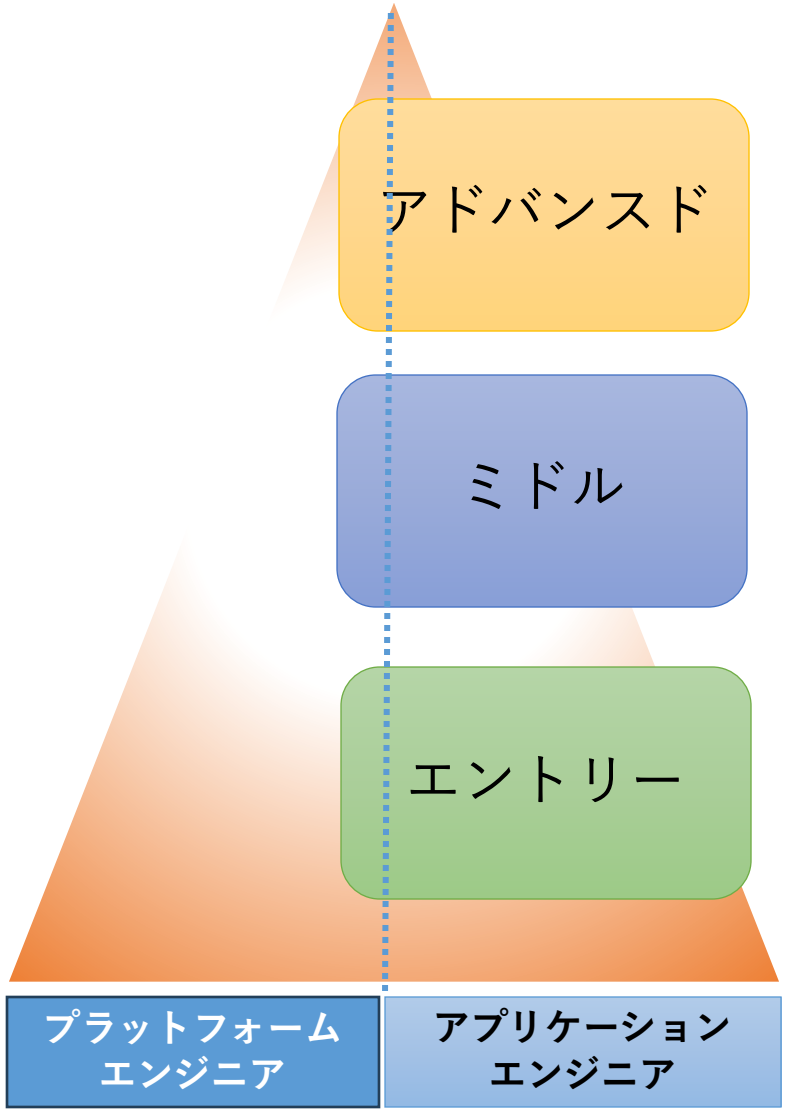
次世代の組込みOSSエンジニアの人物像

一般社団法人
組込みシステム技術協会
Japan Embedded System Technology Association



システム全体のHW、OSのアーキテクチャを理解し、全体設計、実装、システムチューニング、高度なデバッグまで、幅広い専門知識とOSSの効果的な利用と開発能力を発揮、開発チームをリードし、コミュニティと連携、貢献できるエンジニア

システムの一部の技術に精通し、OSの機能理解と十分な実装経験を有し、システムに最適なOSSの選定と利用を通じて、効率的な設計、開発、担当機能のデバッグを開発チームと連携し、独立して実施できるエンジニア

組込みシステム開発プロセスとOS、OSSの基本的な理解と、あらかじめ設計されたシステム仕様と準備された開発環境に基づいて、開発リーダーの指示の下、基礎的なシステム開発と単体テストと基本的なデバッグが実施できるエンジニア

プラットフォームエンジニア：システムの開発基盤、デバイスドライバ、OSパッケージを開発、構築するエンジニア
アプリケーションエンジニア：システム上で機能するアプリケーションを開発するエンジニア

トレーニングプログラムのスキームについて

一般社団法人
組込みシステム技術協会
Japan Embedded System Technology Association

各スキルレベル（エントリー/ミドル/アドバンスド）に応じたトレーニングコースを修了、認定試験をパスすることでエンジニアスキルバッチを付与

組込みOSSエンジニアスキル標準（エントリー）

LF トレーニングコース（例）

- 組込みLinux開発(LFD450)
- LFD461-JP(Yocto)
- OSS開発入門
  (LFD102/103,LFC191)
- Linux 入門
- 組込みLinuxカーネル開発
  入門(LFD435)

Certificate

JASAトレーニングコース（例）

- 組込みLinuxアプリケーション開発入門
- 組込みLinuxシステム管理
- 組込みLinux導入(ビルド)
- 組込みOSS
  SBOM管理入門
- 組込み技術者試験制度
  (ETECクラス1,2)

組込みOSSエンジニアに求めるスキルマップ案

一般社団法人
組込みシステム技術協会
Japan Embedded System Technology Association

レベル	組込みOSSエンジニアの人物像	
	プラットフォームエンジニア	アプリケーションエンジニア
エントリー	組込みソフトウェア開発知識：ETECクラス２の範囲において、グレードB以上のスコアを取得できる知識を有している OSSコンプライアンス：OSSライセンス・コンプライアンス・コミニュティ貢献の知識がある OSSエンジニアとしてできること： ①要件に合わせたカーネルコンフィグの設定 ②既存に組み込まれたデバイスドライバの導入 ③デバイスツリーの修正、およびデバイスドライバの動作確認 ④デバイスドライバの正常動作のためのudevの設定、動作確認のためシステムコール・procfs・sysfsを使ってテストプログラムを作成、実行	組込みソフトウェア開発知識：仕様が決まっているアプリケーションをLinuxのアプリケーション実装セオリーに準拠して実装する能力を有している OSSコンプライアンス：OSSライセンス・コンプライアンス・コミニュティ貢献の知識がある OSSエンジニアとしてできること： ①開発言語：MUST　C言語、WANT:C++言語 ②POSIX（システムコール）、シグナル、スレッド、プロセス間通信（IPC、その他）、ファイル制御、デバイス制御（デバイスファイル経由、sysfs経由）→基本的なLinuxのドライバ仕様を自力で確認が行える ③Systemd、googleTESTなどの管理プログラム、ツール利用が行える
ミドル	組込みソフトウェア開発知識：エントリーレベルと同じ OSSコンプライアンス：エントリーレベルと同じ OSSエンジニアとしてできること：エントリーレベルにプラスして、以下が行える ①既存のデバイスドライバのテストを実施、および不具合（性能期待値でないも含む）原因をソース上で特定 ②新規のデバイスドライバを作成 ③Yoctoなど組込みシステム開発環境構築の実施	組込みソフトウェア開発知識：エントリーレベルにプラスして、自身が作成するプログラム外を活用してシステム構築が可能。 OSSコンプライアンス：エントリーレベルと同じ OSSエンジニアとしてできること：エントリーレベルにプラスして、以下が行える ①開発言語：エントリーレベルにプラスして、RUST利用も可能 ②使用するソースコードのライセンスについてもケア ③ミドルウェアを自力で調査し、導入可能かどうか判断、導入 ④CGROUPを活用してリソース管理 ⑤ファームウエアアップデートも含めたシステムの安全性を配慮したユーザランド設計
アドバンスド	組込みソフトウェア開発知識：エントリーレベルと同じ OSSコンプライアンス：エントリーレベルにプラスして、以下を追加 ①コミニュティ貢献活動を実施 ②OSSライセンス・コンプライアンスの正しい理解、およびソフトウェアを正しく選択・適用 OSSエンジニアとしてできること：ミドルレベルにプラスして、以下が行える ①ブートローダの仕組みの理解と修正 ②ファームウェアアップデート・セキュリティ・性能面を考慮したマイコン上のLinux環境における設計（パーティション構成、RAMFSの使用有無、セキュリティ対策等） ③他ＯＳからの移行をする場合、OS観点（デバイスドライバ含む）の移行方針設計（アプリケーション観点は除外） ④構築したLinux環境について、評価方針の策定・指示、および要件未充足の場合、問題特定・修正実施	組込みソフトウェア開発知識：ミドルウェアレベルにプラスして、組込みシステム全体を考慮した上での設計（対SoC内に存在する他環境との連携）→組込みシステムのライフ設計（寿命等）も意識した上での方針指示 OSSコンプライアンス：プラスして、以下を追加 ①OSSライセンス・コンプライアンスを正しく理解し、ソフトウェアを正しく選択・適用が行える OSSエンジニアとしてできること：ミドルレベルにプラスして、以下が行える ①開発言語：ミドルレベルと同じ ②③セキュリティ、信頼性、リアルタイム性についても考慮 ④コミニュティ活動において貢献

組込みOSSエンジニアスキルマップ案：プラットフォームエンジニア

一般社団法人
組込みシステム技術協会
Japan Embedded System Technology Association

スキルレベル	Linux		習得すべきスキルセット（共通）	RTOS(従来型組込みOS)	
	技術要素（知識）	開発技術（できること）		技術要素（知識）	開発技術（できること）
エントリーレベル	• OSSライセンス/コンプライアンス/コミュニティ貢献の基本 • Linuxの基本構造（ユーザ／カーネル空間、プロセス、メモリ、ファイルシステム） • 基本的なハードウェアI/F（GPIO/I²C/SPI/UART）、ブートローダの基礎（例：U-Boot） • Device Treeの読み方、クロスコンパイル環境、Git/パッチの基礎	• 正しいOSSの利活用と運用知識 • カーネル設定（menuconfig）とビルド • 単純なカーネルモジュールの作成とロード • dmesg/strace/gdbserverを用いた基本的なデバッグ、RootFS作成（BusyBox等）	OSの基本設計概念理解とターゲットボード上でのOS構成・設定とビルド、実行 基本的なドライバー、ミドルウェアの正常動作の確認とデバック手順の習得	• RTOSの基本（タスク／優先度／プリエンプション、割り込み、同期：ミューテックス・セマフォ・イベント、タイマ） • HAL/BSPの基本 • リンカスクリプト • JTAG/SWD	• タスク定義とスレッド制御 • **割り込みハンドラ**の実装 • 基本I/Oドライバ（GPIO/I²C/SPI/UART）実装 • 静的メモリ割り当て • RTOS設定（スタックサイズ／タイムスライス）
ミドルレベル	• BSP構成（ブート→ブートローダ→カーネル→ルートFS） • 主要ファイルシステム（ext4/squashfs/UBI） • ネットワークスタック（TCP/IP、VLAN、CAN等） • 電源管理（cpufreq、suspend） • セキュリティ機構（SELinux/AppArmor） • リアルタイム化（PREEMPT_RTの概念）	• ボード固有のDevice Tree作成・適用 • **キャラクタ／ブロック／ネットワーク系**ドライバの開発・移植 • Yocto/Buildrootでのディストリビューション構築 • perf/ftraceなどでの性能解析 • 起動時間短縮、カーネルパッチ適用と再構成	OSのチューニングとビルドツールを用いたOSのパッケージ・パッチ追加/変更、デバイスドライバーの組込みとビルドなどBSP開発とメンテナンス知識の習得	• ミドルウェア（**ファイルシステム、TCP/IPスタック**等）の組み込み • MPUによる領域保護 • フラッシュ管理（Wear Leveling） • 時刻・イベント管理 • 低消費電力手法（Tickless Idle等）	• lwIP/uIP等のネットワーク導入 • FAT/FlashFSの統合 • **ドライバ仕様書の策定**とAPI設計 • **割り込み遅延**測定・短縮 • ブートシーケンス設計 • HIL（Hardware-in-the-Loop）での動作検証
アドバンスドレベル	• SoCアーキテクチャ（MMU/キャッシュ、DMA、割り込み制御） • スケジューラ／メモリ管理の深い理解 • セキュアブート／Trusted Execution（TPM等）	• 新規SoCへのLinux移植・**ボードブリングアップ** • 複雑な周辺（PCIe/USB/Display/Camera等）のドライバ設計 • リアルタイムチューニング（PREEMPT_RT、IRQバランシング） • BSP全体の最適化・保守方針（Upstream/Downstream）設計	OS全体の深い知識とセキュリティ、リアルタイム処理、厳しい要求に耐えうるパフォーマンスチューニング、複雑なドライバー開発と高度なデバッグ知識を習得 BSP全体の最適化、保守方針のリードをコミュニティと連携してサイクル化	• スケジューラ特性の理解（Rate Monotonic/Deadline等） • SMP/AMP構成 • フェイルセーフ設計 • 機能安全規格を意識したコーディング（MISRA C等）	• 新規ボードへのRTOS移植（BSP・HAL再構成） • **高信頼ドライバ**（ストレージ・通信）の設計 • タイミング保証設計（WCET計測、優先度反転対策） • パーティショニングによる機能分離 • 電源管理の最適化

組込みOSSエンジニアスキルマップ案：アプリケーションエンジニア

一般社団法人
組込みシステム技術協会
Japan Embedded System Technology Association

スキルレベル	Linux		習得すべきスキルセット（共通）	RTOS(従来型組込みOS)	
	技術要素（知識）	開発技術（できること）		技術要素（知識）	開発技術（できること）
エントリーレベル	• OSSライセンス/コンプライアンス/コミュニティ貢献の基本 • POSIX基礎（プロセス／スレッド、ファイルI/O） • IPC（パイプ／メッセージキュー／共有メモリ） • ソケット通信 • systemd/サービス管理 • パッケージ管理	• 正しいOSSの利活用と運用知識 • C/C++でのCLIアプリ実装 • **スレッド＋同期**による並行処理 • 基本ネットワーククライアント／サーバの作成 • gdb/valgrind/straceでの不具合解析 • ユニットテスト導入	• OSの基本機能、APIを含む開発手法を理解した基本的なアプリケーション設計と開発、実装、デバッグ、単体テスト知識の習得	• タスクモデル／イベント駆動 • メッセージキュー・イベントフラグ • 有限状態機械（FSM） • ドライバAPIの使い方	• RTOSアプリの基本構築（周期タスク／イベント処理） • **ドライバAPI経由**でのI/O制御 • タイミング計測（周期／ジッタ）と簡単な改善 • スタブ／モックでの単体テスト
ミドルレベル	• ミドルウェア活用（例：Qt/GTK、GStreamer、DB、MQTT/AMQP等） • メモリ管理／性能プロファイリング • ログ設計 • セキュアコーディング	• GUI/メディア/通信アプリの構築 • **デーモン化とサービス管理** • perf/heaptrack等でのボトルネック解析 • OTAアップデート連携（A/B、ロールバック設計） • Yoctoレイヤーへのレシピ追加	• OS各種周辺機器IF、ミドルウェアを活用した高度なアプリケーション設計、開発、実装（レシピ追加なども含む）、ターゲットボードを利用した実機テスト知識を習得 • システム全体の提供機能を考慮した性能設計とチューニングなどの知識を習得	• デターミニズム設計（静的割り当て、ヒープ非依存化） • 状態遷移設計（状態図／時系列図） • ネットワーク／ファイルシステムミドルウェアの利用	• **タイミング保証**（最悪実行時間の測定） • 優先度設計 • メモリ使用量の削減 • プロトコル実装（Modbus/CANなど）と再接続・リトライ設計 • HIL/実機自動テスト
アドバンスドレベル	• システムアーキテクチャ設計（プロセス分割 • IPC設計、スケーラビリティ） • リアルタイム制約下の設計（周期処理／遅延許容） • セキュリティ機能（権限分離、MAC、暗号）	• **高信頼アプリ**の設計（ウォッチドッグ、フェイルセーフ） • ゼロダウンタイム更新設計 • eBPF等による運用時観測 • 性能チューニングとメモリフットプリント最適化 • 障害注入テスト	• 高度かつ高信頼性、セキュアなアプリケーション開発とシステムアーキテクチャ設計、実装の知識を習得 • 高度な障害解析、デバッグ、対策などの知識を習得	• システム全体のリソース配分／タスク構成の最適化 • フェイルセーフ／デグレード運転 • セーフティ要求を満たす設計（例：ASIL準拠の設計指針の理解）	• **高信頼リアルタイムアプリ**の設計・性能保証 • パーティショニング／メモリ保護境界に配慮した実装 • 異常系（電源断／通信断）シナリオの包括的試験 • 現場ログからの再現性向上