25. 組み込み開発環境に関する知識 I

1. 科目の概要

組み込みシステムの開発手法と組み込みシステムを開発するための開発環境について、その基本構成と特徴を解説する。さらにデバッガやエミュレータを利用した組み込みシステム開発の実際を、ツールの紹介を交えて具体的に示す。

2. 習得ポイント

本科目の学習により習得することが期待されるポイントは以下の通り。

習得ポイント	説　明	シラバスの対応コマ
I-25-1. 組み込みシステムの開発の概要	組み込みシステムを開発するための開発環境についてその基本構成と特徴を概説し、必要なハードウェア、ソフトウェア、ツール群を紹介する。	1
I-25-2. 組み込みLinuxによる実装とその開発	組み込みLinuxを用いた組み込みシステムの実装について説明する。ブートローダからLinux実装までの基本構成や、実際のアプリケーション開発方法、プログラミングからROM化までの手順について、組み込みLinuxシステムを題材に解説する。	1
I-25-3. 組み込みシステムの開発環境	組み込みシステムの開発環境について、全体像とその構成、特徴や開発手法などを説明する。ターゲットOS、言語、ミドルウェア、プログラミング方法、エミュレーション、デバッグといった話題について触れ、統合開発環境が持つ管理機能についても言及する。	2
I-25-4. 組み込みシステムの開発技術	組み込みシステム開発の全体像について、システム全体の設計段階、ハードウェア設計段階、ソフトウェア設計段階からそれぞれの開発を経てシステム提供に至るまでの手順を説明する。またそれぞれの手順で利用される開発環境やツールを紹介し、それらに求められる機能についても解説する。	3
I-25-5. 組み込みソフトウェアのデバッグの基本手順	デバッグの基本であるプログラムのトレースにおいて、組み込みシステムならではの特徴について説明し、トレースの基本手順を解説する。またリアルタイム処理に係るトレースの留意点について説明する。	4
I-25-6. デバッガを利用したプログラムのデバッグ	デバッガを利用したデバッグ環境の全体像と構成、特徴を説明する。デバッグ環境構築上の留意点や組み込みシステムを対象としたデバッグ特有の特徴や留意点などについて説明する。	5
I-25-7. エミュレータの利用方法	エミュレータとは何か、その基本構成と特徴、役割を説明する。またエミュレータを用いたデバッグやプログラム実行の検証、解析方法について解説を加える。	6
I-25-8. ICEを利用したデバッグ環境	組み込みシステムに特有のハードウェアエミュレーションについて説明し、ICE (In Circuit Emulator)の必要性と目的、使い方、ICEを利用したハードウェア関連のデバッグ方法を解説する。また組み込みLinuxに対応した代表的なICEを紹介する。	6
I-25-9. ツールチェーンを利用したデバッグ	ツールチェーンとは何か、その機能と特徴、利点と欠点などについて説明する。またエンディアンの問題やクロスコンパイル環境といった組み込みシステム開発特有の話題について触れ、ツールチェーンを利用したプログラム開発とデバッグの手順を紹介する。	7
I-25-10. 実機レベルのデバッグとトラブルシューティング	組み込みシステムを対象としたデバッグの実際に関して、デバッガや開発環境の制限事項で留意すべき点や、チェックすべき対象、様々な状況下でのトラブルシューティング方法、トラブル解決に利用できるツールなど、様々な話題を紹介する。	8

【学習ガイダンスの使い方】

1. 「習得ポイント」により、当該科目で習得することが期待される概念・知識の全体像を把握する。
2. 「シラバス」、「IT 知識体系との対応関係」、「OSS モデルカリキュラム固有知識」をもとに、必要に応じて、従来の IT 教育プログラム等との相違を把握した上で、具体的な講義計画を考案する。
3. 習得ポイント毎の「学習の要点」と「解説」を参考にして、講義で使用する教材等を準備する。

3. IT 知識体系との対応関係

「25. 組み込み開発環境に関する知識Ⅰ」と IT 知識体系との対応関係は以下の通り。

科目名	基本レベル(Ⅰ)								応用レベル(Ⅱ)						
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
25. 組み込み開発環境に関する知識	<組み込み開発の流れと環境>	<組み込み開発環境の概要>	<組み込み開発手順>	<プログラムデバッグの環境>	<デバガソフトを使用したデバッグ環境>	<ICEを使用したデバッグ環境>	<ツールチェーンによるデバッグ>	<組み込みアプリケーションデバッグの手順>	<組み込みアプリケーションの事例と開発環境>	<GNU開発環境の構築>	<GNU開発環境の特徴>	<GNU開発環境における組み込みアプリケーション開発>	<組み込みLinux開発最新デバッグ環境>	<組み込み開発環境の評価>	<組み込み開発環境におけるデバッグのパターン演習>

[シラバス：http://www.ipa.go.jp/software/open/ossc/download/Model_Curriculum_05_25.pdf]

<IT 知識体系上の関連部分>

分野		科目名	1	2	3	4	5	6	7	8	9	10	11	12	13
組織関連事項と情報システム	1	IT-IAS 情報保証と情報セキュリティ	IT-IAS1. 基礎的な問題	IT-IAS2. 情報セキュリティの仕組み（対策）	IT-IAS3. 運用上の問題	IT-IAS4. ポリシー	IT-IAS5. 攻撃	IT-IAS6. 情報セキュリティ分野	IT-IAS7. フォレンジック（情報証拠）	IT-IAS8. 情報の状態	IT-IAS9. 情報セキュリティサービス	IT-IAS10. 脅威分析モデル	IT-IAS11. 脆弱性		
	2	IT-SP 社会的な観点とプロフェッショナルとしての課題	IT-SP1. プロフェッショナルとしてのコミュニケーション	IT-SP2. コンピュータの歴史	IT-SP3. コンピュータを取り巻く社会環境	IT-SP4. チームワーク	IT-SP5. 知的財産権	IT-SP6. コンピュータの法的問題	IT-SP7. 組織の中のIT	IT-SP8. プロフェッショナルとしての倫理的な問題と責任	IT-SP9. プライバシーと個人の自由				
応用技術	3	IT-IM 情報管理	IT-IM1. 情報管理の概念と基礎	IT-IM2. データベース問合わせ言語	IT-IM3. データアーキテクチャ	IT-IM4. データモデリングとデータベース設計	IT-IM5. データと情報の管理	IT-IM6. データベースの応用分野							
	4	IT-WS Webシステムとその技術	IT-WS1. Web技術	IT-WS2. 情報アーキテクチャ	IT-WS3. デジタルメディア	IT-WS4. Web開発	IT-WS5. 脆弱性	IT-WS6. ソーシャルソフトウェア							
ソフトウェアの方法と技術	5	IT-PF プログラミング基礎	IT-PF1. 基本データ構造	IT-PF2. プログラミングの基本的構成要素	IT-PF3. オブジェクト指向プログラミング	IT-PF4. アルゴリズムと問題解決	IT-PF5. イベント駆動プログラミング	IT-PF6. 再帰							
	6	IT-IPT 技術を統合するためのプログラミング	IT-IPT1. システム間通信	IT-IPT2. データ割り当てと交換	IT-IPT3. 統合的コーディング	IT-IPT4. スクリプティング手法	IT-IPT5. ソフトウェアセキュリティの実現	IT-IPT6. 種々の問題	IT-IPT7. プログラミング言語の概要						
	7	CE-SWE ソフトウェア工学	CE-SWE0. 歴史と概要	CE-SWE1. ソフトウェアプロセス	CE-SWE2. ソフトウェアの要求と仕様	CE-SWE3. ソフトウェアの設計	CE-SWE4. ソフトウェアのテストと検証	CE-SWE5. ソフトウェアの保守	CE-SWE6. ソフトウェア開発・保守ツールと環境	CE-SWE7. ソフトウェアプロジェクト管理	CE-SWE8. 言語翻訳	CE-SWE9. ソフトウェアのフォールトトレランス	CE-SWE10. ソフトウェアの構成管理	CE-SWE11. ソフトェアの標準化	
	8	IT-SIA システムインテグレーションとアーキテクチャ	IT-SIA1. 要求仕様	IT-SIA2. 調達/手配	IT-SIA3. インテグレーション	IT-SIA4. プロジェクト管理	IT-SIA5. テストと品質保証	IT-SIA6. 組織の特性	IT-SIA7. アーキテクチャ						
システム基盤	9	IT-NET ネットワーク	IT-NET1. ネットワークの基礎	IT-NET2. ルーティングとスイッチング	IT-NET3. 物理層	IT-NET4. セキュリティ	IT-NET5. アプリケーション分野	IT-NET6. ネットワーク管理							
	10	CE-NWK テレコミュニケーション	CE-NWK0. 歴史と概要	CE-NWK1. 通信ネットワークのアーキテクチャ	CE-NWK2. 通信ネットワークのプロトコル	CE-NWK3. LANとWAN	CE-NWK4. クライアントサーバコンピューティング	CE-NWK5. データのセキュリティと整合性	CE-NWK6. ワイヤレスコンピューティングとモバイルコンピューティング	CE-NWK7. データ通信	CE-NWK8. 組込み機器向けネットワーク	CE-NWK9. 通信技術とネットワーク概要	CE-NWK10. 性能評価	CE-NWK11. ネットワーク管理	CE-NWK12. 圧縮と伸張
			CE-NWK13. クラスタシステム	CE-NWK14. インターネットアプリケーション	CE-NWK15. 次世代インターネット	CE-NWK16. 放送									
	11	IT-PT プラットフォーム技術	IT-PT1. オペレーティングシステム	IT-PT2. アーキテクチャと機構	IT-PT3. コンピュータインフラストラクチャ	IT-PT4. デプロイメントソフトウェア	IT-PT5. ファームウェア	IT-PT6. ハードウェア							
	12	CE-OPS オペレーティングシステム	CE-OPS0. 歴史と概要	CE-OPS1. 並行性	CE-OPS2. スケジューリングとディスパッチ	CE-OPS3. メモリ管理	CE-OPS4. セキュリティと保護	CE-OPS5. ファイル管理	CE-OPS6. リアルタイムOS	CE-OPS7. OSの概要	CE-OPS8. 設計の原則	CE-OPS9. デバイス管理	CE-OPS10. システム性能評価		
コンピュータアーキテクチャとハードウェア	13	CE-CAO コンピュータのアーキテクチャと構成	CE-CA00. 歴史と概要	CE-CA01. コンピュータアーキテクチャの基礎	CE-CA02. メモリシステムの構成とアーキテクチャ	CE-CA03. インタフェースと通信	CE-CA04. デバイスサブシステム	CE-CA05. CPUアーキテクチャ	CE-CA06. 性能・コスト評価	CE-CA07. 分散・並列処理	CE-CA08. コンピュータによる計算	CE-CA09. 性能向上			
複数領域にまたがるもの	14	IT-ITF IT基礎	IT-ITF1. ITの一般的なテーマ	IT-ITF2. 組織の問題	IT-ITF3. ITの歴史	IT-ITF4. IT分野（学科）とそれに関連のある分野（学科）	IT-ITF5. 応用領域	IT-ITF6. IT分野における数学と統計学の活用							
	15	CE-ESY 組込みシステム	CE-ESY0. 歴史と概要	CE-ESY1. 低電力コンピューティング	CE-ESY2. 高信頼性システムの設計	CE-ESY3. 組込み用アーキテクチャ	CE-ESY4. 開発環境 [25-Ⅰ-1, 2, 3, 4]	CE-ESY5. ライフサイクル	CE-ESY6. 要件分析	CE-ESY7. 仕様定義	CE-ESY8. 構造設計	CE-ESY9. テスト	CE-ESY10. プロジェクト管理	CE-ESY11. 並行設計（ハードウェア、ソフトウェア	CE-ESY12. 実装
			CE-ESY13. リアルタイムシステム設計	CE-ESY14. 組込みマイクロコントローラ	CE-ESY15. 組込みプログラム [25-Ⅰ-1, 4]	CE-ESY16. 設計手法	CE-ESY17. ツールによるサポート [25-Ⅰ-5]	CE-ESY18. ネットワーク型組込みシステム	CE-ESY19. インタフェースシステムと混合信号システム	CE-ESY20. センサ技術	CE-ESY21. デバイスドライバ [25-Ⅰ-1]	CE-ESY22. メンテナンス	CE-ESY23. 専門システム	CE-ESY24. 信頼性とフォールトトレランス	

4. OSS モデルカリキュラム固有の知識

OSS モデルカリキュラム固有の知識として、組み込み Linux 環境における開発支援ツールの利用手法がある。デバッガやエミュレータなどを Linux 上で実践的に利用してデバッグを行う手法を含む。

科目名	第1回	第2回	第3回	第4回	第5回	第6回	第7回	第8回
25.組み込み開発環境に関する知識Ⅰ	(1)組み込みシステムの特徴と開発環境 (2)組み込みLinux 実装 (3)アプリケーション開発 (4)インタフェース利用プログラミング	(1)組み込み開発環境の必要要素 (2)統合開発環境の構築	(1)組み込み型コンピュータ応用システムの開発技術 (2)組み込みコンピュータ応用システム開発支援ツール	(1)基本的なデバッグの方法 (2)トレース機能 (3)複雑に絡み合うアプリケーションとデバイスドライバ間のデバッグ (4)ハードウェアの関連するデバッグ	(1)デバッガソフトによるカーネルやアプリケーションのデバッグ (2)エミュレータの必要性	(1)ハードウェアエミュレーションとは (2)ICE とは (3)ICE によるハードウェア関連のデバッグ (4)Linux 対応ICE（UniSTAC/J、UniSTAC Ⅱ/J、UniSTAC/ASSP、EJ-	(1)ツールチェーンとは (2)ツールチェーンの種類と特徴	(1)デバッグ対象と方法 (2)割り込み発生時のトラブル解消

（網掛け部分は IT 知識体系で学習できる知識を示し、それ以外は OSS モデルカリキュラム固有の知識を示している）

スキル区分	OSS モデルカリキュラムの科目	レベル
組み込み分野	25　組み込みシステムに関する知識 I	基本
習得ポイント	I-25-1. 組み込みシステムの開発の概要	
対応するコースウェア	第1回（組み込み開発の流れと環境）	

I-25-1. 組み込みシステムの開発の概要

組み込みシステムを開発するための開発環境についてその基本構成と特徴を概説し、必要なハードウェア、ソフトウェア、ツール群を紹介する。

【学習の要点】

＊ 組み込みソフトウェアの開発には、ハードウェア開発に依存して行われる部分と、それとは独立して行われる部分とがある。

＊ 組み込みソフトウェア開発は主に PC 上（ホストシステム）で行われ、開発されたシステムの実行は組み込みシステム（ターゲットシステム）で行われる。ホストシステムとターゲットシステムの異なる環境での開発をクロス開発と呼ぶ。

＊ ホストシステム上のコンパイラとリンカはターゲットシステムで実行可能なプログラムを生成する。このようなコンパイラをクロスコンパイラと呼ぶ。

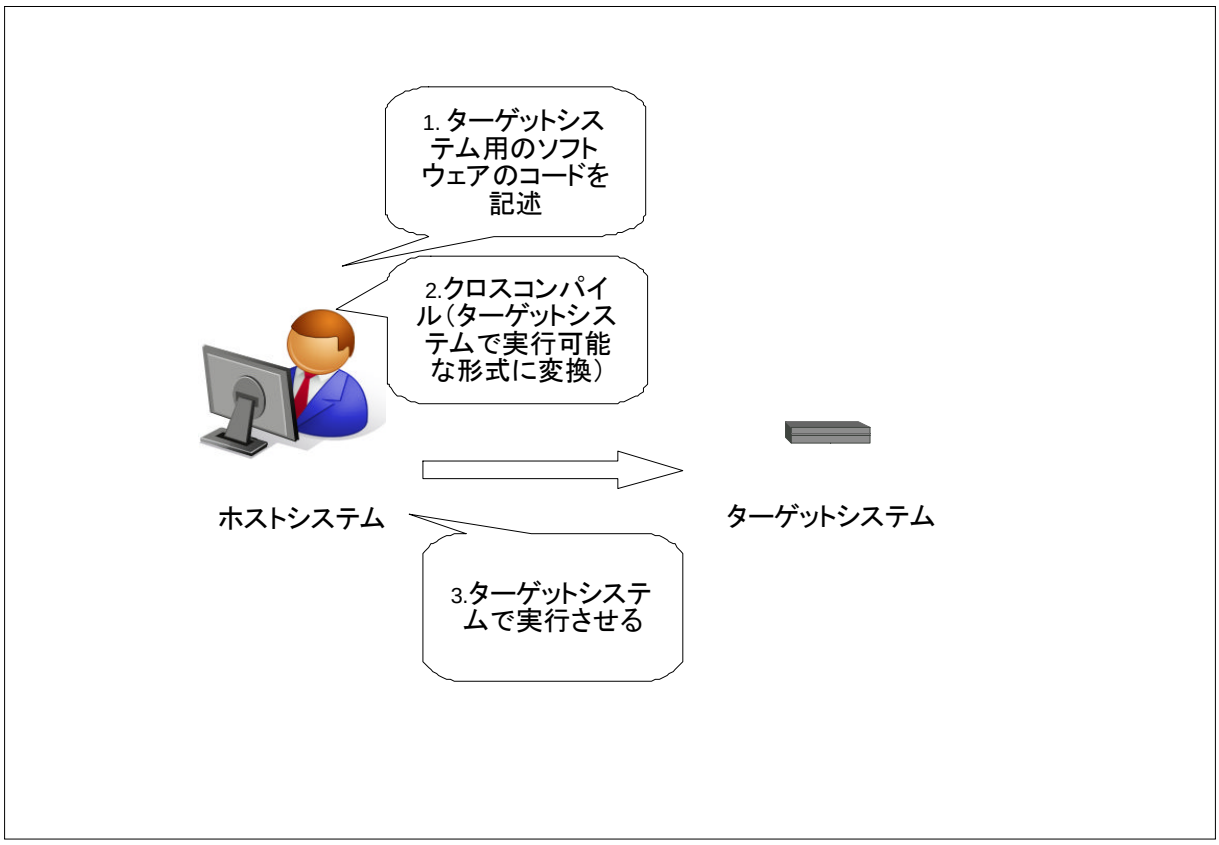

図 I-25-1.　組み込みシステム開発の概略

【解説】

1) 組み込みソフトウェア開発の特徴

＊ 組み込みソフトウェア開発では、ハードウェア開発に依存して行われる部分と、それとは独立して行われる部分とがある。

＊ 組み込みソフトウェア開発の特徴としては、クロス開発を行うことが挙げられる。

- クロス開発とは、ターゲットシステムと異なるホストシステム上で開発することを指す。
- ホストシステムは PC などの汎用コンピュータを指す。
- ターゲットシステムは開発対象の組み込みシステムを指す。
- 開発されたシステム・アプリケーションの実行はターゲットシステム上で行われる。

＊ クロス開発に対応したコンパイラはクロスコンパイラと呼ばれる。

- クロスコンパイラとリンカはターゲットシステムで実行可能な形式のプログラムを生成する。

＊ 実行可能な形式のプログラムは、専用の開発ツール群を用いてターゲットシステムに転送、実行、デバッグする。

2) アプリケーション開発の流れ

＊ 組み込みシステム開発では、システム設計を行った後、ハードウェア・ソフトウェアの並行開発を経て、最終的な製造試作ボードの大量生産を行う。

＊ システム設計では、設計基準に基づくシステムの概念設計を行う。

＊ ハードウェアの開発は、試作品の製造、評価、実機の製造などに一定の期間を必要とする。

＊ ソフトウェア開発の工程では、環境依存・非依存それぞれのソフトウェアの設計・開発・製造・結合テストを経たのち、ソフトウェア全体の結合テスト、製造試作ボードを利用した最終テスト・評価を行う。

- 最終的にソフトウェアを ROM (Read Only Memory)と呼ばれるメモリに焼き付ける。

スキル区分	OSS モデルカリキュラムの科目	レベル
組み込み分野	25　組み込みシステムに関する知識 I	基本
習得ポイント	I-25-2. 組み込み Linux による実装とその開発	
対応する コースウェア	第1回（組み込みコンピュータシステムとは何か）	

I-25-2. 組み込み Linux による実装とその開発

組み込み Linux を用いた組み込みシステムの実装について説明する。ブートローダから Linux 実装までの基本構成や、実際のアプリケーション開発方法、プログラミングから ROM 化までの手順について、組み込み Linux システムを題材に解説する。

【学習の要点】

＊ 組み込み環境での Linux のシェアが大きくなっている理由として、OSS であり、必要に応じてメンテナンスが行えること、標準 API が存在していること、開発環境が充実していることなどが挙げられる。

＊ 組み込み Linux の実装として、ブートローダ、ファイルシステム、カーネルのビルド、デバイスドライバの用意、クロスコンパイル、動作確認といった手順を踏む。

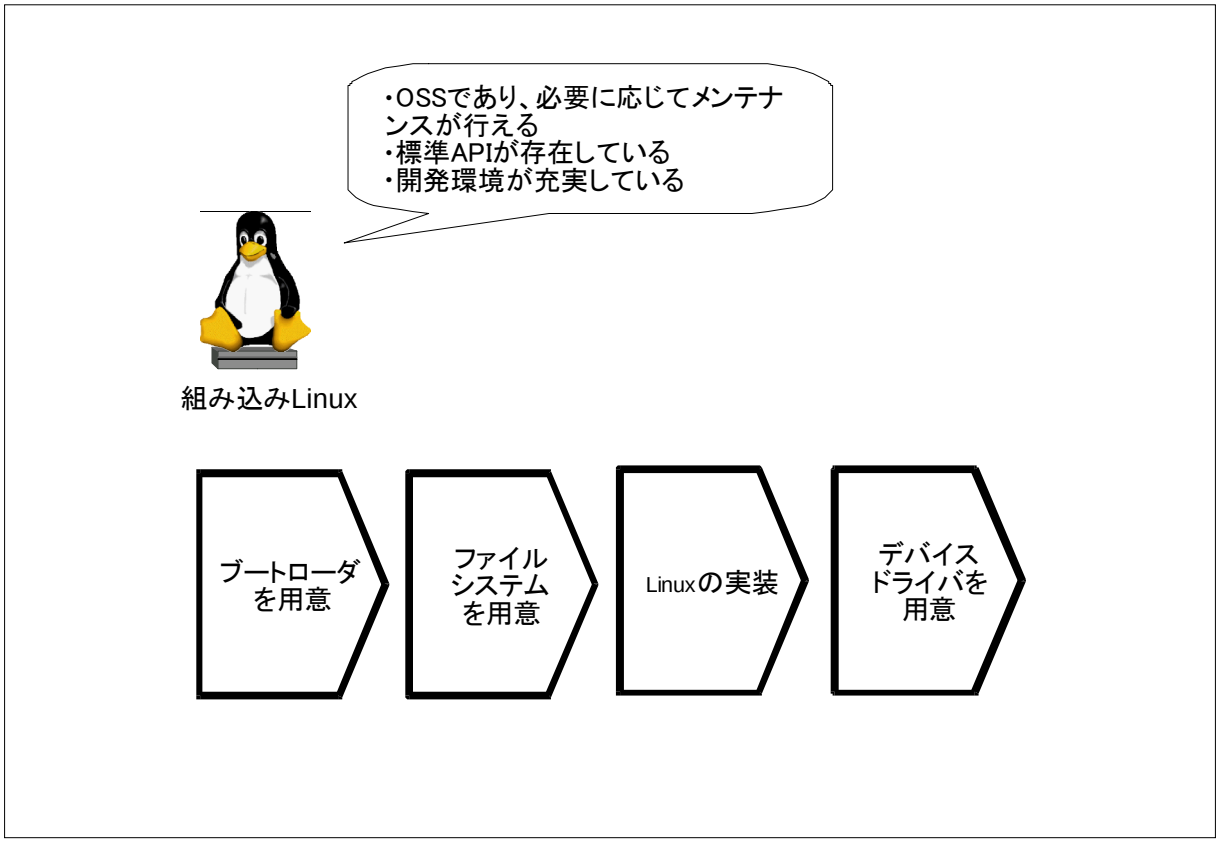

図 I-25-2.　組み込み Linux の長所と開発手順

【解説】

1) 組み込み Linux

＊ 組み込み Linux の特徴、特に長所としては以下のことが言える。

- OSS であり、必要に応じてメンテナンスが行える
- 標準 API が存在している
- 開発環境が充実している

＊ 組み込み Linux の特徴、特に短所としては以下のことが言える。

- リアルタイム性がリアルタイム OS に比べ劣る
- ハードウェア資源の消費量が大きい

＊ 組み込み Linux 開発環境

- ホスト PC およびターゲットシステム上の基盤
  実際にプログラム開発を行うホスト PC、作成した実行ファイルを動作させるターゲットシステム上の基盤に Linux を導入する。

- クロスコンパイル環境構築
  クロスコンパイラ、クロスライブラリ、クロスデバッガがクロス開発には必要である。クロスコンパイラは作成したプログラムをターゲット用にコンパイルし、クロスライブラリはその際に必要となる。クロスデバッガはターゲット上で動作するプログラムをデバッグする負担を低減する。Linux の場合、GNU のツールチェーン(I-25-9 参照)を用いることが多い。

2) 組み込み Linux 実装

＊ ブートローダ

- ターゲット上で Linux を動作・起動させる。
- ブートローダにより、カーネルイメージをターゲットのメモリ上にロードする。

＊ ファイルシステム

- ターゲットに載せるファイルシステムを決定(または開発)する。

＊ Linux の実装

- ターゲットの制約を満たすカーネルをビルドする

＊ デバイスドライバ

- 必要なデバイスドライバを用意(または開発)する。

＊ アプリケーションの開発とクロスコンパイル

- アプリケーションを開発しクロス開発を行う。

＊ 動作確認

- ターゲットに転送し、実行する。

スキル区分	OSS モデルカリキュラムの科目	レベル
組み込み分野	25　組み込みシステムに関する知識 I	基本
習得ポイント	I-25-3. 組み込みシステムの開発環境	
対応する コースウェア	第 2 回（組み込み開発環境の概要）	

I-25-3. 組み込みシステムの開発環境

組み込みシステムの開発環境について、全体像とその構成、特徴や開発手法などを説明する。ターゲット OS、言語、ミドルウェア、プログラミング方法、エミュレーション、デバッグといった話題について触れ、統合開発環境が持つ管理機能についても言及する。

【学習の要点】

＊ 組み込み開発環境を行う上で必要な要素は、OS、プログラミング言語処理系、ミドルウェア、ドライバ、プログラミング支援ツール、エミュレータである。

＊ 統合開発環境は、編集、コンパイル、リンク、デバッグを簡単に行えるように、各種のツールを統合的に提供するツールである。

＊ 統合開発環境は、プログラム履歴のバージョン管理、複数のプログラマによる開発支援など、組み込みソフトウェアの開発規模の増大するにつれ必須となる機能を備え、実際に組み込みシステム開発の現場でも使われている。

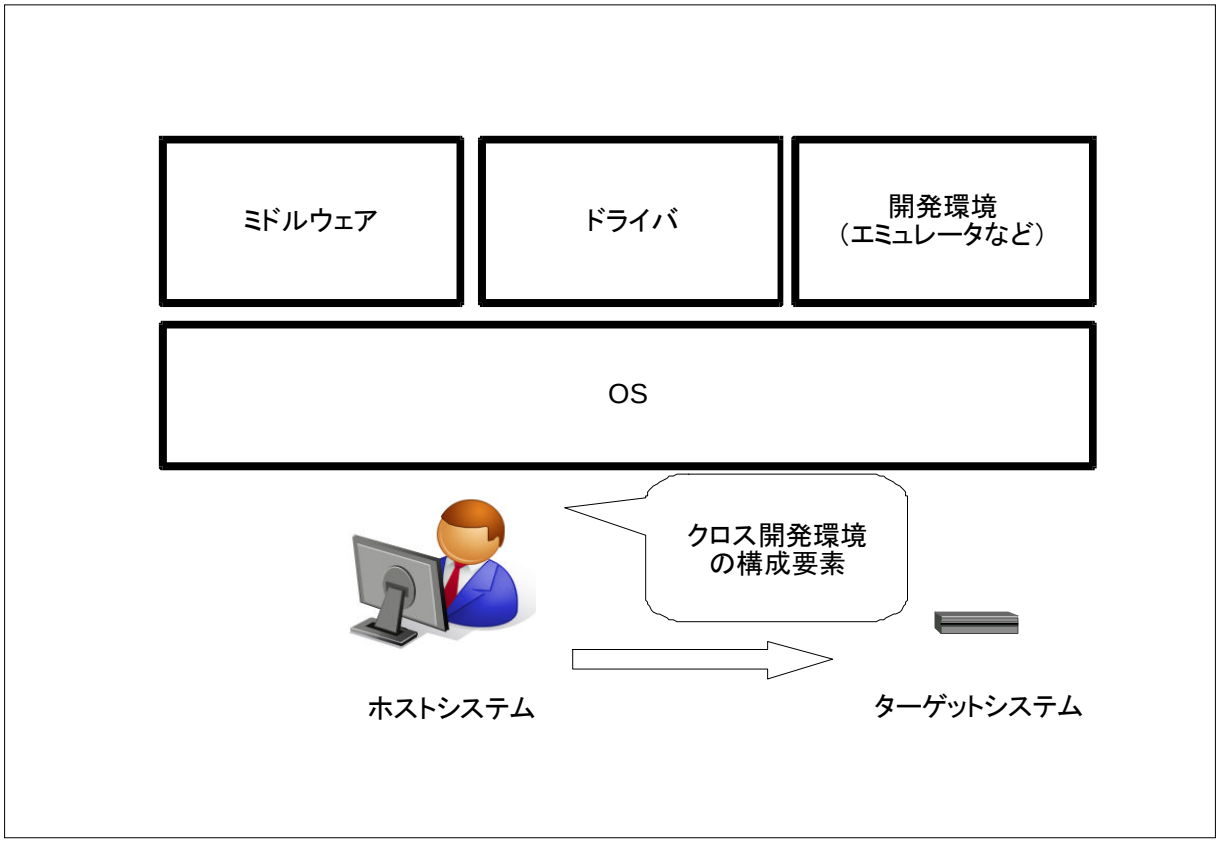

図 I-25-3.　クロス開発環境の構成要素

【解説】

1) 組み込み開発環境の必要要素

＊ OS

組み込み Linux の他 OS に関係する話題として以下の項目がある。

- iTRON
  日本発の機器組込みシステム用のリアルタイム OS の仕様。iTRON 準拠の OS は組み込み OS におけるシェアの半分近くを占める。

- OSEK
  欧州発の車載制御用 ECU（電子制御ユニット）であり、業界標準仕様である。

＊ ミドルウェア

- OS に無い特定用途に使われる機能をモジュール化したもの
- 画像圧縮伸張などが例として挙げられる。

＊ ドライバ

- アプリケーションからのハードウェア制御を可能にするソフトウェアである。
- 個々のプログラムにハードウェア制御のコードを書く負担を軽減する。
- 記述負荷の軽減に加え、一般に効率的な実行ができるようになっている。

＊ プログラミング支援ツール

- ルールチェッカ
- コンパイラ・クロスコンパイラ
- フラッシュ開発キット
- PROM プログラマ
- オンボードプログラマ

＊ エミュレータ

- ターゲット上の CPU の動作を実装し、デバッグ機能を備える。

2) 統合開発環境の構築

＊ 統合開発環境は、プログラム履歴のバージョン管理、複数のプログラマによる開発支援など、組み込みソフトウェアの開発規模の増大に伴い必須となる機能を備える。

＊ 統合開発環境の機能は、一般に以下のような分類ができる。

- プラットフォーム管理
- エミュレーション管理
- インターフェイス管理
- ドキュメント管理

スキル区分	OSS モデルカリキュラムの科目	レベル
組み込み分野	25　組み込みシステムに関する知識 I	基本
習得ポイント	I-25-4. 組み込みシステムの開発技術	
対応するコースウェア	第 3 回 （組み込み開発環境を用いた開発手順）	

I-25-4. 組み込みシステムの開発技術

組み込みシステム開発の全体像について、システム全体の設計段階、ハードウェア設計段階、ソフトウェア設計段階からそれぞれの開発を経てシステム提供に至るまでの手順を説明する。またそれぞれの手順で利用される開発環境やツールを紹介し、それらに求められる機能についても解説する。

【学習の要点】

＊ ホストシステム上で、ソースコードの作成・修正を行い、ソースプログラムをコンパイラとリンカによって実行可能形式プログラムへ変換する(コンパイル・ビルド)。その実行可能形式プログラムを各種のツールを使用してターゲットシステムへ送り込む(ダウンロード)。

＊ ターゲットシステムに送られたプログラムは、ホストの監視を受けながら実行されデバッグされる。エラーが発見されれば、再度ソースコードの編集を行い、コンパイル、ビルド、ダウンロードのデバッグ作業を繰り返す。

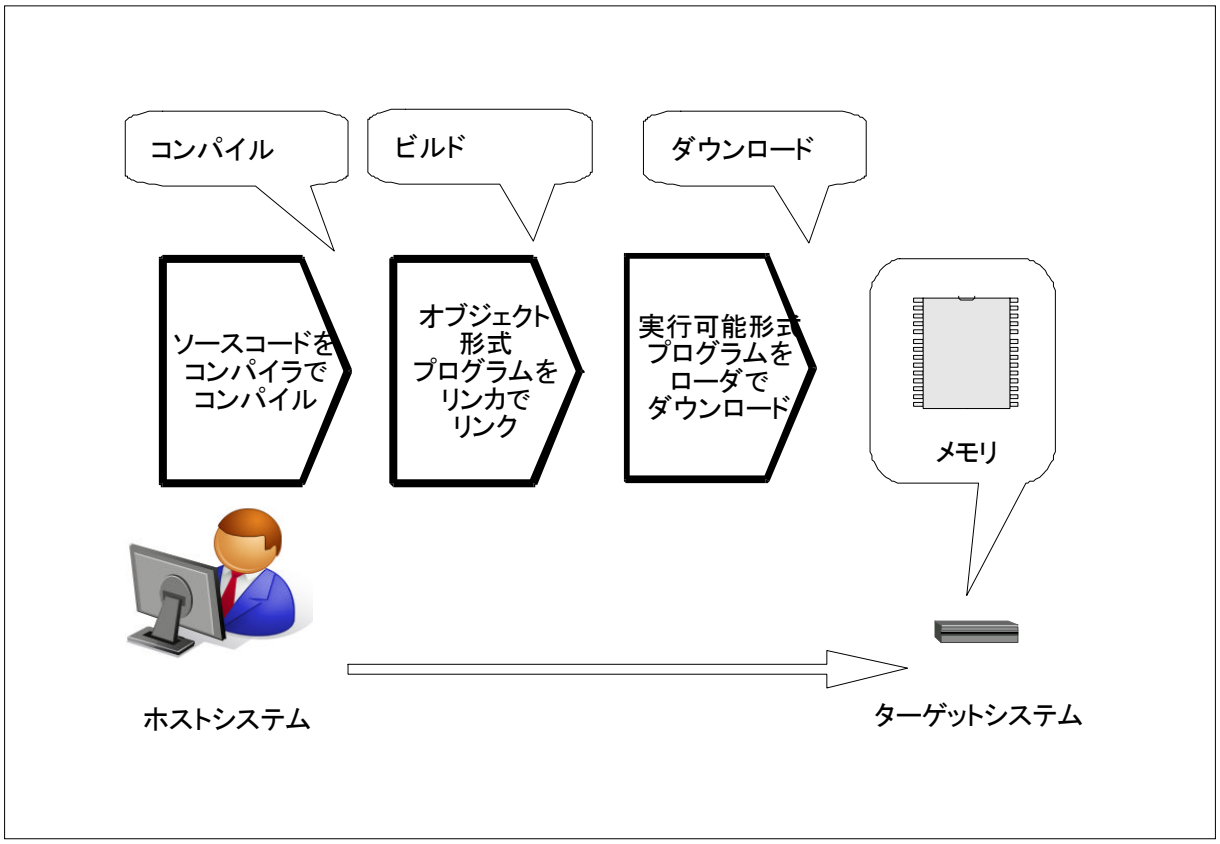

図 I-25-4.　クロス開発の手順

【解説】

1) 組み込みアプリケーションの開発技術

＊ 組み込みアプリケーションの開発は、一般に以下の工程で行われる。

- システム設計
  機能実現の手法を確定し、その実現のための要求仕様を確定させる。性能に対して要求がある場合は、CPU の処理速度と実行するソフトウェア規模などを概算し、要求を満たす実現性の確認を行う。
- ハードウェア設計・開発
  一般にソフトウェア設計・開発と並行に行われる。設計試作ボード、製造試作ボードなどがソフトウェア設計・開発のために適宜提供される。
- ソフトウェア設計・開発
  一般にハードウェア設計・開発と並行に行われる。ハードウェアの開発段階と相互に依存する。

2) 組み込みコンピュータ応用システム開発支援ツール

＊ クロスコンパイラ

- ホストシステム上で、ソースコードの作成・修正を行い、ソースプログラムをコンパイラとリンカによって実行可能形式プログラムへ変換する(コンパイル・ビルド)。
- その実行可能形式プログラムを各種のツールを使用してターゲットシステムへ送り込む(ダウンロード)。

＊ クロスデバッガ

- ターゲットシステムに送られたプログラムは、ホストの監視を受けながら実行されデバッグされる。
- エラーが発見されれば、再度ソースコードの編集を行い、コンパイル、ビルド、ダウンロードのデバッグ作業を繰り返す。

スキル区分	OSS モデルカリキュラムの科目	レベル
組み込み分野	25　組み込みシステムに関する知識 I	基本
習得ポイント	I-25-5. 組み込みソフトウェアのデバッグの基本手順	
対応する コースウェア	第 4 回 （プログラムデバッグの環境）	

I-25-5. 組み込みソフトウェアのデバッグの基本手順

デバッグの基本であるプログラムのトレースにおいて、組み込みシステムならではの特徴について説明し、トレースの基本手順を解説する。またリアルタイム処理に係るトレースの留意点について説明する。

【学習の要点】

* 基本的なデバッグには、printk 命令、GDB(The GNU Project Debugger)、モニタプログラム等を利用した方法がある。
* トレースの基本機能にはターゲット実行データの記録、現象のシミュレート、秒単位のイベントの検知などがあり、組み込みシステムならではのトレース機能の拡張としては長時間トレース、分岐トレース、ハードディスクドライブの利用などがある。

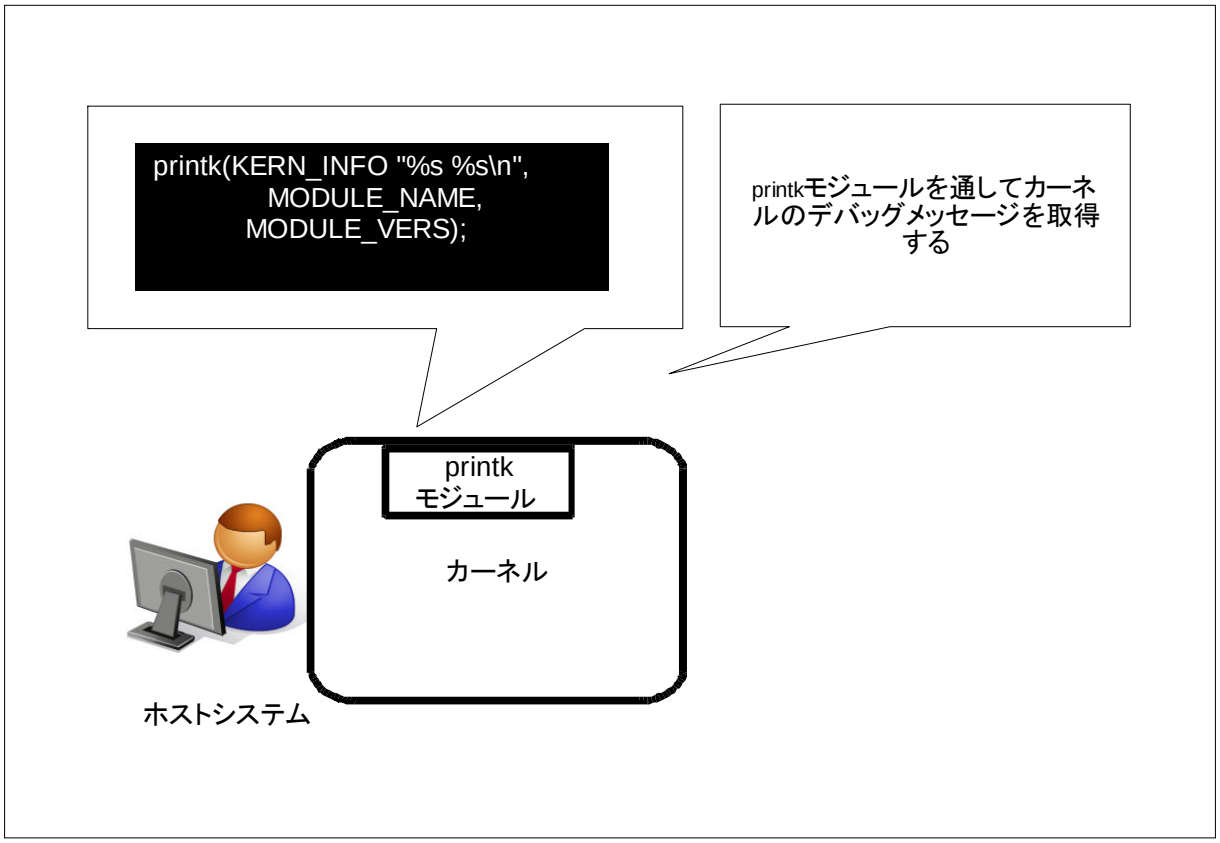

図 I-25-5.　printk()を利用したデバッグ

【解説】

1) 基本的なデバッグの方法

＊ printk()など基本的な命令を使ったデバッグ

- 例えば、printk()は一般的に利用されるメッセージ出力であり、Linux カーネルからのデバッグ情報を収集する際に利用される。

＊ GDB (The GNU Project Debugger) を使用したデバッグ

- ソースコードレベルのデバッグを行う。例えば、ブレークポイントを設定したステップ実行によるデバッグを行う。

＊ モニタプログラムを利用したデバッグ

- ターゲットにホストのデバッガのエージェントプログラム(ROM モニタ)を動作させ、そのエージェントプログラムを使ってデバッグ対象のプログラムを操作する。

2) トレース機能

＊ トレースの基本機能は以下の項目が挙げられる。

- ターゲット実行データの記録
- 現象のシミュレート
- 秒単位のイベントの検知
- トレースメモリの評価
- データトレース

＊ トレース機能の拡張としては以下の項目が挙げられる。

- 長時間トレース
- 分岐トレース
- ハードディスクドライブの利用
- トリガ設定可能(ポイント、エリア、シーケンシャル、カウントなど)
- タイムスタンプ機能

スキル区分	OSS モデルカリキュラムの科目	レベル
組み込み分野	25　組み込みシステムに関する知識 I	基本
習得ポイント	I-25-6. デバッガを利用したプログラムのデバッグ	
対応する コースウェア	第 5 回（デバッガソフトを使用したデバッグ環境）	

I-25-6. デバッガを利用したプログラムのデバッグ

デバッガを利用したデバッグ環境の全体像と構成、特徴を説明する。デバッグ環境構築上の留意点や組み込みシステムを対象としたデバッグ特有の特徴や留意点などについて説明する。

【学習の要点】

＊ クロス開発において、ホストのファイルとして作成されたプログラムを、ターゲットにロード、実行、デバッグするためのツールをクロスデバッガと呼ぶ。

＊ クロスデバッガがターゲットシステムを制御するための方法には、ソフトウェアデバッガ、ICE(in Circuit Emulator)、JTAG(Joint Test Action Group)の 3 通りが存在する。

＊ ソフトウェアデバッガでは、ターゲットシステムにホストシステムのデバッガソフト（エージェント）を内蔵させ、このソフトを使ってターゲットを操作する。

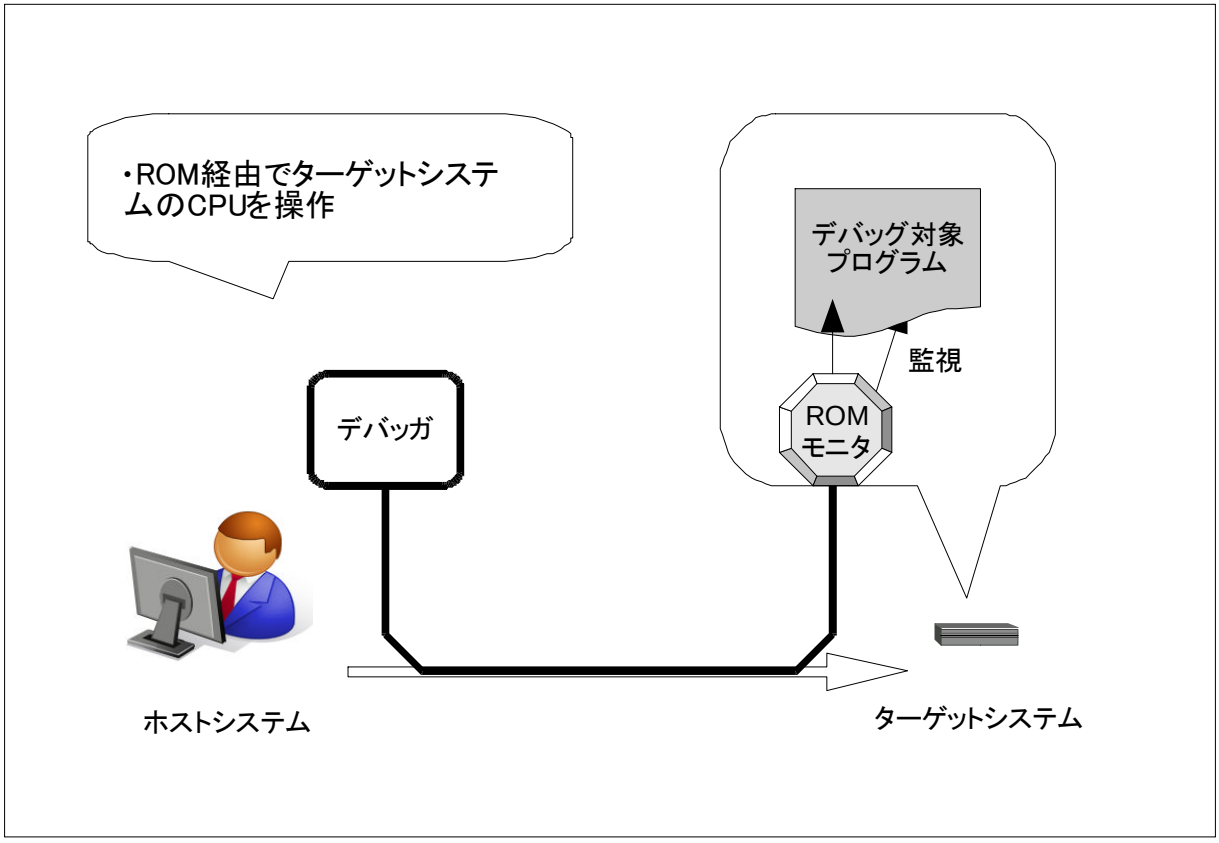

図 I-25-6. モニタプログラムを使ったデバッグ

【解説】

1) クロスデバッガ

＊ クロスデバッガとは

- クロス開発環境において、ホストシステムのファイルとして作成されたプログラムを、ターゲットシステムにロード、実行、デバッグするためのツールをクロスデバッガと呼ぶ。
- GDB はクロスデバッガの機能を持っている。

＊ クロスデバッガの機能

- ローディング機能
  ホストシステムで作成したファイルをターゲットに送り込む。
- ソースコードレベルのデバッグ機能
  ブレークやステップ実行などを行い、変数の値の参照・変更などを行う。
- インクリメンタルデバッグ機能
  デバッグの完了したモジュールを漸次追加しながらデバッグを進めていく。
- シミュレータ機能
  ホストシステム上でターゲットシステムの機械語を擬似的に実行させて結果をデバッグに利用する。

2) デバッガソフトによるカーネルやアプリケーションのデバッグ

＊ ソフトウェアデバッガでは、ターゲットシステムにホストシステムのデバッガソフト(エージェント)を内蔵させ、このソフトを使ってターゲットを操作する。このソフトを ROM モニタと呼ぶこともある。

＊ ROM モニタは、シリアル、パラレル、イーサネットなどを使ってホストと接続される。

＊ ROM モニタは、デバッグ対象のプログラムと一緒にリンクしてターゲットの ROM や RAM に配置する場合と、ROM モニタだけを ROM に常駐させる場合とがある。

スキル区分	OSS モデルカリキュラムの科目	レベル
組み込み分野	25　組み込みシステムに関する知識 I	基本
習得ポイント	I-25-7. エミュレータの利用方法	
対応する コースウェア	第 6 回 （ICE を使用したデバッグ環境）	

I-25-7. エミュレータの利用方法

エミュレータとは何か、その基本構成と特徴、役割を説明する。またエミュレータを用いたデバッグやプログラム実行の検証、解析方法について解説を加える。

【学習の要点】

＊ エミュレータはあるシステムの機能を異なるシステムにより再現するためのツールである。組み込みシステム開発で使われる ROM エミュレータは実装される ROM の動作を再現する。これにより、実際に ROM 化する作業コストを大幅に低減できる。

＊ エミュレータを使うことにより、プロセスの状態遷移解析、プロセスのパターン解析、プロセスの異常解析、OS の動作に関する検証などをリアルタイムに行うことができる。

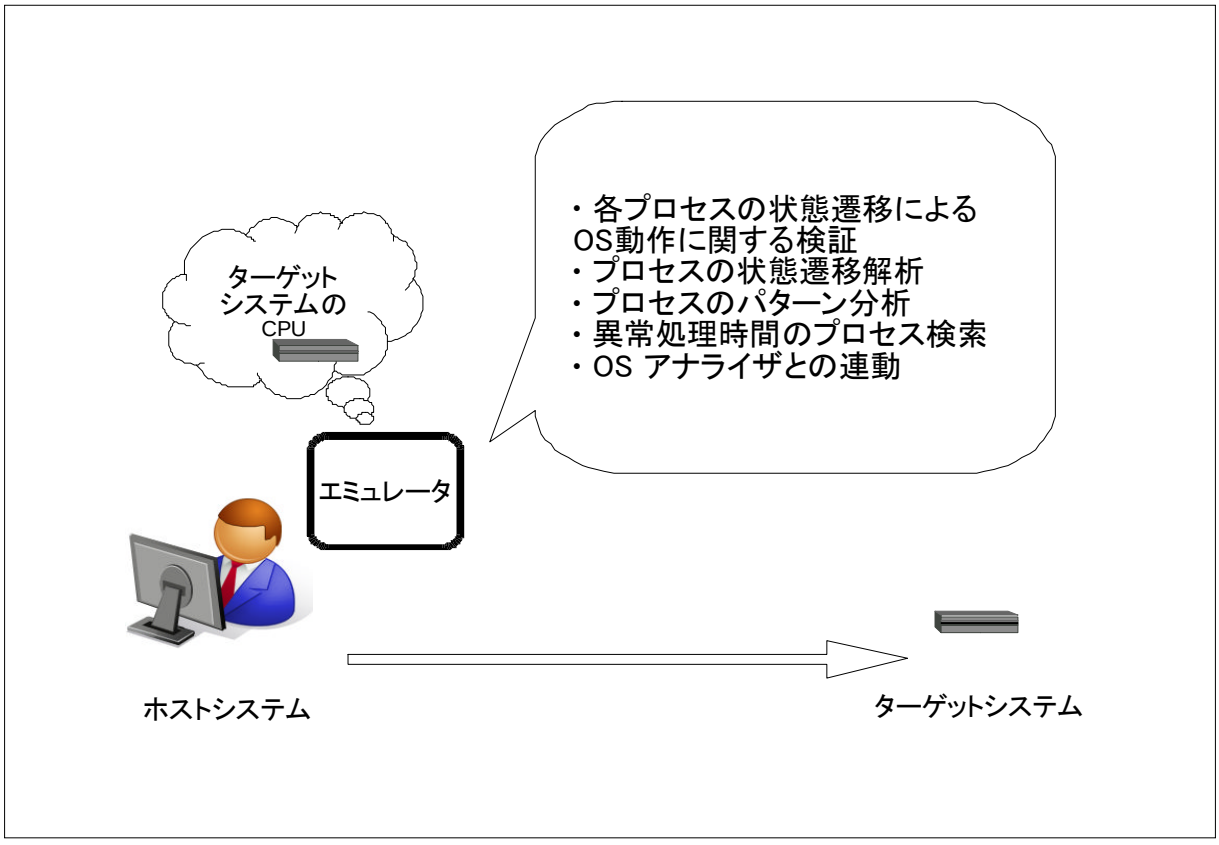

図 I-25-7.　組み込み開発におけるエミュレータ

【解説】

1) エミュレータの必要性

＊ エミュレータはあるシステムの機能を異なるシステムにより再現するためのツールである。

＊ ソフトウェアデバッガには ROM 上のプログラムに対してブレークポイントを設定する際などに問題があるため、以下の項目に対応するためにはエミュレータの仕組みが必要である。

- 各プロセスの状態遷移による OS の動作に関する検証
- プロセスの状態遷移解析
- プロセスのパターン分析
- 異常処理時間のプロセス検索
- OS アナライザとの連動

＊ また、組み込みシステム開発で使われる ROM エミュレータは、実装される ROM の動作を再現するため、実際に ROM 化する作業コストを大幅に低減できる。

2) ハードウェアエミュレーションとは

＊ エミュレーションのための条件

- チップ内部に ROM を内蔵していなければ ROM エミュレータは利用できる。
- システムバスが開放されていれば ICE(第 8 回で学習)というシステムバスに直結させるエミュレーション手法が利用できる。

＊ エミュレーションする内容

- ハードウェアエミュレータは、CPU の諸機能だけでなく、必要に応じてピンなどもエミュレートする。

スキル区分	OSS モデルカリキュラムの科目	レベル
組み込み分野	25　組み込みシステムに関する知識 I	基本
習得ポイント	I-25-8. ICE を利用したデバッグ環境	
対応するコースウェア	第 6 回 （ICE を使用したデバッグ環境）	

I-25-8. ICE を利用したデバッグ環境

組み込みシステムに特有のハードウェアエミュレーションについて説明し、ICE (In Circuit Emulator) の必要性と目的、使い方、ICE を利用したハードウェア関連のデバッグ方法を解説する。また組み込み Linux に対応した代表的な ICE を紹介する。

【学習の要点】

* ハードウェアに近いデバッグを行う際に、古くから使われ続けているツールとして ICE(In Circuit Emulator)がある。ICE ではターゲットシステムの CPU の代わりに ICE の端子を取り付けることで CPU の代替をすることができる。
* プログラムのブレーク(Break)機能はデバッグに欠かせないものとなっている。これは、特定の条件（特定アドレスのフェッチ、特定の信号入力）を指定して、プログラムの実行状態をメモリに保持したまま CPU を停止させるという機能である。
* 代表的な Linux 対応 ICE 製品として、UniSTAC/J、UniSTAC Ⅱ/J、UniSTAC/ASSP、EJ-Debug などがある。

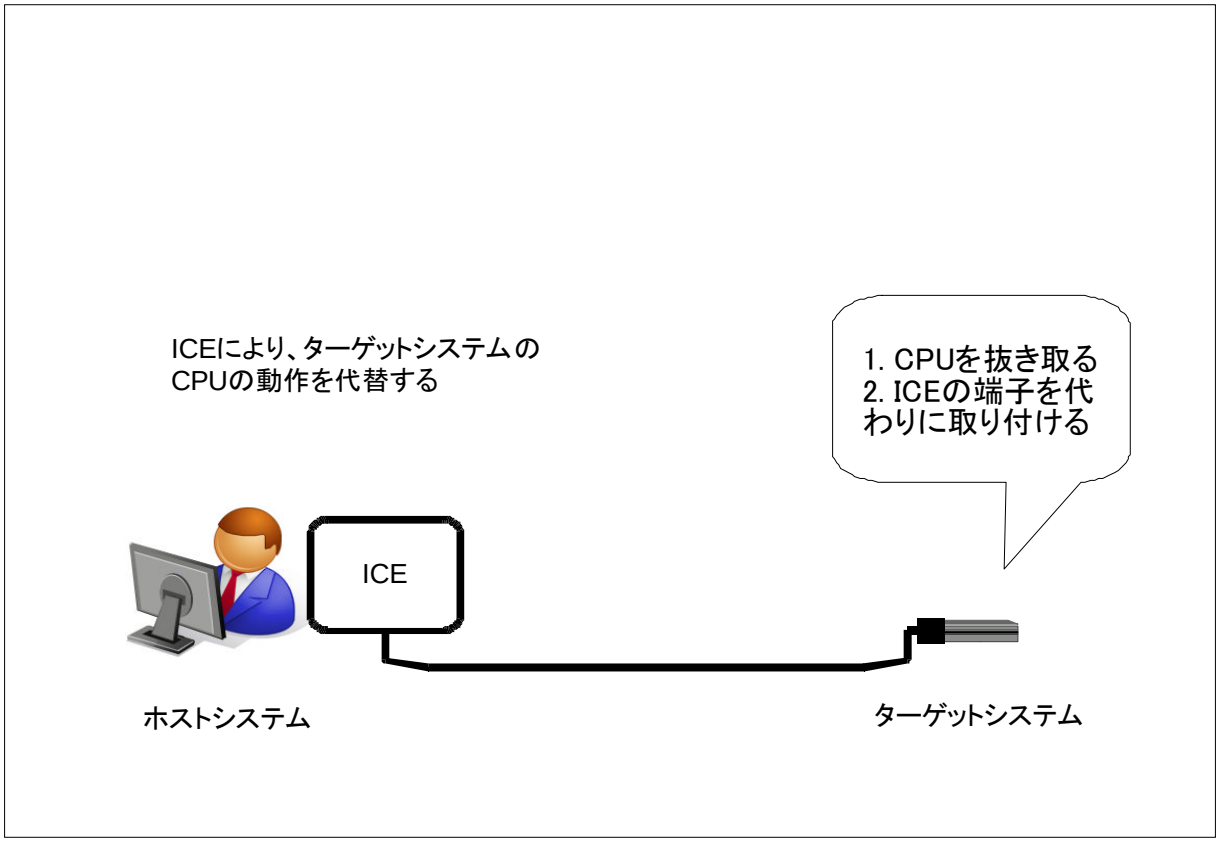

図 I-25-8.　ICE によるデバッグ

【解説】

1) ICE とは

＊ ICE の目的はターゲット上で実行するプログラムをデバッグすることである。

＊ ハードウェアのエミュレーション方式

- ICE はターゲットボード上の CPU を抜き去り、代わりに ICE の端子を取り付ける。
- ICE は端子からバス信号の送受信を行う。
- CPU 命令の全てを代替することができる。

2) ICE によるハードウェア関連のデバッグ

＊ リアルタイムトレース機能

- 外部バス情報をリアルタイムに検証し、リアルタイム実行時に発生するエラーを発見する。

＊ プログラムのブレーク(Break)機能

- 特定の条件（特定アドレスのフェッチ、特定の信号入力）を指定して、プログラムの実行状態をメモリに保持したまま CPU を停止させる。

3) Linux 対応 ICE

Linux 対応 ICE 製品も充実してきており、代表的なものに UniSTAC/J、UniSTAC Ⅱ /J、UniSTAC/ASSP、EJ-Debug がある。

＊ JTAG による接続

- JTAG(Joint Test Action Group) ICE は CPU にテスト用の機能を備えたものであり、フル ICE（CPU を代替する方式）と同等の機能を実現するための方式である。
- また、近年の周辺入出力回路とあわせてワンチップ化したマイコンが主流になり CPU コアだけの置き換えが難しくなった影響もあり、JTAG-ICE が主流になっている。

＊ LAN、USB インターフェイスによる接続

- LAN で用いられる Ethernet ポートや、USB など、多くの組み込み機器が搭載している標準的なインターフェイスを用いてデバッグデータをやりとりする方式も利用される。

スキル区分	OSS モデルカリキュラムの科目	レベル
組み込み分野	25　組み込みシステムに関する知識 I	基本
習得ポイント	I-25-9. ツールチェーンを利用したデバッグ	
対応するコースウェア	第 7 回（ツールチェーンによるデバッグ）	

I-25-9. ツールチェーンを利用したデバッグ

組み込み Linux を用いた組み込みシステムの実装について説明する。ブートローダから Linux 実装までの基本構成や、実際のアプリケーション開発方法、プログラミングから ROM 化までの手順について、組み込み Linux システムを題材に解説する。

【学習の要点】

* 組み込み Linux を用いて、実際に組み込みアプリケーション開発の手順を理解する。
* ツールチェーンとは、一般にコンパイラやリンカ、および各種バイナリ操作プログラムを組み合わせた開発環境の一式のことを指す。
* ツールチェーンにより、クロスコンパイラ機能を実現できる。

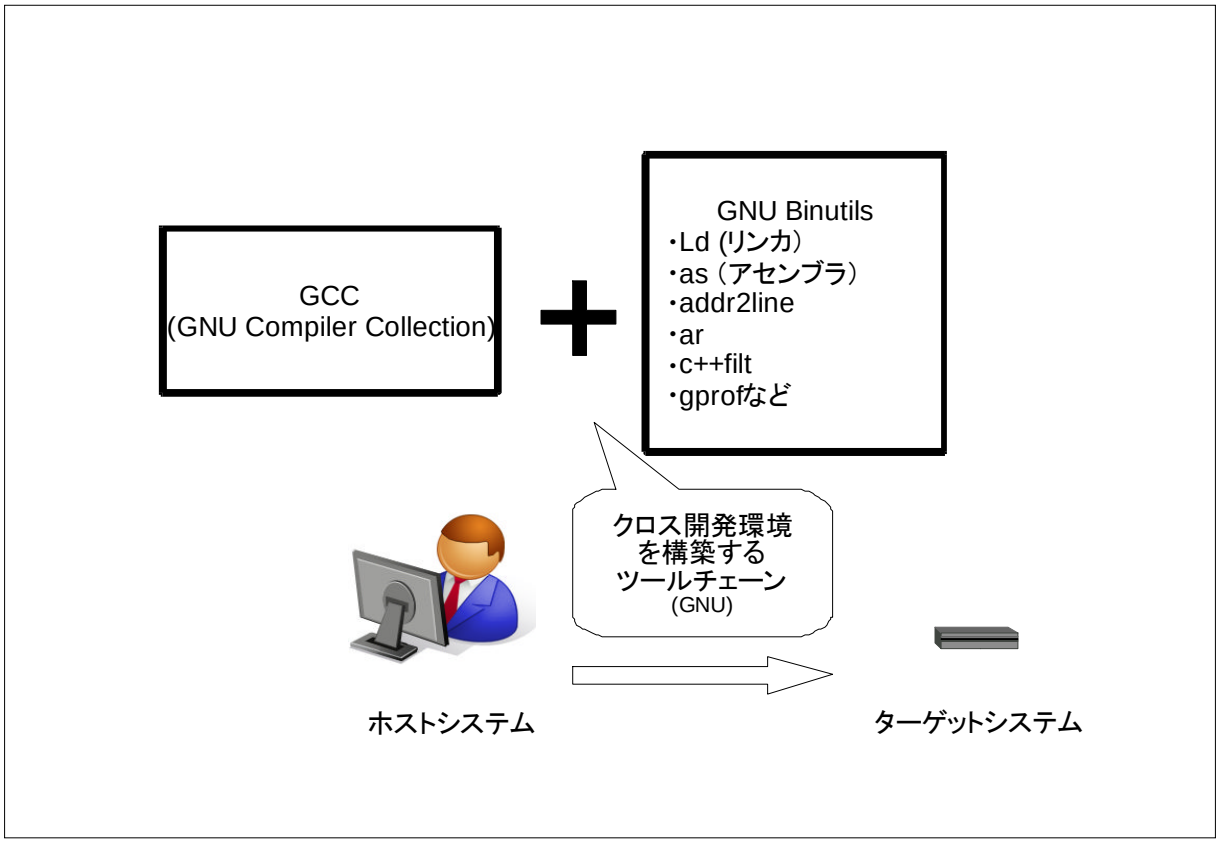

図 I-25-9.　GNU ツールチェーンによるクロス開発環境

【解説】

1) ツールチェーンとは

Linux 環境でのツールチェーンとは、一般にコンパイラと binutils プログラム(GNU Binary Utilities)を組み合わせた開発環境の一式のことを指す。性能的な制約を満たせば、クロスコンパイラ機能、ネイティブコンパイラ機能、双方の役割を果たすことができる。

＊ ツールチェーンの構成

- コンパイラ
  GCC （GNU Compiler Collection）が主に利用される。
- binutils プログラム(GNU Binary Utilities)
  ld （GNU リンカ）、as （GNU アセンブラ） などを含む、 バイナリ操作に用いるプログラムの集合である。

2) ツールチェーンの種類と特徴

＊ GNU ツールチェーン

- 全てのツールは基本的に OSS であり、Linux などの Unix 系 OS では 容易にクロスコンパイラ環境を作成することができる。
- メジャーなターゲット向けにはビルド済みのバイナリが企業やコミュニティにより提供されている場合も多い。

＊ EABI ツールチェーン

- ARM Ltd.の ABI（application binary interface）仕様「ABI for the ARM Architecture（EABI）」に対応したツールチェーンであり、ARM 対応のコンパイラなどを含んでいる。

スキル区分	OSS モデルカリキュラムの科目	レベル
組み込み分野	25　組み込みシステムに関する知識 I	基本
習得ポイント	I-25-10. 実機レベルのデバッグとトラブルシューティング	
対応する コースウェア	第 8 回 （リアルタイムシステムの構成と仕組み）	

I-25-10. 実機レベルのデバッグとトラブルシューティング

組み込みシステムを対象としたデバッグの実際に関して、デバッガや開発環境の制限事項で留意すべき点や、チェックすべき対象、様々な状況下でのトラブルシューティング方法、トラブル解決に利用できるツールなど、様々な話題を紹介する。

【学習の要点】

＊ デバッグの実践的な力をつけるため、組み込みシステムのデバッグと、割り込み発生時のトラブル解消に関するプラクティスを共有する。

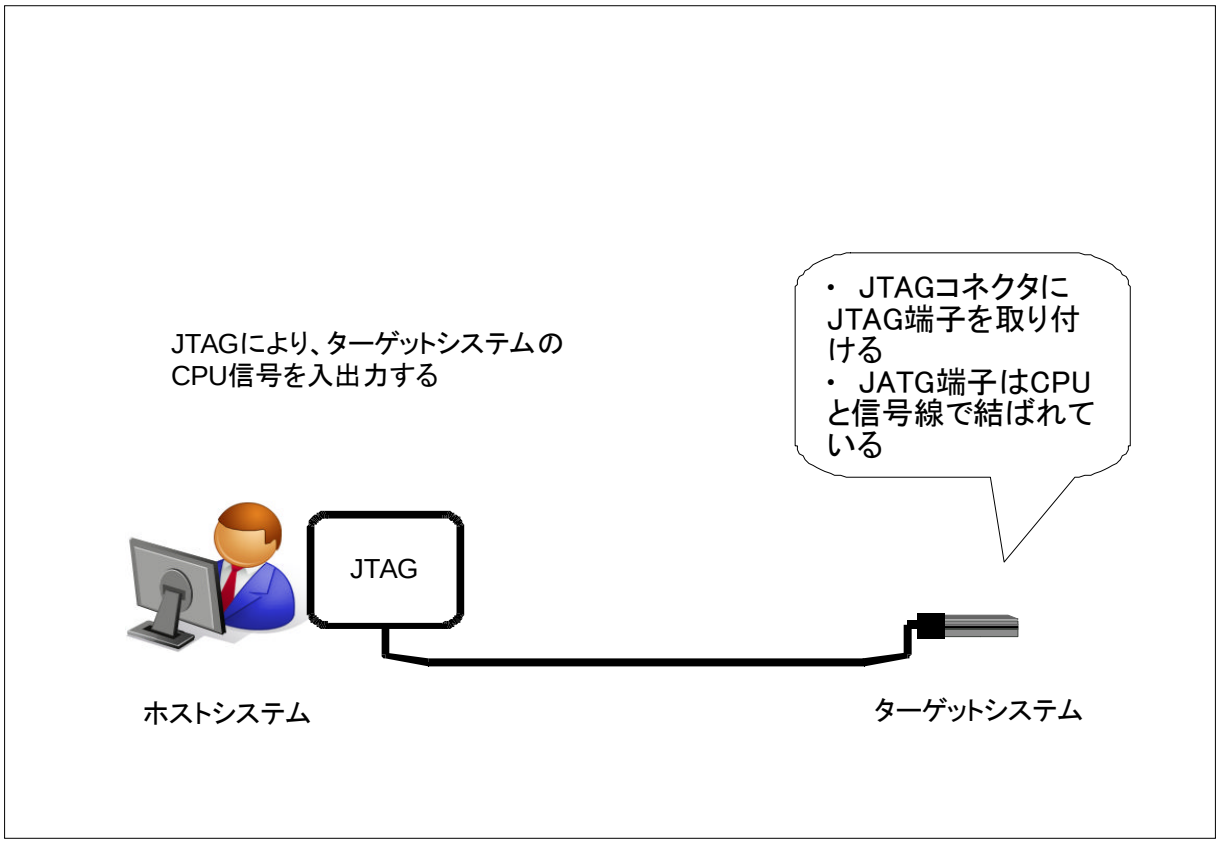

図 I-25-10.　JTAG によるデバッグ

【解説】

1) デバッグ対象と方法

＊ 組み込み Linux のデバッグ環境の向上

- 従来は、シリアルやLAN経由で接続し、さらにモニタプログラムなどをターゲットボード上で動作させる必要があった。
- 最近では Linux 対応 ICE から JTAG への接続が可能になり、デバッグ機能の自由度が増している。また、デバッグのためだけに LAN コネクタを設ける必要がなくなった。

＊ デバッグ対象とその方法

- 実行中のプログラムの変数など、ブレークポイントを利用して確認する。
  このときのブレークポイントの制約が緩和されている。例えばフラッシュ ROM などの読み込み専用メモリでもブレークポイントの設定が可能になっている。
- メモリが正しく読み書きできているか、バスチェック機能により確認する。
- プログラムがリアルタイム動作で正常に動作しているか、リアルタイムトレース機能により確認する。

2) 割り込み発生時のトラブル解消

＊ ICE によるデバッグ環境により不足している資源（メモリ）の補完が可能

- 不足した SRAM、SDRAM 等をエミュレーションメモリで代用
- ROM をエミュレーションメモリで代用

＊ ICE によるシステムのパフォーマンス計測が可能

- システムの動作時間
- カバレッジ
- プロファイル