゲームボーイアドバンス、ニンテンドーDS Lite など）や、その他組み込みシステムでの利用が多い。他に代表的なプロセッサ IP コアとして米 Altera 社の Nios II などが上げられる。

5.3.3 ロジック IC

ロジック IC（Integrated Circuit）とは特定の機能を果たす論理回路をひとつの集積回路としてパッケージしたものである[20]。マイクロプロセッサは大型のロジック IC であるといえるが、組込みシステム内ではそれ以外にも各種コンポーネントを制御するためのコントローラや特定の信号処理や演算を行うためのロジックなどもしばしばロジック IC として実装される。また、携帯電話などで用いる通信モジュールでは、ロジックを実装するデジタル回路だけでなく、信号を処理するアナログ回路をひとつのロジック IC 内に実装し、利用する場合がある[21]。

ロジック IC には用途や費用対効果に合わせてさまざまな種類が用意されており、それぞれその設計・製造方法が異なる。図表 5-7～5-9 ではロジック IC を分類する。

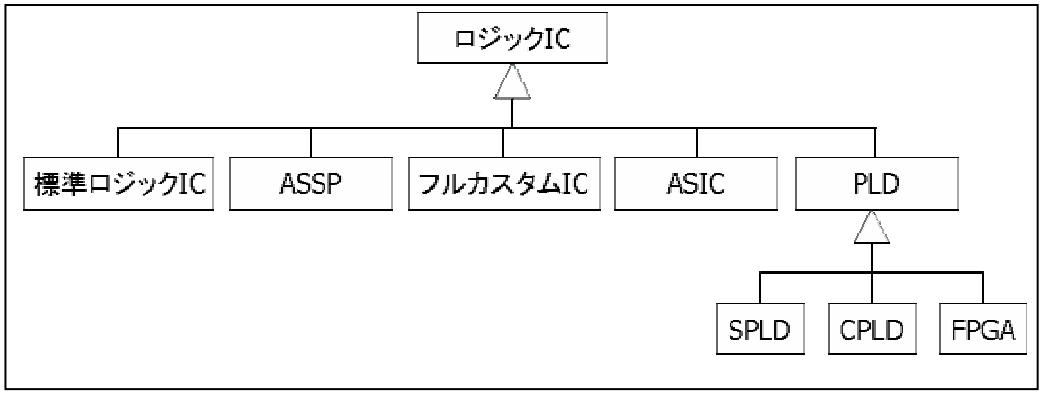
図表 5-7 ロジック IC の種類

標準ロジック IC	**特定の機能を持ったロジック回路を一つのパッケージとしてまとめた IC。かつては標準ロジック IC を組み合わせてシステムを設計することもあったが、ASIC の登場などにより、その役目は限定的なものとなっている。**
フルカスタム IC	**ロジック設計、及びロジック回路からカスタムで設計・製造する IC。設計に多大な時間とマン・パワーを要するが、最小のゲート数で最大限の性能を引き出せる。少量生産には向かず、マイクロプロセッサなど出荷量が大きい LSI 以外では用いられない方式。**
ASIC	**Application-Specific Integrated Circuit。セミカスタム IC。セルと呼ばれるロジック回路設計の一種の作り置き部品を組み合わせることで IC を設計する。VHDL の普及にともない、1985 年ごろから一般的となった。EDA ツールを使って、VHDL や RTL による仕様・振る舞い記述から ASIC の設計を（半）自動で生成できる。フルカスタム IC と比べ設計時間とコストを大幅に削減できるが、やはり少量生産には向かない。** **その製造方法にはいくつかあるが、ゲートアレイ(Gate Array)はその代表的なものである。こ**

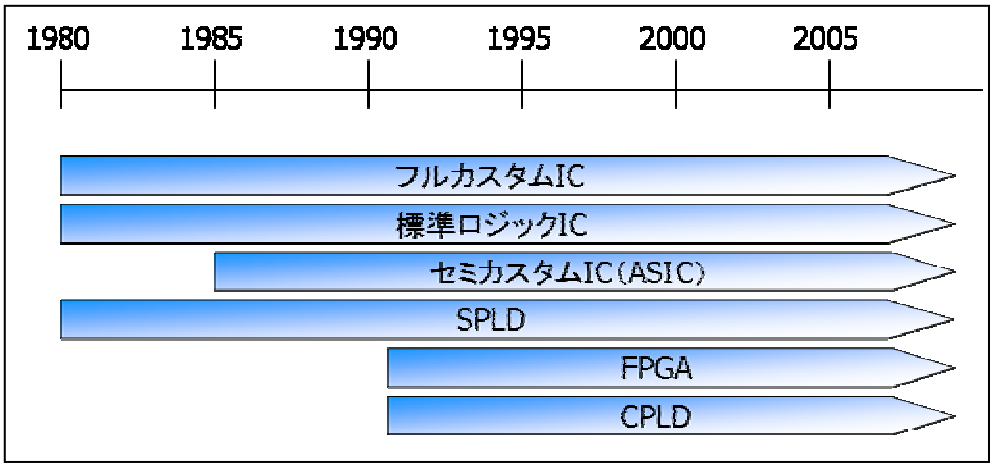
[20] したがってメモリはロジック IC ではない。
[21] この分野では米 Texas Instruments 社が有力であるとされる。

	れは、基本的なロジック回路（ゲート）を予め配列（アレイ）状に配置した LSI 設計があり、ユーザの発注に応じてゲート間の配線層だけを新たに設計することによって、フルカスタム IC よりも集積度や性能に劣るものの、低コストで製造できるようにしたものである。
ASSP	Application-Specific Standard Product。特定用途のために作られた LSI。例えば、DSP（Digital Signal Processor）は信号処理用の汎用プロセッサだが、この用途を特定し、例えば画像処理用の DSP として設計されたものは ASSP である。
PLD	Programmable Logic Device。ユーザによって（購入後に）プログラミング可能な（ロジック回路の）IC の総称。狭義には SPLD、CPLD を含み、広義にはさらに FPGA を含む。PLD による LSI の設計時間は、ASIC と比べ 3～4 倍短いといわれる。
FPGA	Field Programmable Gate Array。その名の通り「現場でプログラミングできるゲートアレイ」。もともと Xilinx 社の製品。プログラミングによってユーザが設計したロジック回路を実現できる LSI。CPLD とともに 1990 年代に普及した。ASIC と比べ、動作速度は遅く大量生産時の一個当たりの製造コストが高いものの、ユーザ（組込みシステムの開発者）自身による LSI 設計が簡単で設計コストも大幅に低い。また、ある機能をあとからハードウェアとして実装する場合などにも重宝する。その特性から、ハードウェアとソフトウェアの中間にある LSI とも言われる。
CPLD	Programmable Logic Device。ユーザによって（購入後に）プログラミング可能な（ロジック回路の）IC の総称。狭義には SPLD、CPLD を含み、広義にはさらに FPGA を含む。PLD による LSI の設計時間は、ASIC と比べ 3～4 倍短いといわれる。
SPLD	Simple Programmable Logic Device。基本的なロジック回路を小規模に集積したプログラミング可能な LSI。1970 年代後半に登場した。GAL（Gate Array Logic）や PAL（Programmable Array Logic）は SPLD の具体的な方式。
SoC	System on a Chip。2000 年ごろに登場した、FPGA などを利用したシステム設計の考え方で、厳密には上記の分類と異なる概念だが、製造プロセスにも関係するのでここで解説する。SoC は、システムに必要なマイクロプロセッサ、メモリ、チップセット、ビデオチップなどを全て一つの LSI として設計・実装したもの。LSI の付加価値を高めるとともに、ハードウェアのトータルコストを抑えることを目的としている。しかし、マイクロプロセッサとメモリなど、異なる設計・製造プロセスを用いる機能を一つのチップ上で混載するため、単機能 LSI と比べ、製造コストが高く、ダイサイズ（チップのサイズ）が大きいために歩留まりも悪いといった欠点もあるとされる。これに対する代替手段として、SiP（System in a Package）がある。これは、個々の機能は別々に設計・製造し、あとからそれを一つのパッケージ内で結合・封止することで SoC の欠点を克服しようとするものである。

図表 5-8　ロジック IC の分類



図表 5-9　ロジック IC の普及時期



５．４　組込みソフトウェア開発がおかれる現状

Software is everywhere！今日、ソフトウェアを含まない機器を見つけるのは難しい。一説には全ての機器の 9 割以上に組込みソフトウェアが搭載されている。例えば自動車は今やコンピュータの塊といっていい組込みシステム製品の代表例である。ドアの開閉、エンジン制御、カーナビなど全ての制御・情報化に渡ってコンピュータが関与している。1 台の自動車が搭載する電子制御ユニット（ECU）の数は、1980 年代は 5 個程度だったものが、現在では 30～100 個以上にまで増えていると言われている。

1971 年の 4 ビットマイコンの誕生を以って産声を上げ、1980 年代以降マイコン搭載をうたった製品が急激に増えることによって普及した組込みソフトウェアだが、特にここ 10 年ほどは製品の中でのその重要度が急激に増している。これほどまでに組込みソフトウェア開発が重要視されるようになったのにはいくつかの理由がある。

ひとつめの理由は、LSI やセンサー等の小型化、高集積化、高性能化に伴い、一台の製品に搭載されるハードウェア・コンポーネントの数が急速に増大し、それを（純機械工学的な方法やアナログ制御では実現できないほど高度に）制御し、またセンサーの情報を利用した付加機能を実現する

ためにソフトウェアの利用度が高まっていることである。つまり製品の付加価値を高める上でソフトウェアは欠かせない要素となっている。一般的には、製品内のおのおののユニット―たとえばエンジンならエンジン―を単体で制御するために組込ソフトウェアが用いられる。しかし、最近ではユニットとユニットの間の協調制御を行うことで、製品全体としての最適で高度な制御を指向するまでになっている。それにともない、組込みソフトウェアも複雑・大規模化している。

上記は製品内のコンポーネント間の協調制御であるが、製品とその周囲の別の製品とのネットワーク化もしばしば要求されるようになった。これがふたつめの理由である。製品間のネットワーク化が進み、インターネットや携帯電話網、その他無線ネットワークを通じた通信を行う製品が増えるに伴い、通信制御、製品の組込みソフト自体の更新やアプリケーションのダウンロード、セキュリティ機構など、ソフトウェアで実現すべき機能が増えているのである。

いまひとつの理由は、製品開発サイクルが短くなり、また、ハードウェア・コンポーネントの変化に対応するため、組み合わせや変更がしやすいソフトウェアであえて実現すべき機能が増えたということである。ただし、ソフトウェアが大変「やわらかいもの」で、自由に変更が利く、というのはハードウェア技術者の誤解である。ソフトウェアがかように大規模・複雑化した現在では、ソフトウェアはやわらかいものというよりも、相当に「ねばねばした」取り扱いにくいものと考えたほうがいい。

ではその組込みソフトウェア開発が具体的にどう難しいのであろうか。これについて次項で組込みソフトウェア開発の特徴とともに述べてみたい。「難しい」という以上、そこには別の何かとの比較の視点が入る。ここでは業務系システム開発との比較を念頭に組込みソフトウェアの特徴を端的に述べているものとしてご理解いただきたい。なお、公平を期すため、次項に続く項では、組込みソフトウェア開発においてむしろ状況を御しやすくするほうに作用する要素についても触れる。

5.4.1 組込みソフトウェア開発の難しさはどこにあるか？

①技術的な制約：ハードウェア設計プロセスとの兼ね合い

- ハードウェアの設計と並行に進むソフトウェア開発。
- ハードウェアは設計変更が難しく、開発の終盤にかけてハードウェア設計の不備をソフトウェアで補わなければならない場合がある。
- 組込みシステムごとに使うマイクロプロセッサが違う、周辺に接続されているコントローラやデバイスが違う、接続の仕方が違う（自由度の高いハードウェア構成）。ソフトウェア開発開始に当たってハードウェア構成（のみならずそれに応じて使用する OS、ミドルウェアの構成）を完全に固定できない場合が多い。

②技術的な制約：プログラミングとソフトウェア設計における制約

- リアルタイム制御（～ミリ秒以内に処理が完了しなければならない、といった形の実装）
  - 単にプログラムのロジックが正しいというだけでは不十分で、OS によるタスク切り替えの遅延（レイテンシ）などの時間要素も考慮に入れてプログラミングする必

要がある。

- 様々なハードウェア・コンポートネントの同時駆動とコーディネーション（統合制御[22]）
  - 単に「こういうタイミングでこれこれこういう処理を完了してくれ」とハードウェア開発チームから指定があるわけではない。ソフトウェア開発チームもプロセッサ周りに接続されたコンポーネントの仕様を理解して、適切なタイミングでそれらに対する出力を行うようにプログラミングしなければならない場合も多い[23]。
- CPU 処理能力、放熱、消費電力、メモリ消費に関して制約が多く、ソフトウェア設計やプログラミングもこうした条件を意識する場合がある。
  - ただし近年はメモリと処理能力の制約は以前より緩い。
- 製品がおかれる環境がもたらす様々な条件の組み合わせの下で適切な制御を行い、製品を正常に動作させなければならない。
  - 例えば、電気ポットであれば、水温、水位、（変動する）電圧など。
- 多様なプログラミングインタフェース、開発環境。
  - 業務系の世界のように箱から取り出してそのまま使えるようなデファクトインタフェース／開発環境というものがない。例えば同じ ITRON（5.5.2）といってもメーカーごと、製品ごとに実装が違ったり細かな差異があったりする。

③ソフトウェア・エンジニアリング上の制約

- 急激に大規模化するソフトウェア。
  - 過去場当たり的に開発したソフトウェアは気がつくと膨大な資産になっており、きれいに作り直すことはできない。過去の資産を継承しつつ、開発の効率化を図らなければならない状況がある。
  - 経済産業省の調査によると、もっとも重要な組込みソフトウェア開発上の課題として挙げられた上位 3 項目は、「品質管理が難しい」「委託先の人材の継続的確保が難しい」「委託前の仕様や計画の確定が難しい」であった[24]。同じ調査において、人材のレベルについては、エントリレベル人材の供給は過剰だが、ハイレベル人材が不足しているとの結果であった。
  - 経済産業省の調査によると、2004 年時点でソースコード行数が 10 万～50 万行未満だった組込みソフトウェアは調査対象の 20%未満であったのに対し、2007 年には約 30%にまで増えている[25]。

---

[22] たとえば自動車では、かつては組込みシステムはエンジン制御、AT（自動変速機）の制御などをそれぞれ単体で制御するだけであったが、1990 年代以降はこれらの制御の仕組みを協調動作させることで自動車全体としてより性能、燃費、乗り心地を高めるようになった。参考：組み込みネット「需要が拡大する自動車制御 OS を知る 」（http://www.kumikomi.net/article/explanation/2005/03osek/01.html）

[23] 参考：CQ 出版　2005 年『組込みソフトウェア開発スタートアップ』、第 3 章

[24] 経済産業省　2007 年『2007 年版組込みソフトウェア産業実態調査報告書』

[25] 経済産業省　2004 年『2004 年版組込みソフトウェア産業実態調査報告書』及び　同　2007 年『2007 年版組込みソフトウェア産業実態調査報告書』

■ 製品分野ごとのソフトウェア開発規模の増大[26]

- 自動車向け組込みソフトウェア開発規模：2000 年→2005 年で 5 倍
- 携帯電話向け組込みソフトウェア開発規模：2001 年→2004 年で 5 倍
- DVD レコーダ向けソフトウェア開発規模：2002 年→2005 年で 9 倍

● 開発チームの作業場所を分散させにくい。

■ 機密保持やハードウェア設計チームとの刷り合わせのために、全員が一つの作業場所に集まって開発に当たるスタイルが一般的。

● 業務系開発と比べてソフトウェア開発方法論／プロセスの整備が遅れている。

■ ハードウェアと比べてソフトウェアは柔軟・臨機応変で（言い換えると）場当たり的な作り込みが可能と思われてきた。

④ビジネス上の制約

● 人材育成に時間がかかる。

■ 外部のトレーニング機関では教えられない知識／スキルが多い。

● ソフトウェアの大規模化と相反して短くなる製品開発サイクル。

■ あるメーカーの携帯電話の組込みソフトウェアの規模は、1996 年からの約 5 年間で 3 倍以上になった一方、開発期間は 1989 年当時は約 12 ヶ月だったものが、1996 年以降は 6 ヶ月まで短縮した[27]。

● 多様化する製品ラインナップ。

■ 似てはいるが少し仕様が違う複数のバージョンのソフトウェアを次々と展開する。

● 製品の高機能化

■ 特に付加価値部分をソフトウェアによる作り込みに頼るようになってきている。

● ソフトウェア不具合の影響が大きい。不具合によって、自動車などでは人命に関わるかもしれないし、コンシューマー製品では不具合によって回収が必要になるとメーカーは莫大なコストを負う可能性がある。

● ソフトウェアに耐タンパ性を持たせる必要がある場合がある。

■ 特に近年では DVD や SD カードの著作権保護機能や IC カードを使った認証など、これらの実現に不可欠な暗号化アルゴリズムや企業秘密などを第三者によってソフトウェアから解析されないよう（耐タンパ性を持つよう）、プログラムの難読化といった技術の適用が必要となる場合が増えている。（図表 5-10）

[26] 本田勝巳（NEC） 2006 年『組込みソフトウェア開発最前線』（http://hiroshi1.hongo.wide.ad.jp/hiroshi/files/tokul/NEC_Honda.pdf）

[27] 平山雅之（東芝研究開発センター） 2002 年『組込みシステム開発における品質向上の施策』ET2002 TB-6