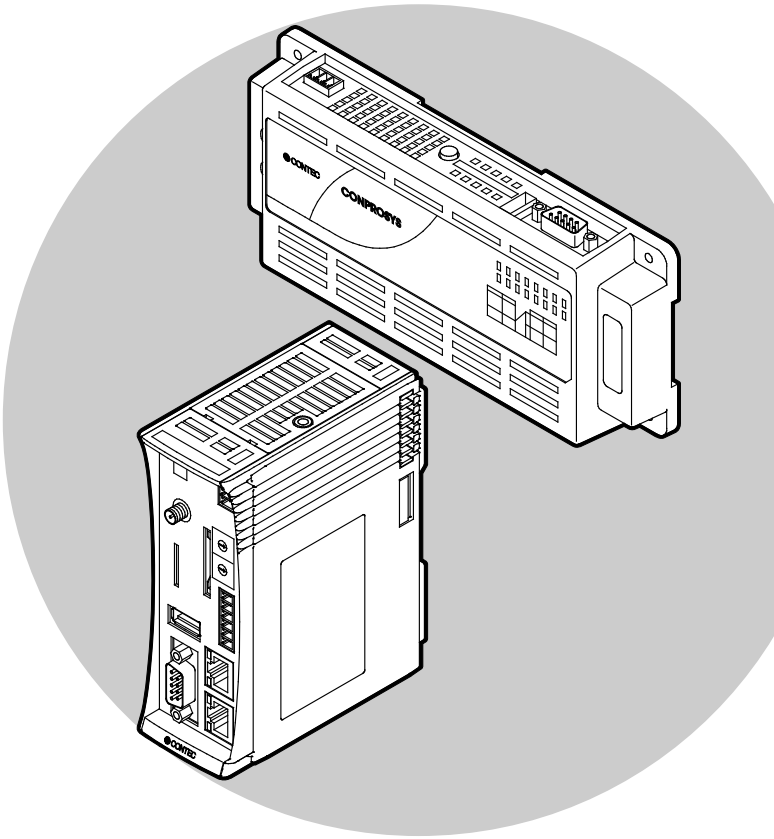
CONTEC

CONPROSYS®

ユーザーズマニュアル

(クロスビルド版)

CONPROSYS Linux SDK Ver. 2.2.0



目 次

株式会社コンテック

目次

目次

はじめに

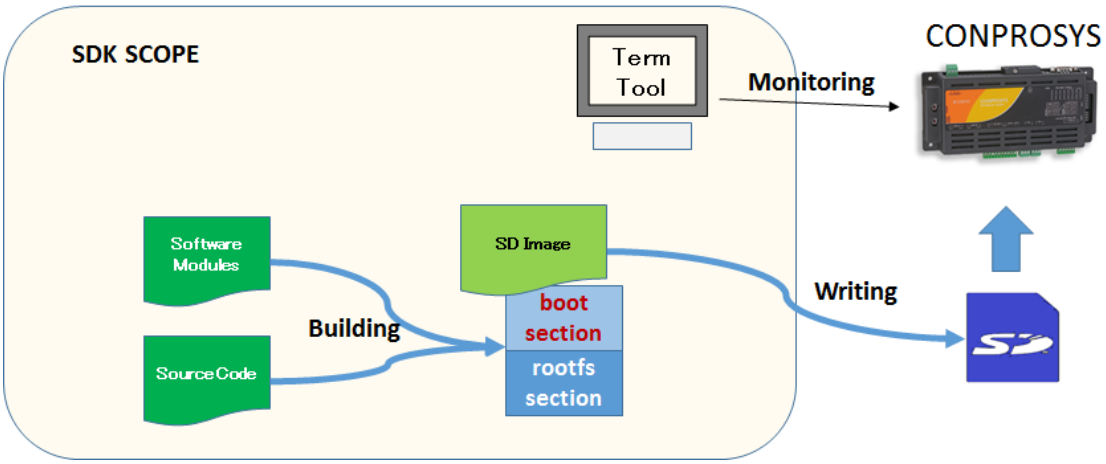
1. 概要

CONPROSYS Linux SDK(Software Development Kit)は、CONPROSYS上で動作するモジュールの生成するためのソフトウェア開発環境を目的とします。

本SDKは、次の範囲になります。

- CONPROSYSが動作するソフトウェアを生成するための開発用ホストPC上のツール
  (ソースコード(kernel, ライブラリ、ドライバ等)やビルドスクリプト等)
- CONPROSYSのソフトウェアをSDカードに書き込むための開発用ホストPC上のツール
- CONPROSYS上のソフトウェア動作をモニタするためのツール
  (シリアルコンソール等)

SDK範囲(SCOPE)



本SDKは、CONPROSYSのソフトウェアモジュールを開発用ホストPC上でクロスビルドによって生成します。
CONPROSYS上でセルフビルドを行う場合は、セルフビルド版のマニュアルを参照ください。
また、本SDKではセルフビルド版のCONPROSYS Linux SDKを生成することも可能です。詳細については、『**ビルド(P58)**』以降の章を参照ください。

2. 対応するCONPROSYS製品一覧

本SDKに対応する製品は次に示します。

【コンパクトタイプ　M2Mコントローラシリーズ】

CPS-MC341-ADSCxシリーズ	マルチI/Oモデル
CPS-MC341G-ADSC1シリーズ	マルチI/O + 3G(日本国内 / グローバル)モデル
CPS-MC341Q-ADSC1	マルチI/O + 920MHz帯通信モデル

【コンパクトタイプ　M2M Gatewayシリーズ】

CPS-MG341-ADSC1シリーズ	マルチI/Oモデル
CPS-MG341G-ADSC1シリーズ	マルチI/O + 3Gモデル
CPS-MG341G5-ADSC1	マルチI/O + LTEモデル

【スタックタイプ　M2Mコントローラシリーズ】

CPS-MCS341-DS1シリーズ	CPUモジュール
CPS-MCS341G-DS1	CPUモジュール + 3Gモデル
CPS-MCS341G5-DS1	CPUモジュール + LTEモデル
CPS-MCS341Q-DS1	CPUモジュール + 920MHz帯通信モデル

【スタックタイプ　M2M Gatewayシリーズ】

CPS-MGS341-DS1	CPUモジュール
CPS-MGS341G5-DS1	CPUモジュール + LTEモデル

※M2Mコントローラシリーズに搭載されているHMI、VTC、OPC-UA、Modbus等の機能は、
CONPROSYS Linux SDKに備わっておりません。別途、ソフトウェアの組み込みが必要です。
※M2M Gatewayシリーズに搭載されているHMI、VTC、OPC-UA、Modbus、PLC、CNC等の機能は、
CONPROSYS Linux SDKに備わっておりません。別途、ソフトウェアの組み込みが必要です。
※PACシステムシリーズ、nanoシリーズは対応しておりません。

3. ソフトウェア使用許諾契約書

本契約は、お客様と株式会社コンテック（以下「当社」といいます。）との間で、本製品に含まれるソフトウェアプログラム（以下「本ソフトウェア」といいます。）の使用許諾に関して合意するものです。本ソフトウェアを使用、又は本ソフトウェアをインストールした機器を使用することによって、お客様は本契約の各条項に同意されたものとさせていただきます。このお客様の同意をもって、本契約は成立し、効力を生じます。本契約に同意されない場合、本ソフトウェアの使用、又は本ソフトウェアをインストールした機器を使用することはできません。

第１条(知的財産権)

本ソフトウェア及びマニュアル等付属するドキュメント並びにその複製物（以下「本ソフトウェア等」といいます。）の著作権、特許権その他知的財産権は当社もしくは正当な権利者が所有するものであり、お客様には、本契約書において明示的に許諾されたものを除き、何らの権利も発生しません。

第2条(使用許諾)

1. 当社は、お客様に対し、本ソフトウェアに対応する当社ハードウェア製品を使用する目的で、本ソフトウェアをインストール及び使用する非独占的な権利を許諾します。
2. お客様は、緊急時のバックアップの目的でのみ、本ソフトウェアを使用する上で最低限必要な本数に限り、本ソフトウェアを複製することができます。但し、複製物には、当社が提供する、本ソフトウェアについての諸権利に関する表示を添付するものとします。
3. お客様は、当社がライブラリとして提供するソフトウェアをお客様の作成するソフトウェアに組み込むことができます。

第3条(利用の制限)

お客様は、次の各号に定める行為を行わないとします。

(1) 本契約に定める場合以外の本ソフトウェアから派生するソフトウェアの制作
(2) 本契約に定める場合以外の本ソフトウェアの複製
(3) 本ソフトウェアの改変、翻案、逆コンパイル、逆アセンブル、リバース・エンジニアリング
(4) 本ソフトウェア上の権限の表示や商標の削除又は変更

第4条(免責)

1. 当社は本ソフトウェアに関しいかなる保証もいたしません。
2. 本ソフトウェアをダウンロード、インストール、使用又は利用した結果、ハードウェア又はデータに支障が生じた場合等、本ソフトウェアに起因し又は関連して損害が発生した場合であっても、当社は一切責任を負いません。本ソフトウェアを複製し、組み込み又は改変したソフトウェア及びこれらを使用又は利用して作成されたソフトウェアについても同様とします。

第5条(譲渡)

1. お客様は、次の各号に定める条件を全て満たした場合に限り、本ソフトウェア及び本契約において許諾されたお客様の権利を第三者に対し、譲渡することができます。
   (1) 本契約書と共に本ソフトウェア等を全て当該第三者に譲渡すること
   (2) 本ソフトウェアがダウンロードされた当社のハードウェア製品の全てを当該第三者に譲渡すること
   (3) 譲渡を受ける方が本契約の条件に同意すること
2. 前項の規定によって本ソフトウェア及び権利の譲渡がなされた場合には、譲渡を受けた方は、譲渡を受けたときからこの契約に拘束されるものとします。

第6条（契約の解除）

1. お客様が本契約の各条項に従わなかった場合、当社は、お客様に対し、何らの通知・催告を行うことなく直ちに本契約を終了させることができます。
2. 本契約の終了と同時に、お客様に与えられていた使用許諾は全て失われます。直ちに本ソフトウェアの一切の使用を中止し、本ソフトウェアをアンインストールし、全ての複製物を破棄するものとします。

第7条（物理的欠陥について）

1. 本ソフトウェア等が格納されている記録媒体に本ソフトウェア等の使用に支障をきたす物理的欠陥があった場合、当社は、お客様が本ソフトウェア等をお受け取りになった日から30日以内にご購入いただいた販売店を通して記録媒体を交換するものとします。

第8条（ソフトウェアプログラムに関する情報）

1. 本ソフトウェアに関する各種情報やアップデートプログラムは、当社ウェブサイトで提供するものとします。
2. 前項の情報やアップデートプログラムは、本契約に基づきお客様に対して許諾されます。お客様は、必要に応じて独自の判断でこれらの情報やアップデートプログラムを、使用することができますが、その場合には、その情報やアップデートプログラムについても本契約の条項を遵守しなければなりません。

第9条（輸出規制）

1. 本ソフトウェア等を外国に持ち出す場合には、お客様は日本国外国為替及び外国貿易法、米国輸出管理法及びその他の国の法令を遵守しなければなりません。
2. お客様は、本ソフトウェア等を核兵器、生物化学兵器の設計、開発、製造若しくはミサイルの設計、開発、製造に使用するおそれがある個人又は法人に譲渡、輸出又は再輸出してはいけません。
3. 次の各号で定める国、地域、個人又は法人に、本ソフトウェア等を譲渡、輸出、再輸出してはいけません。
   (1) キューバ、イラン、イラク、リビア、北朝鮮
   (2) 輸出貿易管理令に基づく「外国ユーザーリスト」又は、米国商務省の「Denied Persons List」に記載されている個人又は法人
   (3) 日本国政府、米国政府、その他関係国の政府により指定された国、地域、個人又は法人

第10条（準拠法）

1. 本契約は日本国法に従い理解、解釈されるものとします。

第11条（管轄の合意）

1. 本契約ないし本ソフトウェアに関して紛争が生じ、訴訟提起等の法的手続きが必要となった場合には、大阪簡易裁判所ないし大阪地方裁判所をもって、第１審の専属的合意裁判所とします。

第12条（契約の分離）

1. 本契約の一部の条項が無効とされ又は法的強制力を失ったとしても、その他の条項には影響を与えることはなく、各条項は有効であり、法により許された範囲内で法的強制力を有するものとします。

安全にご使用いただくために

1. 注意記号の説明

本書では、人身事故や機器の破壊をさけるため、次のシンボルで安全に関する情報を提供しています。
内容をよく理解し、安全に機器を操作してください。

⚠危険	「死亡または重傷を負うことがあり、かつその切迫の度合いが高い内容」を示します。
⚠警告	「死亡または重傷を負うことが想定される内容」を示します。
⚠注意	「傷害を負うことが想定されるか、または物的損害の発生が想定される内容」を示します。

2. 取り扱い上の注意

⚠注意

- 本製品または本書は機能追加、品質向上のため予告なく仕様を変更する場合があります。継続的にご利用いただく場合でも、必ず当社Webサイトのマニュアルを読み、内容を確認してください。
- 本製品を改造しないでください。
  改造をしたものに対しては、当社は一切の責任を負いません。
- 本製品の運用を理由とする損失、逸失利益などの請求につきましては、前項にかかわらず、いかなる責任も負いかねますのであらかじめご了承ください。

3. セキュリティに関する注意

ネットワークに接続する際は、存在するセキュリティリスクを考慮の上、セキュリティ対策事例を参考に本体および関連するネットワーク機器を適切に設定してください。

1. セキュリティリスク

- 外部ネットワークからの不正侵入に伴うシステムの停止、データの破損、情報の窃取、マルウェア※1への感染。
- 侵入後にその機器を踏み台として、外部ネットワークへの攻撃。(被害者から加害者になる)
- 外部へのネットワーク接続に伴う意図しない情報漏洩。
- これら事故の二次被害として、風評被害、損害賠償負担、信用の失墜、機会損失等。

※1： マルウェア(Malicious Software)：悪意あるプログラム。ユーザーの望まない動作をするプログラム

2. セキュリティ対策事例

- 初期パスワードを変更する。(パスワード設定方法は、ご使用の製品の解説書/マニュアルを参照してください)
- パスワード強度の高いものを設定する。

  半角英字小文字、大文字、数字等を含み、類推されにくいパスワードを使用する

- 定期的にパスワードを変更する。
- 不要なネットワークサービスや、不要な機能を停止(無効化)する。
- ネットワーク接続機器において、ネットワークでのアクセス元を制限する。※2
- ネットワーク接続機器において、ネットワークの解放ポートを制限する。※2
- 専用ネットワークやVPN※3 など閉域網を使ってネットワークを構築する。

※2： 設定方法はネットワーク機器のメーカー各社へお問い合わせください。

※3： VPN(Virtual Private Network)：通信経路を認証や暗号化を用いて保護することにより、第三者が侵入することができない、安全なネットワークです。

不正アクセスの手段や抜け道(セキュリティホール)は、日夜新たに発見されており、それを防ぐ完璧な手段はありません。

インターネット接続には、常に危険が伴うことをご理解いただくとともに、常に新しい情報を入手し、セキュリティ対策を行うことを強くおすすめします。

開発環境

1. 開発に必要なもの

- 開発用ホストPC (Linux)
- SDHCカード (2Gbyte以上。SDXC非対応)
- シリアルモニタ用ケーブル (推奨ケーブル： FTDI製　TTL-232R-3V3-AJ)
- LAN Cable

2. SDKスペック

開発用ホストPC Linux Distribution:	Ubuntu 20.04 LTS (64bit版) Desktop 40Gbyte以上の空きHDD容量必要 sudoの実行出来る管理者権限ユーザー
ターゲットKernel version:	4.19.79
ターゲットDistribution:	arm版 Ubuntu 20.04 (SDブート用のみ)
クロスコンパイルGCC version:	gcc 9.3 (Hardware float)

必要なLinux toolchain :

gcc, g++, gcc-arm-linux-gnueabihf, g++-arm-linux-gnueabihf, device-tree-compiler, make, crossbuild-essential-armhf, binutils-arm-linux-gnueabihf, gawk, u-boot-tools, xinetd, kpartx, gperf, bison, flex, libncurses-dev, libssl-dev, openssh-server, samba, libncurses-dev, gawk

※上記は本SDKを動作させるための必要なものを示しています。
各々の開発環境で他に必要なパッケージがあれば、別途入手しインストールしてください。
(例：git, wget, subversion等)

3. SDK内容

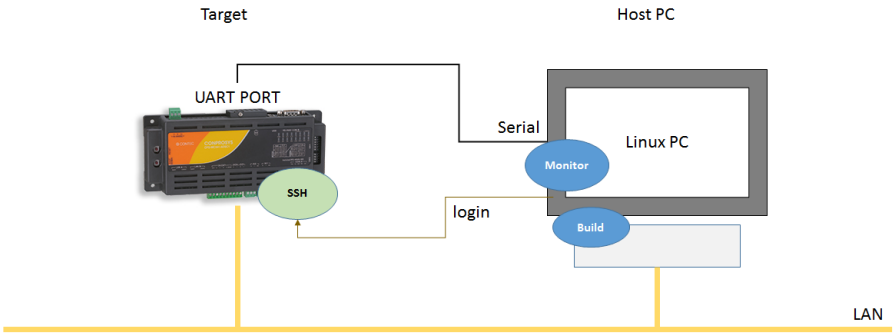
- SDKドキュメント
- ビルドツール
- ソースコード
  u-boot, kernel, サンプルアプリケーション, サンプルライブラリ, サンプルドライバ
- CONPROSYS製品毎のベースモジュール(u-boot, kernel, 設定等)

4. 開発環境構成

ホストPC（ビルド、モニター用）とターゲットの構成例を次に示します。

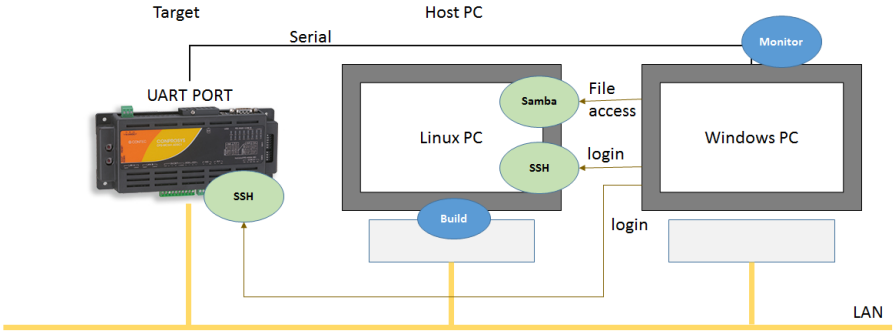
例1) 開発ホストPC１台でビルドとターゲットにシリアルモニタを使用する場合

Linux PC１台でビルドおよびシリアルモニタリングする場合



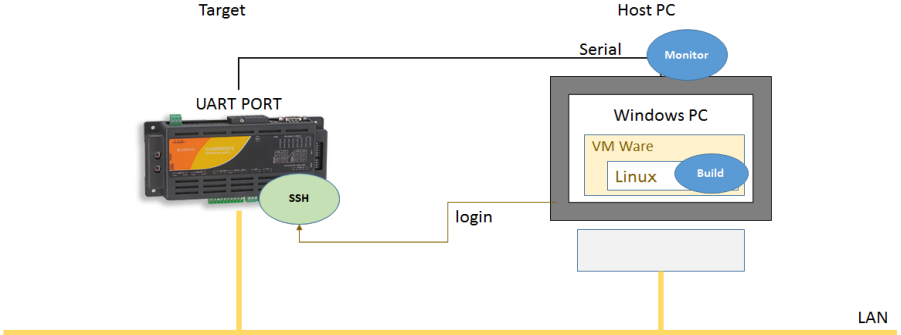
例2) 開発ホストPCでビルド(またはソースコード編集)、ターゲットにシリアルモニタを別のWindows PCの２台で使用する場合

ビルド用にLinux PC１台、シリアルモニタリング用にWindows PCを使う場合



例3) Windows OS等に仮想OSシステム(VM Ware, Virtual BOX等)を入れて、その上でLinux OSをインストールし開発ホストPCとして使用する場合

１台のPCでビルド用にLinux(VM Ware利用)、シリアルモニタリング用にWindowsを使う場合



5. SDKのインストール

ダウンロード版を次のように準備します。

1 ダウンロードした.isoファイルをマウントします。

マウント先ディレクトリは任意で作成してください。

```
sudo -E mount -o loop CPSSDK_xxxx.iso マウント先ディレクトリ
```

2 マウント先のディレクトリに移ります。

1. SDKに必要なツールチェーンのインストール

ubuntu のOS上で次のツールチェーンをaptコマンドによってインストールします。

ツールチェーンをインストールするにはインターネットに接続可能な環境が必要です。

gcc, g++, gcc-arm-linux-gnueabihf, g++-arm-linux-gnueabihf, device-tree-compiler, make, crossbuild-essential-armhf, binutils-arm-linux-gnueabihf, gawk, u-boot-tools, xinetd, kpartx, gperf, bison, flex, libncurses-dev, libssl-dev, openssh-server, samba, libncurses-dev, gawk

ツールチェーンをインストールする前に、aptのパッケージリストを更新してください。

aptのパッケージリスト更新コマンド：

```
sudo apt update
```

インストールコマンド：

```
sudo apt install gcc g++ gcc-arm-linux-gnueabihf g++-arm-linux-gnueabihf device-tree-compiler make crossbuild-essential-armhf binutils-arm-linux-gnueabihf gawk u-boot-tools xinetd kpartx gperf bison flex libncurses-dev libssl-dev openssh-server samba libncurses-dev gawk
```

2. CONPROSYS linux SDKのインストール

次のコマンドでSDKのインストールを開始します。

コマンド：

```
./install_sdk.sh [-C インストール先ディレクトリ]
```

オプション：

-C インストール先ディレクトリ

指定されるインストール先ディレクトリを生成し、そのディレクトリ下にインストールします。

※インストール先ディレクトリが指定されない場合は、カレントディレクトリ下に”CPS_SDK”というディレクトリを自動生成され、そのディレクトリにインストールされます。isoファイルをマウントした場合やDVDメディアの場合、カレントディレクトリ下にディレクトリは生成出来ませんので、必ずインストール先ディレクトリを指定してください。

※インストール先ディレクトリは、ログインユーザーのhomeディレクトリ下を推奨します。

コマンド例：

```
./install_sdk.sh -C ~/CPS_SDK
```

インストール後のディレクトリ構成は次のようになります。

インストール後のディレクトリ構成

```
~/
 |
 +-- [CPS_SDK]
        |
        +-- [Document]     ドキュメント
        |
        +-- [application]  アプリケーションソースディレクトリ
        |
        +-- [base]         デフォルトファイル格納ディレクトリ
        |
        +-- [kernel]       kernelソースディレクトリ
        |
        +-- [driver]       ドライバソースディレクトリ
        |
        +-- [lib]          ライブラリソースディレクトリ
        |
        +-- [ramdisk]      ramdisk(NOR Flash用)作成ディレクトリ
        |
        +-- [rootfs]       rootfsソースディレクトリ
        |
        +-- [tools]        ツール群
        |
        +-- [u-boot]       u-bootソースディレクトリ
        |
        +-- [target]       Targetディレクトリ
```

[Document]

SDKのドキュメントファイルを格納しているディレクトリです。

[application]

アプリケーションのソースコードを格納しているディレクトリです。

[base]

targetのベースとなるbootセクションとrootfsセクションを格納しているディレクトリです。

[tools]

SDKのツール群です。

[kernel]

kernelのソースコードを格納しているディレクトリです。

[driver]

ドライバのソースコードを格納しているディレクトリです。

[lib]

ライブラリのソースコードを格納しているディレクトリです。

[ramdisk]

NOR Flashにインストールするrootfsをパッケージしたramdiskを生成するディレクトリです。

[rootfs]

Ubuntu20.04 / light版 rootfsのベース差分を格納しているディレクトリです。

[u-boot]

u-bootのソースコードを格納するディレクトリです。

[target]

CONPROSYSの製品毎にSD起動させるためのセクションを生成するディレクトリです。configure.sh実行後、ターゲット用のディレクトリを生成し、ビルドしたモジュール(boot, kernel, driver, application)の格納先になります。

クロスビルド環境設定

1. SDカード作成フロー

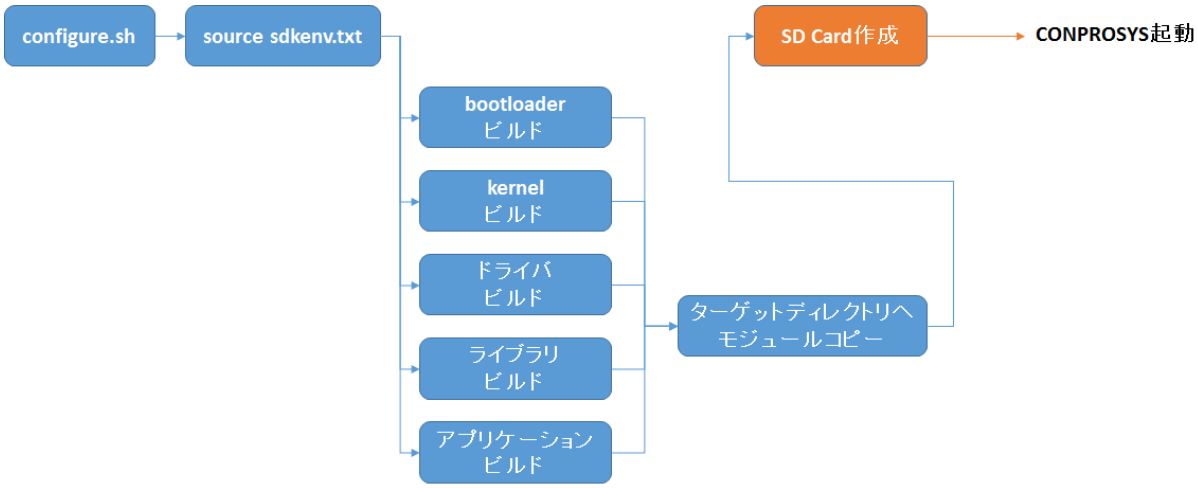
モジュールの生成から起動SDカード作成までのフローは下記のようになります。

◆ セルフビルド版CONPROSYS Linux SDK等、本SDKでビルドを行わないターゲットの場合

セルフビルド版CONPROSYS Linux SDKのSDカード作成フロー

◆ 本SDKでクロスビルドを行うターゲットの場合

クロスビルドによるSDカード作成フロー



SDカードの作成については『**SD起動用SDカードの作成（P31)**』を参照ください。

各ビルドについては『**ビルド（P58)**』を参照ください。

2. 初期設定

SDKインストール先ディレクトリ下で./configure.shを実行しビルドを行うための初期設定を行います。このコマンドを実行することによって次の環境ファイルとディレクトリを生成します。このコマンドは新規でターゲット向けのモジュールを生成する場合に使用します。

- ビルドするための環境設定ファイル(sdkenv.txt)
- kernelの.configファイル
- targetディレクトリ下に機器に応じたSDカードに書き込むためのファイルシステム
  (“boot”セクション, “rootfs”セクション)

コマンド：

```
./configure.sh
```

※途中root権限のコマンドを実行するために、パスワードを要求されることがあります。
パスワードを入力して処理の実行を進めてください。

コマンドを実行すると、ターゲットとなるCONPROSYSの情報を入力するモードになります。メニューに従って該当する番号を入力してください。

rootfsタイプ：

ターゲットのrootfsタイプを番号で入力します。SDK付にはセルフコンパイラが付属しており、CONPROSYS上でソフトウェアの開発が行えます。

1) light (busybox)	軽量版rootfs
3) Ubuntu 20.04 (include SDK)	Ubuntu 20.04 SDK付(セルフビルド版CONPROSYS Linux SDK)

搭載ツールの選択：

軽量版rootfsのrootfsタイプを選択した場合、搭載ツールを選択できます。搭載したいツールのタイプを番号で入力してください。

1) Wireless tools, Apache 2.4, PHP5	Wirelessツール, Apache 2.4, PHP5
0) None	搭載ツールなし

configure.shの実行後、targetディレクトリ下にtarget用のディレクトリとベースとなるbootセクションと./configureで選択したrootfsセクションのディレクトリ/ファイルを生成します。次の構成図は、Ubuntu20.04 with SDKを指定した場合のディレクトリ例です。

target以下のディレクトリ構成

```
~/
   |
   +-- [CPS_SDK]
          |
          +-- [target]                          Targetディレクトリ
                 +-- [conprosys]
                        +-- [boot]                  bootセクション
                        |       +-- MLO
                        |       +-- u-boot.img
                        |       +-- MC34*B**_1lan.dtb
                        |       +-- MC34*B**_2lan.dtb
                        |       +-- uImage.
                        +-- [Ubuntu20.04_dev]       rootfsセクション（SDカード用）
                                +-- [bin]
                                +-- [boot]
                                +-- [dev]
                                +-- [etc]
                                +-- [home]
                                +-- [lib]
                                +-- [media]
                                +-- [mnt]
                                +-- [opt]
                                +-- [proc]
                                +-- [root]
                                +-- [sbin]
                                +-- [selinux]
                                +-- [srv]
                                +-- [sys]
                                +-- [tmp]
                                +-- [usr]
                                +-- [var]
```

3. 環境設定

アプリケーションやカーネル等をビルドする前にSDKインストール先ディレクトリ下で./configure.shによって生成された sdkenv.txtでビルドするための環境変数を設定してください。

コマンド：

```
. sdkenv.txt
```

この環境変数の設定が行われていなければ、この項以降に説明するビルドやFirmware書き込み方法等の操作が正常に行えないことがありますので注意ください。

ターゲットのFirmware書き込み方法

1. システム起動について

ターゲットの起動には、次の方法があります。

◆ SDカードからシステムを起動

SDからのシステム起動について、SD起動用のFirmwareをSDカードに書き込み、そのSDカードを挿入しシステムを起動させます。

◆ 内蔵NOR FLASHからシステムを起動

内蔵NOR FLASHからのシステム起動について、下記の２ステップを行うことで内蔵NOR FLASHからの起動が可能となります。

1 内蔵NOR FLASHへシステムをインストールためのインストールSDカードの作成

2 インストールSDカードを挿入しSDカードより起動させ、内蔵NOR FLASHへシステムをインストール

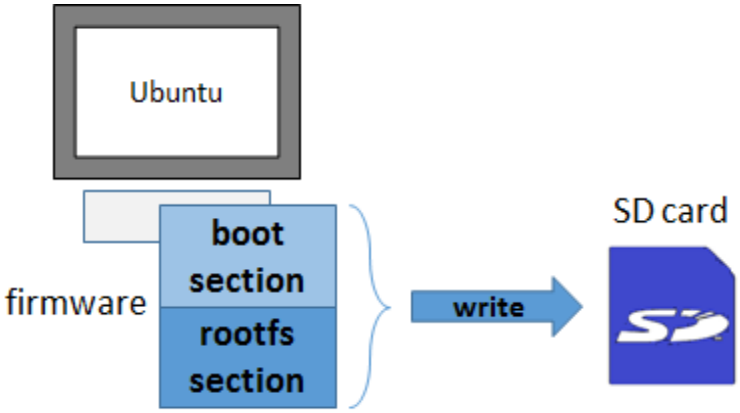
※内蔵NOR FLASHへシステムをインストールする際にNOR FLASH内のデータを消去します。
CONPROSYS工場出荷状態でインストールされているHMI/VTCの入ったソフトウェアも消去されますのでご注意ください。

各々の起動方法に応じたFirmwareの書き込み方法を次に示します。

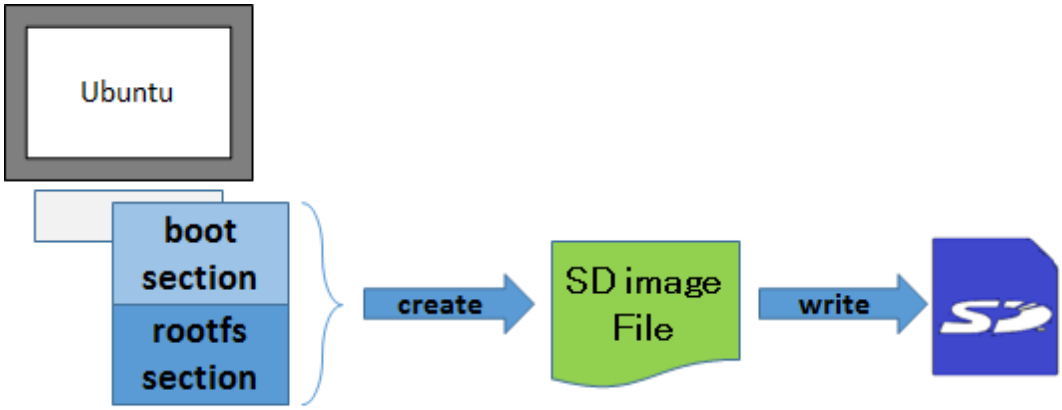
2. SD起動用SDカードの作成

SD起動用のSDカードは次の方法で作成することができます。

1) SDカードへ直接書き込み



2) SDイメージファイルを作成しイメージ書き込みソフトでSDカードへ書き込み



1. SDカードへ直接書き込み

1 ホストPCにSDカードを挿入し認識させます。

SDカードがどのファイルシステムで認識しているか、partedコマンド等で調べることができます。

例)

```
sudo parted -l
```

SDカードが自動マウントされている場合は、アンマウントしてください。

2 SDカードのパーティションの生成を行います。

例) SDカードのデバイスが /dev/sdbの場合

```
sudo -E ${CPS_SDK_ROOTDIR}/tools/mk2PartSDCard.sh /dev/sdb
```

このコマンドで次の２つのパーティションが生成されます。

- Bootパーティション　W95 FAT32 (LBA)
- rootfsパーティション　ext4

3 SDカードをマウントします。

上記で生成したパーティションをマウントする先のディレクトリを予め生成しておきます。

下記は、/mediaの下にboot用とrootfs用のディレクトリを生成するコマンド例です。

```
sudo mkdir /media/boot
sudo mkdir /media/rootfs
```

これらのマウント先に上記で生成したSDカードのパーティションをマウントします。

下記はSDカードが/dev/sdbの場合のコマンド例です。

```
sudo mount /dev/sdb1 /media/boot
sudo mount /dev/sdb2 /media/rootfs
```

4 targetの下に作成したbootディレクトリとrootfsディレクトリのファイルをSDカードにコピーします。

[bootパーティション（fat32)]

```
sudo cp -p ${CPS_SDK_INSTALL_FULLDIR}/boot/* /media/boot
```

[rootfsパーティション（ext4)]

```
sudo -E cp -rp ${CPS_SDK_INSTALL_FULLDIR}/${CPS_SDK_ROOTFS}/* /media/rootfs
```

5 SDカードにコピーしたファイルに対し同期を取ります。

```
sync
sync
sync
```

syncコマンドで同期を取る前にSDカードをアンマウントし、SDカードを抜き取った場合、SDカードにファイルが正しく書けていないことがあります。そのようなことを防止するためにsyncコマンドを実行しておいてください。

6 SDカードをアンマウントし、SDカードを抜き取ります。

```
sudo umount /media/boot
sudo umount /media/rootfs
```

2. SDイメージファイルを作成しイメージ書き込みソフトでSDカードへ書き込み

1 SDイメージファイルを作成します。

次のコマンドでイメージファイルを作成することができます。

コマンド：

```
sudo -E ${CPS_SDK_ROOTDIR}/tools/mk2SDCardImage.sh ${CPS_SDK_ROOTFS} [-f filename] [-s size]
```

オプション：

-f filename

出力するイメージファイル名を指定できます。指定しない場合は SD.img に出力されます。

-s size

出力するイメージファイルサイズを指定できます。指定しない場合は、2000Mbyteで出力されます。

コマンド例： ファイル名をターゲット名_rootfs.imgとし、サイズを4000Mbyteにしたい場合

```
sudo -E ${CPS_SDK_ROOTDIR}/tools/mk2SDCardImage.sh ${CPS_SDK_ROOTFS} ¥
-f cps_${CPS_SDK_ROOTFS}.img -s 4000
```

2 イメージファイルをSDカードに書き込みます。

[Windows版]

SDカードへの書き込みをWin32 Disk Imagerを使った例で示します。

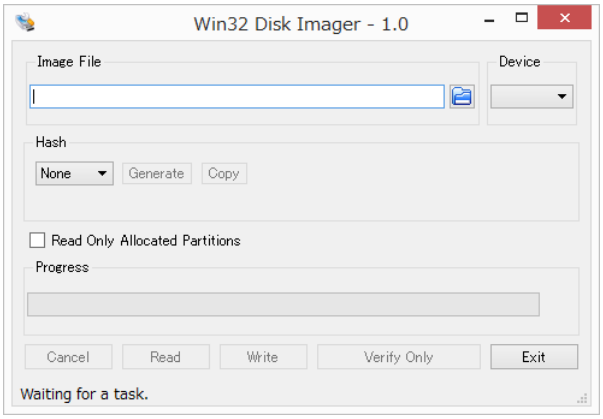
予め下記のサイトよりWin32 Disk Imager のインストーラをWindows PCにダウンロードし、インストールしてください。

https://sourceforge.net/projects/win32diskimager/

i) Windows PCにSDカードを挿入します。

ii) Win32 Disk Imagerを起動します。

Win32 Disk Imagerアプリケーション



iii) 書き込むimageファイルを選択します。
Device欄のドライブが書き込み先のSDカードになっているか確認し、Writeボタンを押して書き込みを開始します。

iv) 書き込みが終了すると通知のポップアップが表示されるのでOKボタンを押し、SDカードを抜いてください。

[Linux版]

i) SDカードがマウントされている場合、アンマウントします。

```
sudo umount /dev/sdb
```

ii) ddコマンドでSDカードへイメージファイルを書き込みます。

```
sudo dd if=イメージファイル名 of=/dev/sdb bs=1M
```

iii) syncコマンドで同期します。

```
sync
```

iv) コマンドが終了したら、SDカードを抜いてください。

3. 内蔵NOR FLASH起動用のインストールSDカードの作成

内蔵NOR FLASH起動用のインストールSDカードを作成する前に下記の準備が必要です。

- 『**初期設定（P26)**』でrootfsタイプの軽量版rootfsを選択した環境構築
- 内蔵NOR FLASH起動用インストールするためのrootfsセクションを作成
- カスタマイズしたbootloaderやアプリケーションなどのビルドとインストール用rootfsセクションへのコピー

1. 内蔵NOR FLASH起動用インストールするためのrootfsセクションを作成

ベースとなる内蔵NOR FLASHインストール用のrootfs (InstallerForFlash)を作成します。

コマンド：

```
./create_FlashInstaller.sh
```

このコマンド実行後にInstallerForFlashというディレクトリがターゲット向けのディレクトリ下に生成されます。ディレクトリ例を次に示します。

NOR Flashインストール用のrootfsが生成された時のディレクトリ構成

```
~/
 |
 +-- [CPS_SDK]
       |
       +-- [target]                          Targetディレクトリ
             +-- [conprosys]
                   +-- [boot]                bootセクション
                   +-- [Ubuntu20.04_dev]       rootfsセクション（SDカード用）
                   +-- [InstallerForFlash]   rootfsセクション（NOR Flashインストール用）
                         |
                         +-- [InstallToMTD] NOR Flashインストールファイルディレクトリ
                         |         +-- MLO.byteswap            NOR Flash版ブートローダ
                         |         +-- u-boot.img              NOR Flash版u-boot
                         |         +-- MC34*B*0_1lan.dtb       1LAN用DeviceTreeファイル
                         |         +-- MC34*B*0_2lan.dtb       2LAN用DeviceTreeファイル
                         |         +-- ramdisk.xz              ramdisk
                         |         +-- uImage                  kernel
                         |         +-- mtd5.tgz                アプリケーションデータ
                         +-- [bin]
                         +-- [dev]
                         +-- [etc]
                         +-- [home]
                         +-- [lib]
                         +-- [libexec]
                         +-- linuxrc
                         +-- [mnt]
                         +-- [proc]
                         +-- [sbin]
                         +-- [sys]
                         +-- [tmp]
                         +-- [usr]
                         +-- [var]
```

アプリケーションデータ mtd5.tgz

```
    [mtd5.tgz]
       |
       +-- [etc]
       |      +-- passwd
       |      +-- group
       |      +-- [conprosys]
       |             +-- config.ini
       |             +-- CPS_SDK_VER
       +-- [opt]
       +-- startup.sh
```

2. 内蔵NOR FLASHインストール用rootfsセクションへのコピー

カスタマイズビルドしたbootloaderやアプリケーションソフト等があれば、インストール用rootfs(InstallerForFlash)のインストールディレクトリ(/InstallToMTD)へコピーします。
コピー前にビルドによって生成されたファイルがあるか確認の上、コピーをしてください。

[bootloader]
『**内蔵NOR FLASH起動用のビルド（P60)**』の項でビルドしたMLO.byteswap u-boot.imgをコピーします。
ビルド方法については『**内蔵NOR FLASH起動用のビルド（P60)**』の項を参照ください。
コマンド：

```
cd ${CPS_SDK_ROOTDIR}/u-boot
cp -p MLO.byteswap u-boot.img ${CPS_SDK_INSTALL_FULLDIR}/InstallerForFlash/InstallToMTD
```

[kernel]
コマンド：

```
cd ${CPS_SDK_ROOTDIR}/kernel/arch/arm/boot/
cp -p uImage ${CPS_SDK_INSTALL_FULLDIR}/InstallerForFlash/InstallToMTD/uImage
cp -p dts/MC34?B*_?lan.dtb ${CPS_SDK_INSTALL_FULLDIR}/InstallerForFlash/InstallToMTD/
```

『**ターゲットのkernelのビルド（P61)**』の項でビルドしたuImage, dtbファイルをコピーします。
ビルド方法については『**ターゲットのkernelのビルド（P61)**』の項を参照ください。

[ramdisk]
コマンド：

```
cd ${CPS_SDK_ROOTDIR}/ramdisk
make install
```

『**内蔵NOR FLASH起動用ramdisk.xzのビルド（P66)**』の項でビルドしたramdiskファイルをコピーします。
ビルド方法については『**内蔵NOR FLASH起動用ramdisk.xzのビルド（P66)**』の項を参照ください。

本SDKに含まれるインストールプログラム(shell script)は、6つのファイル(MLO.byteswap, u-boot.img, MC34xBxx_1lan.dtb, MC34xBxx_2lan.dtb, uImage, ramdisk.xz)と機器専用のアプリケーションデータ(mtd5.tgz)をNOR FLASHへインストールを行います。

${CPS_SDK_INSTALL_FULLDIR}/InstallerForFlash/home/flashwriter.sh

この内容を参考にインストール方法をカスタマイズしてください。
NOR FLASHの容量等についての詳細は『**内蔵NOR FLASHメモリマップ（P86)**』を参照ください。

3. 内蔵NOR FLASHインストール用SDカードの作成 (SDカードへ直接書き込み)

内蔵NOR FLASHインストール用のSDカードは次の手順で作成します。

基本的な手順はSD起動用のものと同じですが、rootfsのコピー元が違います。(こちらのrootfsのコピー元はInstallerForFlashになります)

1 ホストPCにSDカードを挿入し認識させます。

SDカードがどのファイルシステムで認識しているか、partedコマンド等で調べることができます。

例)

```
sudo parted -l
```

2 SDカードのパーティションの生成を行います。

例) SDカードのデバイスが /dev/sdbの場合

```
sudo -E ${CPS_SDK_ROOTDIR}/tools/mk2PartSDCard.sh /dev/sdb
```

このコマンドで次の２つのパーティションが生成されます。

- Bootパーティション　W95 FAT32 (LBA)
- rootfsパーティション　ext4

3 SDカードをマウントします。

上記で生成したパーティションをマウントする先のディレクトリを予め生成しておきます。

下記は、/mediaの下にboot用とrootfs用のディレクトリを生成するコマンド例です。

```
sudo mkdir /media/boot
sudo mkdir /media/rootfs
```

これらのマウント先に上記で生成したSDカードのパーティションをマウントします。

下記はSDカードが/dev/sdbの場合のコマンド例です。

```
sudo mount /dev/sdb1 /media/boot
sudo mount /dev/sdb2 /media/rootfs
```

4 targetの下に作成したbootディレクトリとrootfsディレクトリのファイルをSDカードにコピーします。

[bootパーティション（fat32)]

```
sudo -E cp -p ${CPS_SDK_INSTALL_FULLDIR}/boot/* /media/boot
```

[rootfsパーティション（ext4)]

```
sudo -E cp -rp ${CPS_SDK_INSTALL_FULLDIR}/InstallerForFlash/* /media/rootfs
```

5 SDカードにコピーしたファイルに対し同期を取ります。

```
sync
sync
sync
```

syncコマンドで同期を取る前にSDカードをアンマウントし、SDカードを抜き取った場合、SDカードにファイルが正しく書けていないことがあります。そのようなことを防止するためにsyncコマンドを実行しておいてください。

6 SDカードをアンマウントし、SDカードを抜き取ります。

```
sudo umount /media/boot
sudo umount /media/rootfs
```

4. 内蔵NOR FLASHインストール用SDカードの作成 (SDイメージファイル作成)

1 SDイメージファイルを作成します。

次のコマンドでイメージファイルを作成することができます。

コマンド：

```
sudo -E ${CPS_SDK_ROOTDIR}/tools/mk2SDCardImage.sh InstallerForFlash [-f filename] [-s size]
```

オプション：

-f filename

出力するイメージファイル名を指定できます。指定しない場合は SD.img に出力されます。

-s size

出力するイメージファイルサイズを指定できます。指定しない場合は、2000Mbyteで出力されます。

コマンド例： ファイル名を”ターゲット名_InstallerForFlash.img”とする場合

```
sudo -E ${CPS_SDK_ROOTDIR}/tools/mk2SDCardImage.sh InstallerForFlash ¥
-f InstallerForFlash.img -s 256
```

2 イメージファイルをSDカードに書き込みます。

SDカードへイメージファイルの書き込む方法は、『**SDイメージファイルを作成しイメージ書き込みソフトでSDカードへ書き込み (P34)**』をご参照ください。

4. 内蔵NOR FLASHへのインストール

『**内蔵NOR FLASHインストール用SDカードの作成(SDカードへ直接書き込み) (P39)**』または『**内蔵NOR FLASHインストール用SDカードの作成(SDイメージファイル作成) (P41)**』で作成したSDカードをCONPROSYS本体のSD boot有効にして起動すると、自動的にNOR FLASHへの書き込みが開始します。

起動方法については『**SDカードからの起動 (P44)**』の項を参照ください。

インストール中はST1(緑)、ST2(赤)とPower(緑)のLEDが点滅し、正常に終了するとST1(緑)とPower(緑)のLEDが点灯したままになります。

ターゲット動作確認

1. ターゲット起動方法

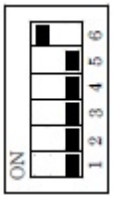
1. SDカードからの起動

各々のCONPROSYS本体のDIP SWがSD bootモードが有効になっていることを確認してください。

◆ コンパクトタイプ(CPS-Mx341-xxx)の場合 (Gatewayシリーズも含む)

DIP SW1の6番PINがON (SD bootモード有効)

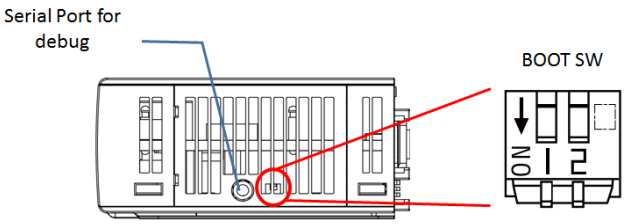
コンパクトタイプBOOT SW設定



◆ スタックタイプ(CPS-MxS341-xxx)の場合

デバッグ用シリアルポート(3.5Φミニジャック)隣のBOOT SW(ケース中)の2番PINがON
(SD bootモード有効)

スタックタイプBOOT SW設定



『**SD起動用SDカードの作成(P31)**』で作成したSDカードを挿し、本体に電源を入れてください。
※SDカードが挿入されていない場合、内蔵NOR FLASHより起動します。

2. 内蔵NOR FLASHから起動

『**SDカードからの起動(P44)**』に記載しているSD bootモードが無効になっていることを確認し、本体に電源を入れてください。

2. シリアルケーブル接続によるログイン

ホストPCからCONPROSYSへ専用シリアルポート(3.5Φミニジャック)にシリアルケーブルを接続することで、コンソールよりCONPROSYSへログインすることができます。
シリアルの設定は次のようになります。

Baud rate:	115200 bps
Data bit:	8 bit
Parity:	none
Stop bit:	1 bit
Hardware flow:	none

ホストPCとCONPROSYSを接続するシリアルケーブルは次のものを推奨しています。
ドライバソフトはシリアルモニタを行うPCのOSに合わせてダウンロードしてください。

- FTDI製　TTL-232R-3V3-AJ
  ドライバ提供URL:　http://www.ftdichip.com/Drivers/VCP.htm

デフォルトのログイン/パスワードは次のようになります。

ログイン:	conprosys
パスワード:	contec

※外部ネットワークに接続出来る環境である場合、パスワードは『**セキュリティに関する注意(P13)**』に従い、必ず変更するようにしてください。

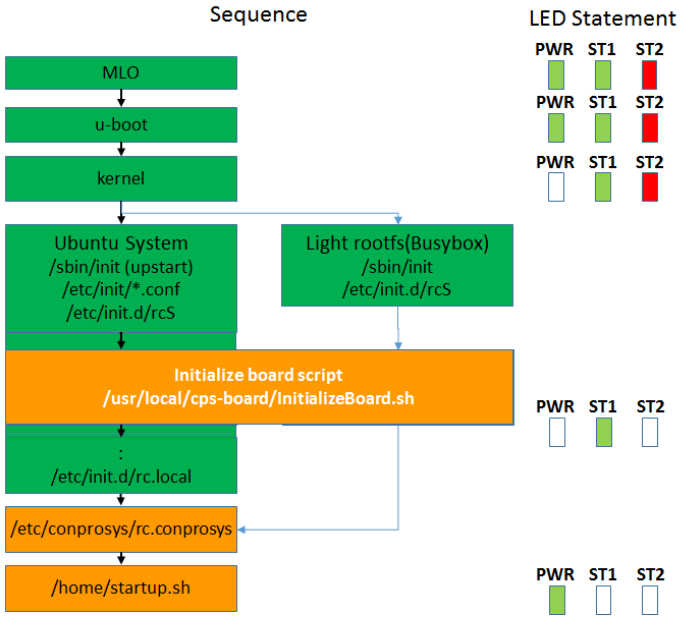
3. ssh接続によるログイン

ホストPCからCONPROSYSが同じネットワーク上で動作している場合、sshを使ってCONPROSYSへログインすることができます。

4. ターゲットの起動シーケンス

本SDKのターゲット上での起動は次の順序で実行されます。

起動シーケンス



ターゲット起動時に実行させたいコマンドは次のスクリプトファイルを編集してください。

[SDカードブートの場合]

/home/startup.sh

[内蔵NOR FLASHブートの場合]

/mnt/mtd/startup.sh

ターゲット上では、ファイルシステムがREAD ONLYモードにされているため、これらのファイルを編集するにはWRITE可能モードにしてから編集し、編集終了後はREAD ONLYモードに戻してください。

[SDカードブートの場合]

WRITE可能モード

sudo rommode rw

READ ONLYモード

sudo rommode ro

[内蔵NOR FLASHブートの場合]

WRITE可能モード

sudo rommode_mtd rw

READ ONLYモード

sudo rommode_mtd ro

5. ターゲットのネットワーク設定

本SDKのターゲットのデフォルトネットワーク設定は次のように設定されています。

[デフォルト設定]

LAN A (eth0):　10.1.1.101

LAN B (eth1):　DHCP　(2 LAN Type設定時のみ)

ネットワーク設定を変更したい場合は、各々のrootfsに合わせて編集してください。

ネットワーク設定を変更したい場合はターゲット上の次のファイルをroot権限で編集してください。

/etc/conprosys/config.ini

LAN設定

項目名	設定内容
eth0_dhcp	LAN A(eth0)のDHCP有効/無効を設定します。 有効: enabled 無効: disabled
eth0_ipaddr	LAN A(eth0)のIPアドレスを設定します。
eth0_netmask	LAN A(eth0)のNetmaskを設定します。
eth0_gateway	LAN A(eth0)のゲートウェイアドレスを設定します。
eth0_dns1	LAN A(eth0)のDNSサーバーアドレスを設定します。
eth1_dhcp	LAN B(eth1)のDHCP有効/無効を設定します。 有効: enabled 無効: disabled
eth1_ipaddr	LAN B(eth1)のIPアドレスを設定します。
eth1_netmask	LAN B(eth1)のNetmaskを設定します。
eth1_gateway	LAN B(eth1)のゲートウェイアドレスを設定します。
eth1_dns1	LAN B(eth1)のDNSサーバーアドレスを設定します。
ntp_addr	NTPサーバーを設定します。
host_name	ホスト名を設定します。 初期状態においては項目がないため下記ホスト名に設定されます。 モデル名＋MACアドレスの下位3バイト

3G/LTE設定（3G/LTEモデルのみ)

項目名	設定内容
m3g_connect	3G/LTE接続の有効/無効を設定します。 有効：enabled 無効：disabled
m3g_apn	通信サービス業者から情報提供されるAPNを設定します。
m3g_user	通信サービス業者から情報提供されるユーザーIDを設定します。
m3g_passwd	通信サービス業者から情報提供されるパスワードを設定します。
m3g_auth	通信サービス業者から情報提供される下記の認証タイプを設定します。 None PAP CHAP

無線LAN設定

項目名	設定内容
wlan_dhcp	無線LAN(wlan0)のDHCP有効/無効を設定します。 有効：enabled 無効：disabled
wlan_ipaddr	無線LAN(wlan0)のIPアドレスを設定します。
wlan_netmask	無線LAN(wlan0)のNetmaskを設定します。
wlan_gateway	無線LAN(wlan0)のゲートウェイアドレスを設定します。
wlan_dns1	無線LAN(wlan0)のDNSサーバーアドレスを設定します。
wlan_essid	無線LAN(wlan0)のSSIDを設定します。
wlan_encrypt	無線LAN(wlan0)の暗号化方式を下記の中から選び設定します。 [設定項目] 暗号化なし: none WEP: wep WPA-PSK AES: wpapsk-aes WPA-PSK TKIP: wpapsk-tkip WPA2-PSK AES: wpa2psk-aes WPA2-PSK TKIP: wpa2psk-tkip WPA/WPA2-PSK自動: wpawpa2psk-auto
wlan_key	無線LAN(wlan0)の暗号化キーを設定します。

※無線LANはUSB搭載モデルに対応しているUSB無線LANアダプターを接続することで使用可能になります。

サービス起動設定

項目名	設定内容
srv_ssh	SSHサーバーの起動を設定します。 enabled: 有効 disabled: 無効
srv_ftp	FTPサーバーの起動を設定します。 enabled: 有効 disabled: 無効
srv_samba	SAMBAサーバーの起動を設定します。 enabled: 有効 disabled: 無効

※軽量版rootfsの場合、sambaサーバーは使用できません。

ルーター機能設定

項目名	設定内容
router	ルーター機能を設定します。 enabled: 有効 disabled: 無効
wan_if	WANインターフェイスを設定します。 3G: eth2 LTE: ppp0 Wireless LAN: wlan0 LAN A: eth0 LAN B: eth1
dhcp_server	DHCPサーバーの起動を設定します。 enabled: 有効 disabled: 無効
dhcp_server_lan_if	DHCPサーバーのLANインターフェイスを設定します。 Wireless LAN: wlan0 LAN A: eth0 LAN B: eth1
dhcp_server_top_addr	DHCP先頭アドレスを設定します。
dhcp_server_alloc_num	DHCPアドレス割り当て数を設定します。

PPPoE機能設定

項目名	設定内容
pppoe	PPPoE機能を設定します。 enabled: 有効 disabled: 無効
pppoe_if	PPPoEインターフェイスを設定します。 LAN A: eth0 LAN B: eth1
pppoe_user	PPPoEのユーザー名を設定します。
pppoe_password	PPPoEのパスワードを設定します。
pppoe_dns	PPPoEのDNSサーバーを設定します。
pppoe_firewall	PPPoEのファイアウォールを設定します。 NONE: 0 STANDALONE: 1 MASQUERADE: 2

※Light版 rootfsの場合、別途PPPoEのソフトウェアが必要です。

スタティックルーティング機能設定

項目名	設定内容
static_route	スタティックルーティング機能を設定します。 enabled: 有効 disabled: 無効
st_route_addr_1	スタティックルーティングの宛先IPアドレスを設定します。
st_route_gw_1	スタティックルーティングのゲートウェイアドレスを設定します。
st_route_mask_1	スタティックルーティングのネットマスクを設定します。
st_route_if_1	スタティックルーティングのインターフェイスを設定します。
:	
st_route_addr_32	スタティックルーティングの宛先IPアドレスを設定します。
st_route_gw_32	スタティックルーティングのゲートウェイアドレスを設定します。
st_route_mask_32	スタティックルーティングのネットマスクを設定します。
st_route_if_32	スタティックルーティングのインターフェイスを設定します。

※項目名の数字は設定No.を示します。(最大32まで)

ポートフォワーディング機能設定

項目名	設定内容
port_forward	ポートフォワーディング機能を設定します。 enabled: 有効 disabled: 無効
port_fw_sif_1	ポートフォワーディング入力インターフェイスを設定します。
port_fw_sport_1	ポートフォワーディング入力ポートを設定します。
port_fw_daddr_1	ポートフォワーディング宛先IPアドレスを設定します。
port_fw_dport_1	ポートフォワーディング宛先ポートを設定します。
:	:
port_fw_sif_32	ポートフォワーディング入力インターフェイスを設定します。
port_fw_sport_32	ポートフォワーディング入力ポートを設定します。
port_fw_daddr_32	ポートフォワーディング宛先IPアドレスを設定します。
port_fw_dport_32	ポートフォワーディング宛先ポートを設定します。

※項目名の数字は設定No.を示します。(最大32まで)

IPフィルタ機能設定

項目名	設定内容
ipfilter	IPフィルタ機能を設定します。 enabled: 有効 disabled: 無効
ipfilter_kind_1	フィルタ種別を設定します。 許可： ACCEPT 破棄： DROP
ipfilter_proto_1	プロトコルを設定します。 tcp, udp, icmp, all
ipfilter_saddr_1	送信元IPアドレスを設定します。
ipfilter_sport_1	送信元ポートを設定します。
ipfilter_daddr_1	送信先IPアドレスを設定します。
ipfilter_dport_1	送信先ポートを設定します。
:	:
ipfilter_kind_64	フィルタ種別を設定します。 許可： ACCEPT 破棄： DROP
ipfilter_proto_64	プロトコルを設定します。 TCP, UDP, ICMP, ALL
ipfilter_saddr_64	送信元IPアドレスを設定します。
ipfilter_sport_64	送信元ポートを設定します。
ipfilter_daddr_64	送信先IPアドレスを設定します。
ipfilter_dport_64	送信先ポートを設定します。

※項目名の数字は設定No.を示します。(最大64まで)

Webサーバー搭載の場合、PC等のWebブラウザからLAN経由でCONPROSYSに接続しネットワーク設定を行うことができます。詳細は『**Web Setupについて（P54)**』を参照ください。

6. Web Setupについて

Rootfsタイプを下記のものに指定した場合、Web Setup機能を搭載します。

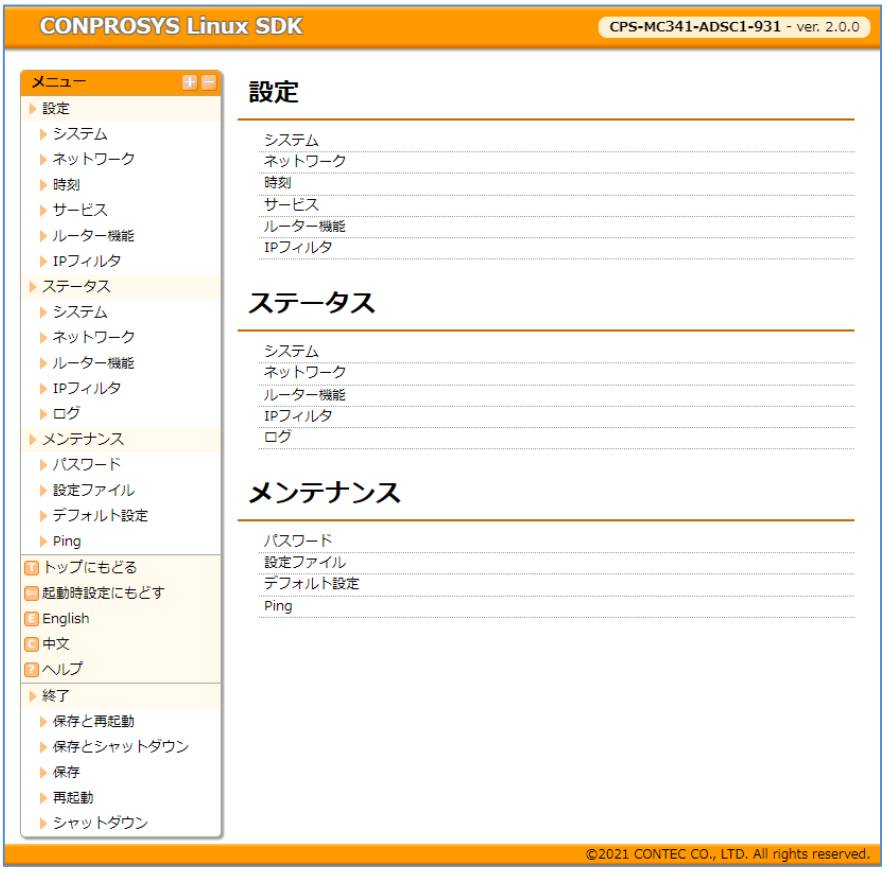
Ubuntu20.04 (include SDK)　Ubuntu20.04 SDK付(セルフビルド版CONPROSYS Linux SDK)
light (busybox)　軽量版rootfs
※light (busybox)は、搭載ツールにApache2, PHP5を含めた時にWeb Setup機能が搭載されます。

このWeb Setup機能にはネットワークや時刻等の設定機能、システム情報やネットワーク等の状態表示機能等が付いています。PC等のWebブラウザからCONPROSYSのIPアドレスへ直接アクセスすることで、CONPROSYSの設定画面が表示されます。

例： 初期設定時、LAN AポートにPCを接続して確認
http://10.1.1.101/

ログイン： admin
パスワード： password

Web設定画面



Web Setup機能は次の機能があります。

1. 設定メニュー

次の設定を行うことができます。

設定メニュー項目

設定種類	設定内容	初期値	備考
システム	ホスト名	(設定無し)	設定が無い場合、下記ホスト名に設定 モデル名＋MACアドレスの下位3バイト
ネットワーク	有線LAN A	10.1.1.101(固定IP)	
	有線LAN B	DHCP	
	3G/LTEネットワーク		3G/LTEモジュール搭載機種のみ
	無線LAN	DHCP	対応USB無線アダプターと接続時のみ
時刻	NTPサーバー	(設定無し)	
	手動設定		
サービス起動	SSHサーバー	システム起動時：有効	
	FTPサーバー	システム起動時：無効	
	SAMBAサーバー	システム起動時：無効	
ルーター機能	ルーター機能	システム起動時：無効	
	WANインターフェイス		
	DHCPサーバー機能	システム起動時：無効	
	スタティックルーティング設定	システム起動時：無効	設定最大数：32
	ポートフォワーディング設定	システム起動時：無効	設定最大数：32
IPフィルタ	IPフィルタ設定	システム起動時：無効	設定最大数：64

2. ステータスメニュー

次のステータスを確認することができます。

ステータスメニュー項目

項目	内容
システム	ホスト名、シリアル番号、ディストリビューション/カーネル情報、ディスク/メモリ使用量等を表示します。
ネットワーク	IPアドレス、MACアドレス、RX/TXバイト等を表示します。
ルーター機能	ルーティング情報を表示します。
IPフィルタ	IPフィルタリング情報を表示します。
ログ	syslog等のログを表示します。

3. メンテナンスメニュー

次のメンテナンス処理を行えます。

メンテナンスメニュー項目

項目	内容
パスワード	Web設定画面にアクセスするためのパスワードを変更できます。
設定ファイル	設定ファイルのバックアップとリストアができます。
デフォルト設定	出荷時のデフォルト設定にもどすことができます。
Ping	ネットワークの導通を確認するためPing をすることができます。

4. 終了メニュー

次の処理を行えます。

終了メニュー項目

項目	内容
保存と再起動	設定項目を保存し、再起動します。
保存とシャットダウン	設定項目を保存し、シャットダウン(システム停止)します。
保存	設定項目を保存します。
再起動	再起動します。設定項目を保存せずに再起動した場合は設定前の状態に戻ります。
シャットダウン	シャットダウン(システム停止)します。設定項目を保存せずにシャットダウンした場合は設定前の状態に戻ります。

Web Setup機能の使い方の詳細については、Webメニューにある「ヘルプ」をご参照ください。
Web設定の項目は下記のファイルで管理しています。

設定ファイル：

/etc/conprosys/config.ini

出荷時設定ファイル：

/etc/conprosys/config_def.ini

Webファイルは下記ディレクトリで管理しています。

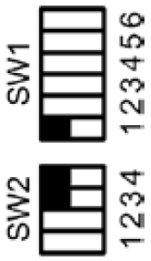
Webコンテンツディレクトリ：

[Ubuntu20.04 (include SDK)]
/var/www/html/
[軽量版rootfs]
/opt/htdocs/

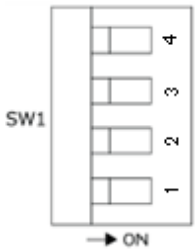
7. DIP SWによる初期化設定

DIP SWの設定により、電源起動後にLAN AのIPアドレスのみを初期化、または出荷時のデフォルト設定にもどすことができます。

コンパクトタイプ



スタックタイプ



DIP SWの設定内容

SW設定	内容
SW1-2のみON	電源をONにすると、IPアドレス設定を出荷時の設定で起動します。 ユーザー/パスワード、グループ設定については元の設定のまま起動します。 Web画面では、現在のIPアドレス設定、ユーザー/パスワード設定を確認できます。
SW1-2、SW1-3をON	電源をONにすると、各種設定を工場出荷時に初期化します。 完了するとLEDのPWRとST1が点滅します。点滅が確認できたら、2と3をOFFに戻し、再起動してください。

ビルド

1. ビルド手順

ターゲットとなるCONPROSYS製品毎の実行環境を生成するために、必要に応じて次の内容を実施します。

1 ビルドの初期設定（configure.sh)

./configure.shを実行し、ベースとなるターゲットの実行環境を生成します。

また、セルフビルド版のCONPROSYS Linux SDKも、このビルドの初期設定より実行環境を生成することができます。

2 ビルドの環境設定（. sdkenv.txt)

ビルドおよびSDカード生成を実行するための環境変数の設定を行います。

ログイン後、ビルド実行前に必ずこの設定を行ってください。

3 bootloaderのビルド

電源ON後の起動プログラムを変更したい場合にビルドを行います。

通常行う必要はありません。

4 kernelのビルド

Linux kernelを変更したい場合にビルドを行います。

デフォルト設定されていないkernelの機能やドライバの追加/変更/削除したい時に行ってください。

kernelを変更しない場合は行う必要はありません。

5 サンプルライブラリのビルド

サンプルライブラリを変更したい場合にビルドを行います。

サンプルライブラリを変更しない場合は行う必要はありません。

6 サンプルアプリケーションのビルド

サンプルアプリケーションをターゲットで実行したい場合ビルドを行います。

サンプルアプリケーションを使用しない場合や、CONPROSYS上でセルフビルドを行う場合は、ビルドの必要はありません。

2. ターゲットのbootloaderのビルド

bootloaderのビルドは u-bootのディレクトリで行うことができます。通常は特に行うことはありませんが、bootloaderのソースコードが変更になった時、もしくはコンパイラオプションを変更する時に実施してください。u-bootはSDカード起動用と内蔵NOR FLASH起動用の２種類があります。

1. SDカード起動用のビルド

カレント移動コマンド：

```
cd ${CPS_SDK_ROOTDIR}/u-boot
```

ビルドコマンド：

```
make am335x_evm
```

ビルドして生成されたモジュール(MLO, u-boot.img)は、次のコマンドでターゲット向けのbootディレクトリにコピーします。

コマンド：

```
cp -p MLO u-boot.img ${CPS_SDK_INSTALL_FULLDIR}/boot
```

ビルド時に生成されたオブジェクトファイル等を削除したい場合は次のコマンドを実行します。

コマンド：

```
make distclean
```

2. 内蔵NOR FLASH起動用のビルド

内蔵NOR FLASHから起動する bootloaderモジュールは、SDカードで起動するbootloaderモジュールと異なります。このbootloaderモジュールは、次の方法でビルドします。

カレント移動コマンド：

```
cd ${CPS_SDK_ROOTDIR}/u-boot
```

ビルドコマンド：

```
make am335x_evm_spiboot
```

ビルドして生成されたモジュール(MLO.byteswap, u-boot.img)は、内蔵NOR FLASHインストール用のrootfs(/InstallToMTD)へコピーします。

※内蔵NOR FLASHインストール用のrootfsの作成方法は『**内蔵NOR FLASH起動用インストールするためのrootfsセクションを作成（P36)**』を参照ください。

コマンド：

```
cp -p MLO.byteswap u-boot.img ${CPS_SDK_INSTALL_FULLDIR}/InstallerForFlash/InstallToMTD
```

3. ターゲットのkernelのビルド

kernelのコンフィグレーション/ビルドは kernelのディレクトリで行うことができます。

通常は特に行うことはありませんが、kernelのオプション変更やサポートされていないUSB等のデバイスドライバを実装する時に実施してください。

カレント移動コマンド：

```
cd ${CPS_SDK_ROOTDIR}/kernel
```

コンフィグレーションコマンド：

```
make menuconfig
```

ビルドコマンド：

```
make                ドライバ、DeviceTreeのビルド
make uImage         uImageのビルド
```

ビルドして生成されたモジュール(uImage)は、次のコマンドでターゲット向けのbootディレクトリにコピーします。

コマンド：

```
cp -p arch/arm/boot/dts/MC34*.dtb ${CPS_SDK_INSTALL_FULLDIR}/boot/
cp -p arch/arm/boot/uImage ${CPS_SDK_INSTALL_FULLDIR}/boot/uImage
```

カーネルビルド後、ドライバモジュールは次のコマンド例でターゲット向けのrootfsディレクトリへコピーします。

インストールコマンド例：

```
sudo -E make modules_install INSTALL_MOD_PATH=${CPS_SDK_INSTALL_FULLDIR}/${CPS_SDK_ROOTFS}
```

※ドライバモジュールはrootfsディレクトリ下にある次のディレクトリにインストールされます。

lib/modules/4.19.79/

他のrootfsディレクトリへインストールする場合はINSTALL_MOD_PATHの内容を変更してください。

ビルド時に生成されたオブジェクトファイル等を削除したい場合は次のコマンドを実行します。

コマンド：

```
make clean
```

kernelのコンフィグレーション情報が壊れるなどで、初期に戻したい場合は次のコマンドを実行します。

コマンド：

```
make distclean
make conprosys_defconfig
```

上記コマンドを実行後、コンフィグレーションおよびビルドを実行してください。

4. CPS-MxS341シリーズドライバのビルド

SDKには次のCPS-MxS341シリーズドライバのソースコードをdriver/cps-driversディレクトリに付属しています。

- cps-driver　(CPS-MxS341用システムドライバ)
- cpsaio　(CPS-MxS341用AIOドライバ)
- cpsdio　(CPS-MxS341用DIOドライバ)
- cpscom　(CPS-MxS341用COMドライバ)
- cpsssi　(CPS-MxS341用SSIドライバ)
- cpscnt　(CPS-MxS341用CNTドライバ)
- cps_iolib　(CPS-MxS341用IO汎用アクセスドライバ)

通常は特に行うことはありませんが、サンプルドライバの変更があった時に実施してください。

ドライバをビルドするにはkernelのビルド結果のソースコードが必要なため、事前にkernelのビルドを実施してください。(『**ターゲットのkernelのビルド (P61)**』: 参照)

各々のディレクトリ下で、下記コマンドでビルドできます。

コマンド:

```
make
```

ドライバビルド後、次のコマンド例ではターゲット向けrootfsディレクトリにコピーします。

コマンド例:

```
sudo -E make modules_install INSTALL_MOD_PATH=${CPS_SDK_INSTALL_FULLDIR}/${CPS_SDK_ROOTFS}
```

※ドライバモジュールはターゲットのrootfs下の次のディレクトリにインストールされます。

lib/modules/4.19.xx/extra

もし、他のrootfsディレクトリへインストールする場合はINSTALL_MOD_PATHの内容を変更してください。

⚠注意

本付属のドライバソフトは全製品に対応していません。

対応機器を確認の上、組み込んでください。

5. ターゲットのサンプルライブラリのビルド

SDKには下記の共有ライブラリ(shared object)のソースコードをlibディレクトリに付属しています。

- libcpsi2c.so (I2Cアクセスモジュールライブラリ)
- libCpsEeprom (EEPROMデータアクセスモジュールライブラリ)
- libCpsAio　(CPS-MxS341用AIOライブラリ)
- libCpsDio　(CPS-MxS341用DIOライブラリ)
- libCpsSsi　(CPS-MxS341用SSIライブラリ)
- libCpsCnt　(CPS-MxS341用CNTライブラリ)
- libconexio (CPS-MC341Q-ADSC1用920MHzモジュールライブラリ)
- SerialFunc (CPS-MC341Q-ADSC1用シリアルモジュールライブラリ)

通常は特に行うことはありませんが、サンプルライブラリの変更があった時に実施してください。

各々のディレクトリ下で、下記コマンドでビルドできます。

コマンド：

```
make
```

ライブラリビルド後、次のコマンドでターゲット向けrootfsディレクトリにコピーします。

コマンド：

```
sudo -E make install TARGET_ROOTFS=${CPS_SDK_INSTALL_FULLDIR}/${CPS_SDK_ROOTFS}
```

※ライブラリモジュールはターゲットのrootfs下の次のディレクトリにインストールされます。

usr/local/lib

もし、他のrootfsディレクトリへインストールする場合はTARGET_ROOTFSの内容を変更してください。

ライブラリを作成する場合、makefileやソースコードを参考ください。

⚠注意

- 付属のライブラリソフトは全製品に対応していません。対応機器を確認の上、組み込んでください。
- 付属の各々のサンプルライブラリソフトは付属のドライバソフトが必要になります。下記の依存関係を参考に組み込んでください。

  libCpsAio.so ---- cpsaio.ko ---- cps-driver.ko
  libCpsDio.so ---- cpsdio.ko ---- cps-driver.ko
  libCpsSsi.so ---- cpsssi.ko ---- cps-driver.ko
  libCpsCnt.so ---- cpscnt.ko ---- cps-driver.ko

6. ターゲットのサンプルアプリケーションのビルド

SDKにはサンプルのアプリケーションソースコードを下記のディレクトリに付属しています。

${CPS_SDK_ROOTDIR}/application/sample/

各ターゲットに応じたサンプルアプリケーションがあります。(『**サンプルプログラム対応表**』参照) ターゲットの動作評価やアプリケーションの参考に使用してください。サンプルプログラムをビルドするには、各々のディレクトリでmakeコマンドを実行すると実行形式のファイルが生成されます。

例：タイマーのサンプルプログラム

```
cd ${CPS_SDK_ROOTDIR}/application/sample/timer
make
```

サンプルプログラム対応表

サンプルプログラム	ディレクトリ application/sample/	CPS-MC341-ADSCx シリーズ CPS-MG341-ADSC1 シリーズ	CPS-MxS341-DSx シリーズ
TCP/IP サーバー/クライアント	socket	○	○
タイマー	timer	○	○
RS-485通信(コンパクトタイプ用)	RS485	○	
DI/DO, AI制御(多機能モデル用)	mc341_io	○	
DI/DO制御(コンパクトタイプ用)	spitest	○	
RTC ツール	rtc	○	○
AI/AO制御(スタックタイプ用)	mcs341_aio		○
DI/DO制御(スタックタイプ用)	mcs341_dio		○
COM制御(スタックタイプ用)	mcs341_com		○
CNT制御(スタックタイプ用)	mcs341_cnt		○
SSI制御(スタックタイプ用)	mcs341_ssi		○

○：対応　△：一部の機種で対応　空欄：非対応

これらのサンプルアプリケーションは全てのCONPROSYS製品に対応していません。

対応機器であってもデバイスポート等の違いにより、動作しないプログラムもありますのでプログラムを確認の上、ビルドしテストしてください。

また、必要なドライバ/ライブラリがあるサンプルソフトのビルドについて、事前にそれらのドライバ/ライブラリをビルドしてからサンプルアプリケーションのビルドをしてください。

デバイスポートについては、『**デバイスI/F (P72)**』を参照ください。

アプリケーションソフトを作成する場合、これらのmakefileやソースコードを参考ください。

7. 内蔵NOR FLASH起動用ramdisk.xzのビルド

内蔵NOR FLASHのrootfsはramdisk.xzというファイルに圧縮されて格納されています。

軽量版rootfsがtargetディレクトリ下にある場合、そのrootfsより次のコマンドでramdisk.xzを生成できます。

カレント移動コマンド：

```
cd ${CPS_SDK_ROOTDIR}/ramdisk
```

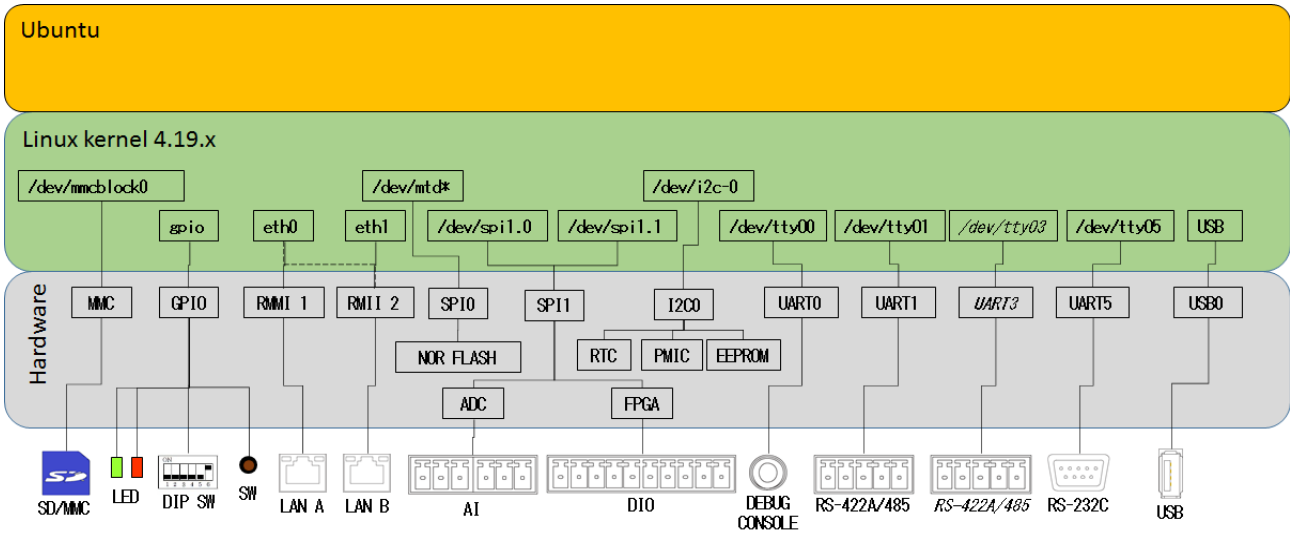
ビルドコマンド：

```
make
```

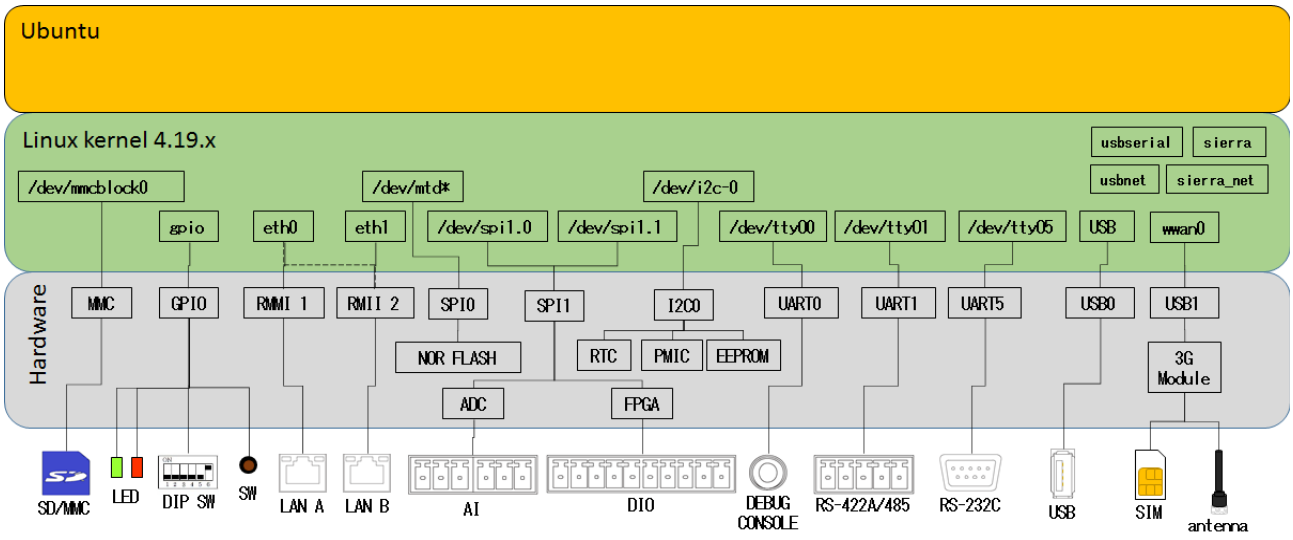
Appendix

1. ブロック図

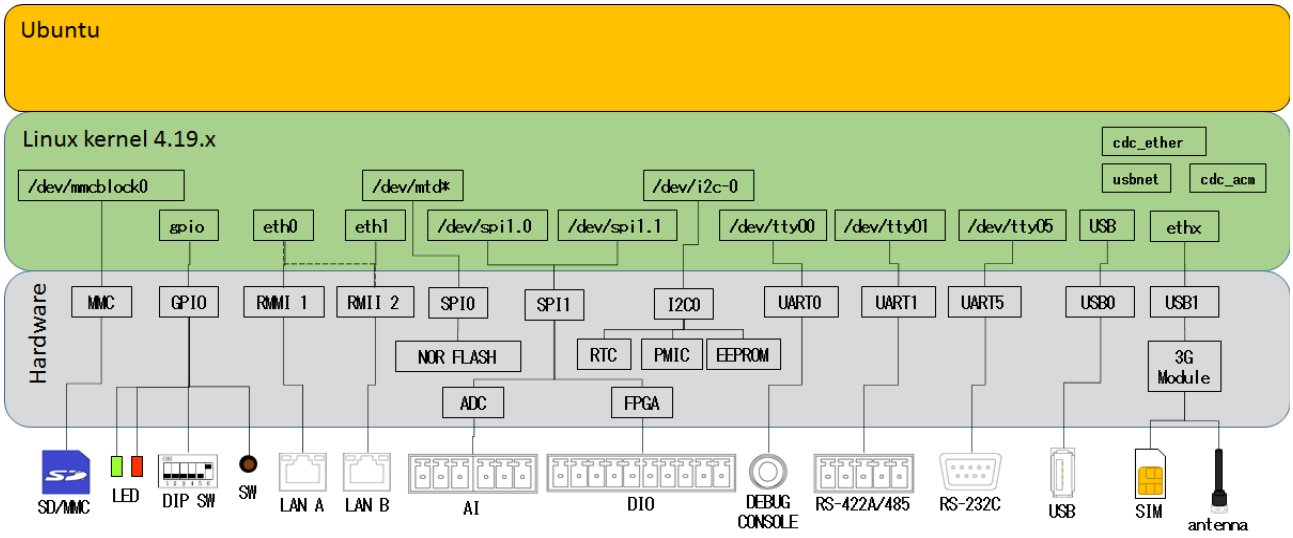
CPS-Mx341-ADSCxシリーズブロック図 (斜字はオプション)



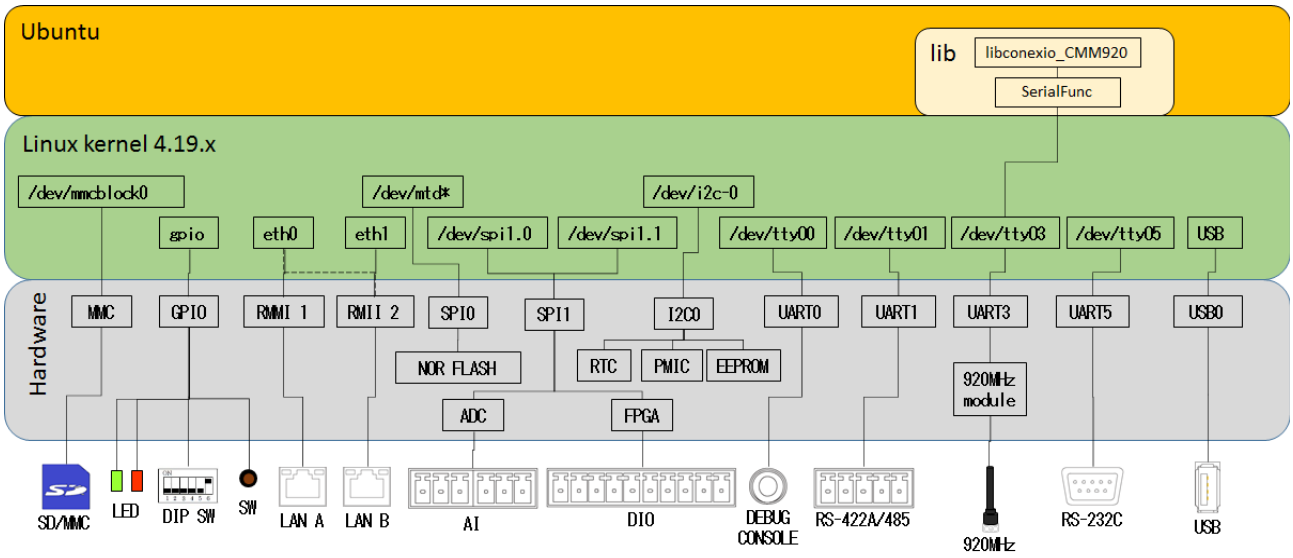
CPS-Mx341G-ADSC1(日本国内モデル)ブロック図



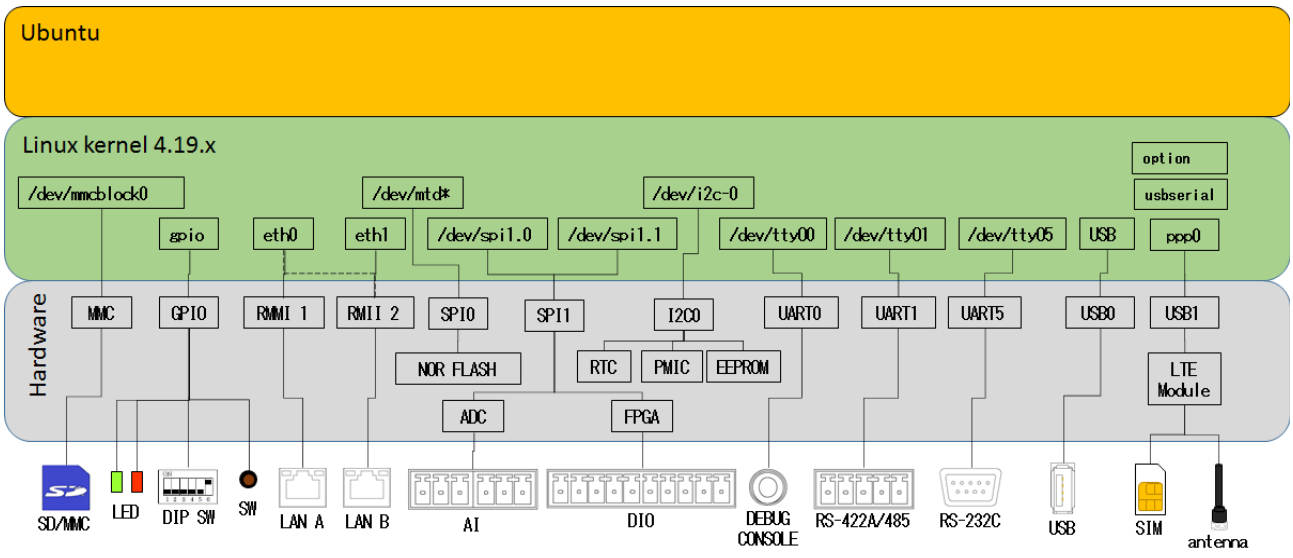
CPS-Mx341G-ADSC1(グローバルモデル)ブロック図



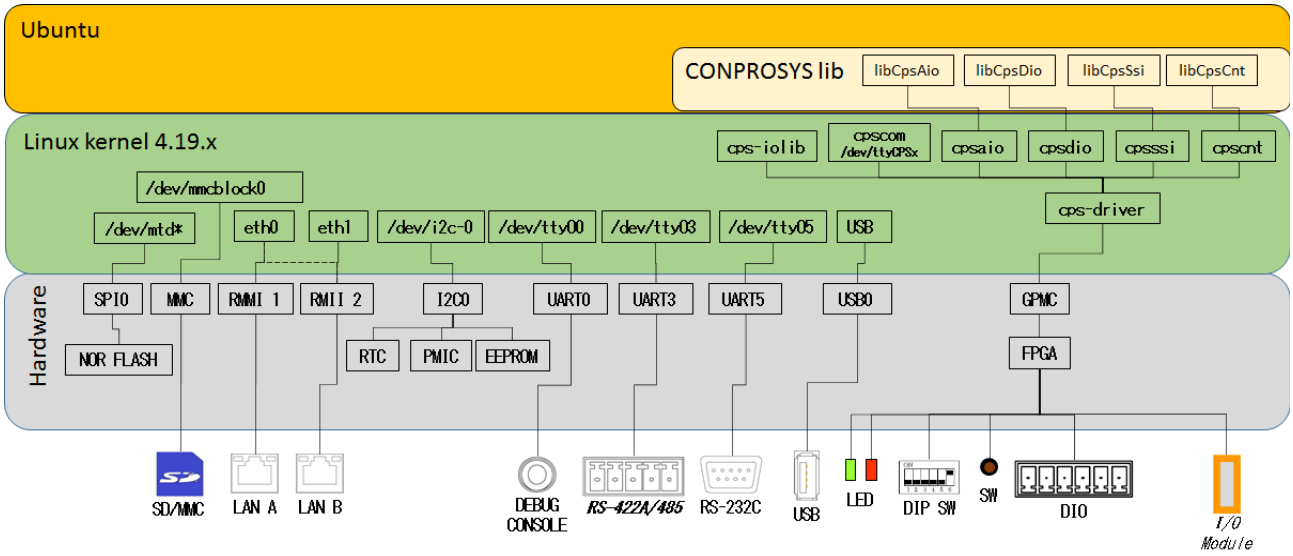
CPS-MC341Q-ADSC1ブロック図



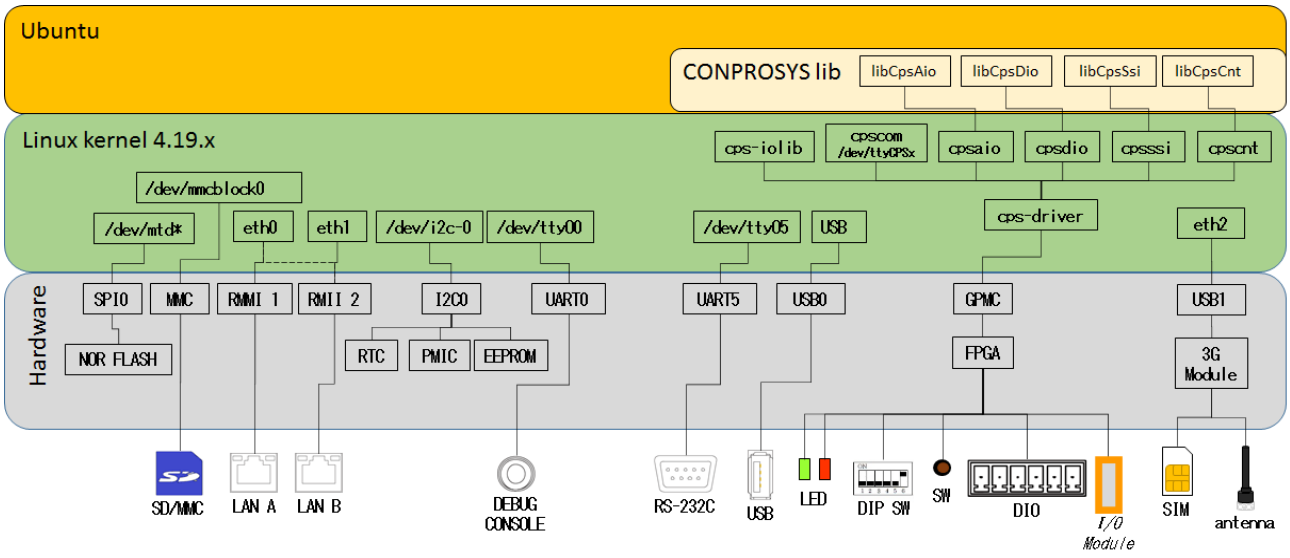
CPS-MG341G5-ADSC1ブロック図



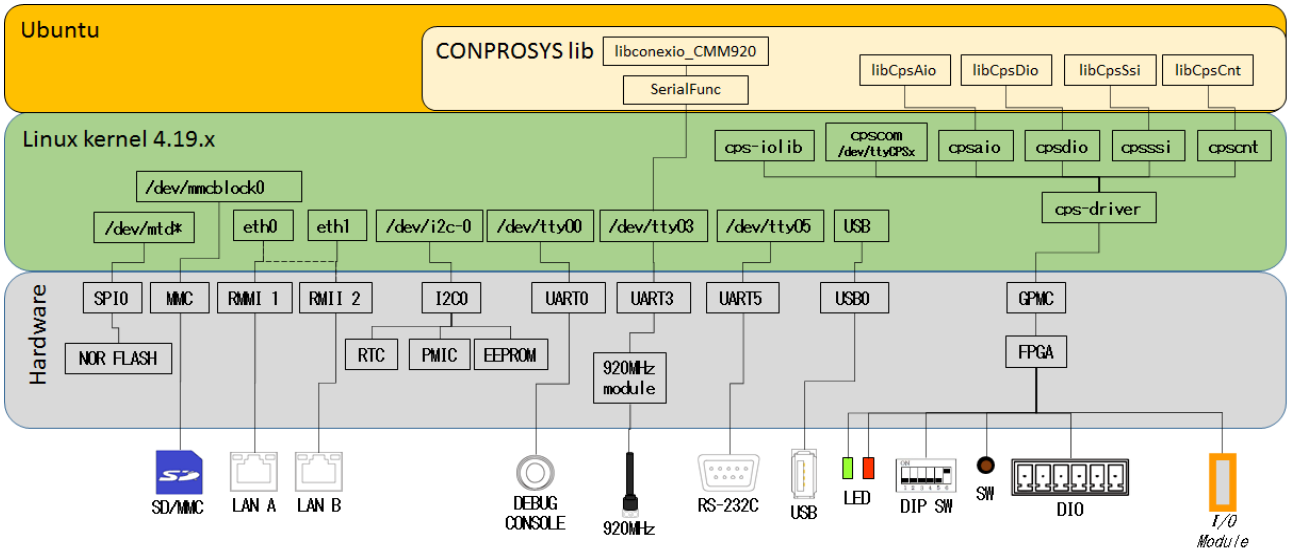
CPS-MxS341-DSxシリーズブロック図（斜字はオプション）



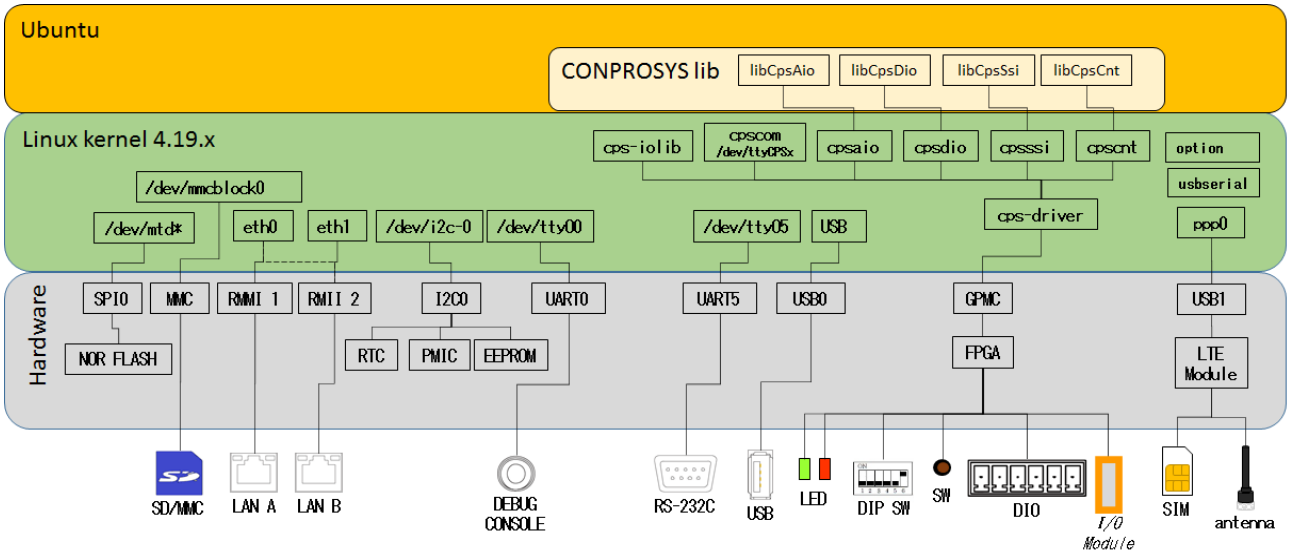
CPS-MCS341G-DS1ブロック図（斜字はオプション）



CPS-MCS341Q-DS1ブロック図（斜字はオプション）



CPS-MxS341G5-DS1ブロック図（斜字はオプション）



2. デバイスI/F

CONPROSYS特有のデバイスI/FはLinux上で次の表に示すようにアクセスすることができます。
機器によってポートが違うことがありますので注意ください。

UART制御デバイス

モデル	/dev/ttyO1	/dev/ttyO2	/dev/ttyO3	/dev/ttyO4	/dev/ttyO5
CPS-MC341-ADSC1	RS-422A/485 (COM A)	-	-	-	RS-232C (COM B)
CPS-MC341-ADSC2	RS-422A/485 (COM A)	-	RS-422A/485 (COM C)	-	RS-232C (COM B)
CPS-MC341G-ADSC1 CPS-MG341G5-ADSC1	RS-422A/485 (COM A)	-	-	-	RS-232C (COM B)
CPS-MC341Q-ADSC1	RS-422A/485 (COM A)	-	920MHz module	-	RS-232C (COM B)
CPS-MCS341-DS1 CPS-MGS341-DS1	-	-	-	-	RS-232C
CPS-MCS341G-DS1 CPS-MCS341G5-DS1 CPS-MGS341G5-DS1	-	-	-	-	RS-232C
CPS-MCS341Q-DS1	-	-	920MHz module	-	RS-232C

SPI制御デバイス

モデル	/dev/spidev1.0	/dev/spidev1.1	/dev/spidev1.2
CPS-MC341-ADSCx CPS-MC341G-ADSC1 CPS-MG341G5-ADSC1 CPS-MC341Q-ADSC1	AI (ADC / CLK=6MHz)	DIO (FPGA / CLK=24MHz)	-
CPS-MCS341-DS1 CPS-MGS341-DS1 CPS-MCS341G-DS1 CPS-MCS341G5-DS1 CPS-MGS341G5-DS1 CPS-MCS341Q-DS1	-	-	-

カッコ内は接続されるデバイスとSPI制御MAXクロック値

GPIO制御デバイス（LED系）

モデル	GPIO 26	GPIO 27	GPIO 67
CPS-MC341-ADSCx CPS-MC341G-ADSC1 CPS-MC341Q-ADSC1	ST1 Green (Out)	ST2 Red (Out)	Power (Out)
CPS-MG341G5-ADSC1	ST1 Green (Out)	ST2 Red (Out)	Power (Out)
CPS-MCS341-DS1 CPS-MGS341-DS1 CPS-MCS341G-DS1 CPS-MCS341Q-DS1	-	-	-
CPS-MCS341G5-DS1 CPS-MGS341G5-DS1	-	-	-

GPIO制御デバイス（Switch系）

モデル	GPIO 32	GPIO 33	GPIO 34	GPIO 35	GPIO 87
CPS-MC341-ADSCx CPS-MC341G-ADSC1 CPS-MG341G5-ADSC1 CPS-MC341Q-ADSC1	DIP SW1-2 (In)	DIP SW1-3 (In)	DIP SW1-4 (In)	Shutdown SW (In)	-
CPS-MCS341-DS1 CPS-MGS341-DS1 CPS-MCS341G-DS1 CPS-MCS341G5-DS1 CPS-MGS341G5-DS1 CPS-MCS341Q-DS1	-	-	-	-	Shutdown SW (In)

GPIO制御デバイス（Board制御系）

モデル	GPIO 22	GPIO 23	GPIO 36	GPIO 37	GPIO 105
CPS-MC341-ADSC1	-	-	-	-	Power RESET (Out)
CPS-MC341-ADSC2	-	-	RS485 Power (Out)	-	Power RESET (Out)
CPS-MC341G-ADSC1	-	LDO_SHUTDOWN (Out)	3G Power (Out)	3G Reset (Out)	Power RESET (Out)
CPS-MG341G5-ADSC1	PWR_ON_N_3V3 (Out)	PWRKEY (Out)	LTE Power (Out)	LTE Reset (Out)	Power RESET (Out)
CPS-MC341Q-ADSC1	-	-	920M Power (Out)	920M Reset (Out)	Power RESET (Out)
CPS-MCS341-DS1 CPS-MGS341-DS1 CPS-MCS341G-DS1 CPS-MCS341G5-DS1 CPS-MGS341G5-DS1 CPS-MCS341Q-DS1	-	-	-	-	Power RESET (Out)

カッコ内は入出力方向を示す

USB-Serial制御デバイス

モデル	/dev/ttyUSB0	/dev/ttyUSB1	/dev/ttyUSB2	/dev/ttyUSB3	/dev/ttyUSB3
CPS-MC341-ADSCx CPS-MC341Q-ADSC1	Optional Serial Device				
CPS-MC341G-ADSC1 (日本国内モデル)	Sierra USB modem	Sierra USB modem	Sierra USB modem	Sierra USB modem	Optional Serial device
CPS-MC341G-ADSC1 (グローバルモデル)	Optional Serial device				
CPS-MG341G5-ADSC1	Quectel USB modem	Quectel USB modem	Quectel USB modem	Quectel USB modem	Optional Serial device
CPS-MCS341-DS1 CPS-MGS341-DS1 CPS-MCS341G-DS1 CPS-MCS341Q-DS1	Optional Serial device				
CPS-MCS341G5-DS1 CPS-MGS341G5-DS1	Quectel USB modem	Quectel USB modem	Quectel USB modem	Quectel USB modem	Optional Serial device

コンパクトタイプ ADC / DAC / FPGA(DIO) 使用デバイス

モデル	デバイス	メーカー	デバイス型番	制御ポート
CPS-MC341-ADSC1	ADC	Analog Devices	ADC7327	/dev/spidev1.0
CPS-MC341-ADSC2 CPS-MC341G-ADSC1 CPS-MC341Q-ADSC1 CPS-MG341G5-ADSC1	FPGA (DIO)	Lattice Semiconductor	LCMXO2-640HC-4TG100I	/dev/spidev1.1

AIOのデバイス制御詳細に関しては、上記の情報より各デバイスメーカーのデータシートを入手し参照ください。DIOのデバイス制御(FPGA)に関しては、『**コンパクトタイプ CPS-Mx341-ADSCxシリーズ (P77)**』を参照ください。

スタックタイプ FPGA使用デバイス

モデル	デバイス	メーカー	デバイス型番	制御ポート
CPS-MCS341-DS1 CPS-MGS341-DS1 CPS-MCS341G-DS1 CPS-MCS341Q-DS1 CPS-MCS341G5-DS1 CPS-MGS341G5-DS1	FPGA	Lattice Semiconductor	LCMXO2-7000HC-4FTG256I	GPMC

デバイス制御(FPGA)に関しては、『**スタックタイプ CPS-MxS341-DSxシリーズ (P84)**』を参照ください。

スタックタイプCOMデバイス

モデル	/dev/ttyCPS0	/dev/ttyCPS1	/dev/ttyCPS2	/dev/ttyCPS3	…	/dev/ttyCPS62	/dev/ttyCPS63
CPS-COM-1PC	RS-232C	-	RS-232C	-	…	RS-232C	-
CPS-COM-2PC	RS-232C	RS-232C	RS-232C	RS-232C	…	RS-232C	RS-232C
CPS-COM-1PD	RS-422A/485	-	RS-422A/485	-	…	RS-422A/485	-
CPS-COM-2PD	RS-422A/485	RS-422A/485	RS-422A/485	RS-422A/485	…	RS-422A/485	RS-422A/485

スタックタイプAIO 制御デバイス

モデル	/dev/cpsaio0	/dev/cpsaio1	…	/dev/cpsaio30	/dev/cpsaio31
CPS-AI-1608LI/ CPS-AI-1608ALI	AI	AI	…	AI	AI
CPS-AO-1604LI CPS-AO-1604ALI	AO	AO	…	AO	AO

スタックタイプDIO 制御デバイス

モデル	/dev/cpsdio0	/dev/cpsdio1	…	/dev/cpsdio30	/dev/cpsdio31
CPS-DIO-0808L/ CPS-DIO-0808BL	DIO	DIO	…	DIO	DIO
CPS-DI-16L/ CPS-DI-16RL	DI	DI	…	DI	DI
CPS-DO-16L/ CPS-DO-16RL/ CPS-RRY-4PCC	DO	DO	…	DO	DO

スタックタイプSSI 制御デバイス

モデル	/dev/cpsssi0	/dev/cpsssi1	…	/dev/cpsssi30	/dev/cpsssi31
CPS-SSI-4P/ CPS-SSI-4C	SSI	SSI	…	SSI	SSI

スタックタイプFPGA制御デバイス

モデル	/dev/cps-iolib
CPS-MCS341-DSx CPS-MGS341-DS1 CPS-MCS341G-DS1 CPS-MCS341Q-DS1 CPS-MCS341G5-DS1 CPS-MGS341G5-DS1	GPMC

Networkデバイス

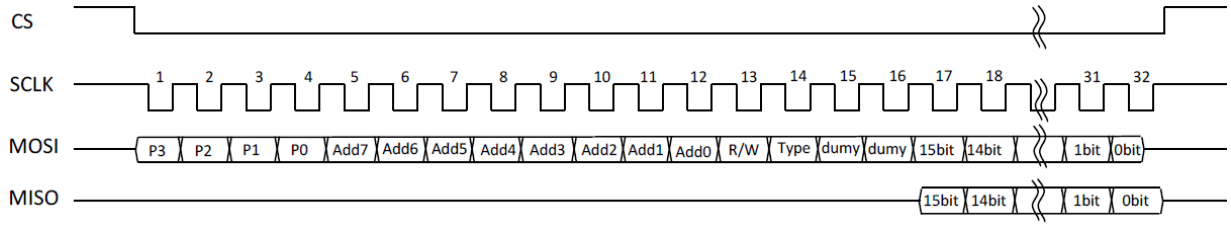
Network Category	eth0	eth1	eth2	can0	can1	wwan0	ppp0
1 LAN(Hub Mode) Type	LAN A/B	-	-	-	-	-	-
2 LAN Type	LAN A	LAN B	-	-	-	-	-
3G搭載モデル(日本国内モデル) 1 LAN(Hub Mode) Type	LAN A/B	-	-	-	-	3G	-
3G搭載モデル(日本国内モデル) 2 LAN Type	LAN A	LAN B	-	-	-	3G	-
3G搭載グローバルモデル 1 LAN(Hub Mode) Type	LAN A/B	3G	-	-	-	-	-
3G搭載グローバルモデル 2 LAN Type	LAN A	LAN B	3G	-	-	-	-
LTE搭載モデル 1 LAN Type	LAN A/B	-	-	-	-	-	LTE
LTE搭載モデル 2 LAN Type	LAN A	LAN B	-	-	-	-	LTE

3. FPGA I/Oマップ

1. コンパクトタイプ CPS-Mx341-ADSCxシリーズ

メーカー　：　Lattice Semiconductor
デバイス型番　：　LCMXO2-640HC-4TG100I
インターフェイス　：　SPI

SPI信号タイミング



MOSI　：　SCLKの立下りで、スレーブが信号をラッチ
MISO　：　SCLKの立ち上がりでスレーブ信号を出力、 SCLKの立下りでマスタが信号をラッチ

SPI信号フォーマット

Register Page	Address	R/W	Access Type	Dummy	Data
4bit	8bit	1bit	1bit	2bit	16bit

- R/W　：　0 = Read、 1 = Write
- Access Type　：　0 = Byte Access、 1 = Word Access
- Dummy　：　常に 0

Byteアクセス時は、 データを下詰めで16bitデータにして送受信を行います。

例：　Page = 0h、Address=12hに00AAhをWrite する場合

0x0 12 C 00AA

Products Category

Products Category	Function	Register Page	適用機器
01h	デジタル入出力部	0h	CPS-MC341-ADSCx
02h	アナログ入力部	1h	CPS-MC341-ADSCx
03h	カウンタ部	2h	CPS-MC341-ADSCx

デジタル入出力部ポートマップ（Page 0h)

Address	Read/Write種別	内容
00h - 01h	R	システム予約エリア
02h - 03h	R	システム予約エリア
04h - 0Ch	R	未使用
0Eh - 0Fh	R	システム予約エリア
10h - 11h	R	デジタル入力ポート
12h - 13h	R/W	デジタル出力ポート
14h - 17h	R	未使用
18h - 19h	R/W	デジタルフィルタ設定時間
1Ah - 1Fh	R	未使用
1Ch - 1Dh	R/W	内蔵電源 ON/OFF※
1Eh - 1Fh	R	未使用
20h - 21h	R/W	システム予約エリア
22h - 23h	R	未使用
24h - 25h	R/W	システム予約エリア
26h - FFh	R	未使用

※CPS-MC341-ADSC1-931のみ対応

アナログ入力部ポートマップ（Page 1h)

Address	Read/Write種別	内容
00h - 01h	R	システム予約エリア
02h - 03h	R	システム予約エリア
04h - 27h	R	未使用
28h - 29h	R/W	アナログ入力部
2Ah - FFh	R	未使用

カウンタ入出力部ポートマップ（Page 2h)

Address	Read/Write種別	内容
00h - 01h	R	システム予約エリア
02h - 03h	R	システム予約エリア
04h - 0Fh	R	未使用
10h - 11h	R/W	Direct Counter Data下位 (R) / Read Channel Select (W)
12h - 13h	R/W	Direct Counter Data上位 (R) / Direct Counter Latch Select (W)
14h - 15h	R/W	Counter Select Enable Status
16h - 17h	R	未使用
18h - 19h	R/W	Command Select
1Ah - 1Bh	R	未使用
1Ch - 1Dh	R/W	Counter Input / Output data 下位
1Eh - 1Fh	R/W	Counter Input / Output data 上位
20h - 21h	W	システム予約エリア
22h - 23h	W	システム予約エリア
24h - 25h	R/W	システム予約エリア
26h - 27h	R/W	システム予約エリア
2Ah - FFh	R	未使用

デジタル入力ポート（Page 0h / Address 10h - 11h） R

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0	DI7	DI6	DI5	DI4	DI3	DI2	DI1	DI0

このポートは、デジタル入力端子の値を取得します。デジタルフィルタを設定している場合は、フィルタ通過後の値が取得されます。

※CPS-MC341-ADSCxシリーズはDI0 - DI3のみ有効です。

デジタル出力ポート（Page 0h / Address 12h -13h） R/W

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D05	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0	DO7	DO6	DO5	DO4	DO3	DO2	DO1	DO0

このポートは、デジタル出力端子の値を設定、または設定値を取得します。

※CPS-MC341-ADSCxシリーズはDO0 - DO1のみ有効です。

デジタルフィルタ設定時間（Page 0h / Address 18h - 19h） R/W

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	ST4	ST3	ST2	ST1	ST0	0	0	0	0	0	0	0	0

このポートは、デジタル入力端子に適用するデジタルフィルタの値を設定、または設定値を取得します。設定値は全入力端子に適用されます。設定値は[デジタルフィルタ設定項目]を参照してください。

デジタルフィルタ設定項目

設定項目	名称	意味	設定項目	初期値
ST4～0	デジタルフィルタ設定時間	デジタルフィルタの時間を設定します。	0: フィルタ機能未使用	0 [フィルタ機能未使用]
			1: 0.25μsec	
			2: 0.5μsec	
			3: 1μsec	
			4: 2μsec	
			5: 4μsec	
			6: 8μsec	
			7: 16μsec	
			8: 32μsec	
			9: 64μsec	
			10: 128μsec	
			11: 256μsec	
			12: 512μsec	
			13: 1.024msec	
			14: 2.048msec	
			15: 4.096msec	
			16: 8.192msec	
			17: 16.384msec	
			18: 32.768msec	
			19: 65.536msec	
			20: 131.072msec	
			21～31: Reserve	

内蔵電源 ON/OFF 設定ポート (Page 0h / Address 1Ch - 1Dh) R/W

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PWEn

このポートは、デジタル入力ポート用の内蔵電源の有効(ON)/無効(OFF)を設定します。

このポートをReadすることで、設定状態を確認することができます。設定値は[内蔵電源 ON/OFF設定]を参照してください。

内蔵電源 ON/OFF設定

設定項目	名称	意味	設定項目	初期値
PWEn	内蔵電源有効	内蔵電源を有効(ON)にします。	0: 無効(OFF) 1: 有効(ON)	0 [無効]

アナログ入力ポート (Page 1h / Address 28h - 29h)　R/W

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	03	D2	D1	00
0	0	0	0	0	0	0	0	0	0	0	0	0	0	AT1	AT0

このポートは、アナログ入力チャンネルの値を取得します。チャンネル間絶縁機能が必要な場合は、両方のスイッチを同時にONしないでください。同時にONするとチャンネル間絶縁の機能が失われます。

デジタル入力ポート (Page 0h / Address 10h - 11h)　R

Addr	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
10h	D15	D14	D13	D12	D11	D10	D09	D08	D07	D06	D05	D04	D03	D02	D01	D00
12h	0	0	0	0	0	0	0	0	D23	D22	D21	D20	D19	D18	D17	D16

このポートは、ラッチされたカウンタデータを読むことができます。

読むデータは『カウンタ読み出しチャンネル設定ポート(Page 2h / Address 10h)　W』で設定します。

カウンタ読み出しチャンネル設定ポート (Page 2h / Address 10h)　W

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Sel0

このポートは、カウンタデータ読み出しポートで読み出すチャンネルを選択します。

カウンタデータの読み出しは、『カウンタデータ読み出しポート(Page 2h / Address 10h - 13h)　R』で行います。

カウンタ読み出し設定

設定項目	名称	意味	設定項目	初期値
Sel0	カウンタ読み出しチャンネル	カウンタデータ読み出しポートから読み出すチェンネルを設定します。	0: Channel 0 1: Channel 1	0 [Channel 0]

カウンタデータラッチ設定ポート (Page 2h / Address 12h)　W

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	Ch01	Ch00

このポートに[1]を書き込むことで、カウンタデータがラッチされます。カウンタデータ読み出しポートからは、ここでラッチしたカウント値が読み出されます。

カウンタ有効チャンネル設定ポート (Page 2h / Address 14h)　R/W

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	Ch01	Ch00

このポートは、カウンタ有効チャンネルの設定および設定状態を読み出します。

カウンタ有効チャンネル設定ポート (Page 2h / Address 14h)　W

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0	0	Cmd06 - 00						

このポートは、次のコマンドコードを実行するためのオペレーションコマンドポートです。

コマンドコード一覧：

08h: Ch0カウンタモード (Write)

09h: ch1カウンタモード (Write)

18h: Ch0比較レジスタ0 (Write)

19h: Ch1比較レジスタ0 (Write)

20h: Ch0比較レジスタ1 (Write)

21h: Ch1比較レジスタ1 (Write)

38h: カウント一致ステータス確認/クリア (Read/Write)

3Ah: キャリーステータス確認/クリア(Read/Write)

3Dh: ソフトゼロクリア (Write)

Writeコマンド使用時にはデータアドレスポート(Page 2h / 1Ch - 1Fh)にデータを設定します。Readコマンド使用時にはデータアドレスポート(Page 2h / 1Ch - 1Fh)からデータを読み出します。コマンドポートを制御した後、データアドレスポートも制御してください。各コマンドコードに対するデータアドレスポートのフォーマットは、『カウンタ入出力部ポートマップ(Page 2h)』～『内蔵電源 ON/OFF 設定ポート(Page 0h / Address 1Ch - 1Dh)　R/W』を参照してください。

Ch0 / Ch1カウンタモード (カウンタコマンドコード：08h / 09h) W

Addr	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
1Ch	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1Eh	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

カウンタの動作モードの設定を行います。設定は入力チャンネル毎に行います。

Ch0 / Ch1比較レジスタ0 (カウンタコマンドコード：18h / 19h) W

Addr	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
1Ch	Data00 - 15															
1Eh	0	0	0	0	0	0	0	0	Data16 - 25							

Ch0 - Ch1のカウント値比較レジスタ0にデータを設定します。

Ch0 / Ch1比較レジスタ1 (カウンタコマンドコード：20h / 21h) W

Addr	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
1Ch	Data00 - 15															
1Eh	0	0	0	0	0	0	0	0	Data16 - 25							

Ch0 - Ch1のカウント値比較レジスタ1にデータを設定します。

カウント一致ステータス確認 / クリア (カウンタコマンドコード：38h) R/W

Addr	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
1Ch	0	0	0	0	0	0	Cmp1_Ch1	Cmp1_Ch0	0	0	0	0	0	0	Cmp0_Ch1	Cmp0_Ch0
1Eh	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Read時、条件が成立したビットが 1になります。

Write時、対応ビットに1をセットすることでリセットします。

キャリーステータス確認 / クリア (カウンタコマンドコード：3Ah) R/W

Addr	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
1Ch	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Carry Ch1	Carry Ch0

Read時、条件が成立したビットが 1になります。

Write時、対応ビットに1をセットすることでリセットします。

ソフトゼロクリア (3Dh) W

Addr	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
1Ch	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Ch1	Ch0

Write時、対応ビットに1をセットすることでリセットします。

2. スタックタイプ CPS-MxS341-DSxシリーズ

メーカー　：　Lattice Semiconductor
デバイス型番　：　LCMXO2-7000HC-4FTG256I
インターフェイス　：　GPMC

レジスタマップ

Address	Read/Write種別	内容
08000000h - 08000001h	R	システム予約エリア
08000002h	R	ロータリースイッチ
08000003h	R	DIP Switch
08000004h	R	デバイス接続台数
08000005h	R/W	システム予約エリア
08000006h	R/W	LED制御
08000007h	R	未使用
08000008h - 0800005fh	R/W	システム予約エリア
08000060h - 08000061h	R/W	DIO制御レジスタ
08000063h - 080000FFh	R/W	未使用
08000100h - 080001FFh	R	デバイス0
08000200h - 080002FFh	R	デバイス1
: :		: :
08000F00h - 08000FFFh	R	デバイス14
08001000h - 080010FFh	R	デバイス15

Rotary Switch Register (08000002h)　R

ロータリースイッチの状態を取得します。

Rotary Switch Register

D7	D6	D5	D4	D3	D2	D1	D0
GROUP ID				UNIT ID			

DIP Switch Register (08000003h)　R

DIP Switchの状態を取得します。

0xFFなど下位4ビットが0や0xFの値が読み出される場合は、故障の可能性があります。

System Status / DIP Switch Register

D7	D6	D5	D4	D3	D2	D1	D0
DIP Switch				System Status			
SW4	SW3	SW2	SW1				

I/O Module Information Register (08000004h)　R

接続するデバイスの接続台数を取得します。

I/O Module Information Register

D7	D6	D5	D4	D3	D2	D1	D0
-		I/O Module Num					

LED Control Register (08000006h)　R/W

LEDを制御します。

Table 1 LED Control Register

D7	D6	D5	D4	D3	D2	D1	D0
-				ERR LED	ST2 LED	ST1 LED	PWR LED

[PWR LED]
0:　ON
1:　OFF
[ST1 LED]
0:　OFF
1:　ON
[ST2 LED]
0:　OFF
1:　ON
[ERR LED]
0:　OFF
1:　ON

DIO Control Register (08000060h)　R/W

DIOのDirectionを設定します。

DIO Control Register

D7	D6	D5	D4	D3	D2	D1	D0
-				DIO Direction			
				DIO4	DIO2	DIO1	DIO0

[DIO Direction]
0:　Input
1:　Output

DIO Value Register (08000061h)　R/W

DI/DOの状態取得、DOの出力設定をします。

DIO Value Register

D7	D6	D5	D4	D3	D2	D1	D0
DO Value				DI Value			
DIO3	DIO2	DIO1	DIO0	DIO3	DIO2	DIO1	DIO0

4. 内蔵NOR FLASHメモリマップ

CONPROSYSは32Mbyteの内蔵NOR FLASHメモリを搭載しています。

メモリの配置と『**内蔵NOR FLASHインストール用rootfsセクションへのコピー (P38)**』の項で示すインストールファイルの関係を次に表します。

NOR FLASHメモリマップ

Address	dev	メモリサイズ	用途	インストールファイル
0000000h - 001FFFFh	mtd0	131,072 byte	マスターブート用	MLO.byteswap
0020000h - 009FFFFh	mtd1	524,288 byte	u-boot用	u-boot.img
00A0000h - 00DFFFFh	mtd2	262,144 byte	DeviceTree用※1	MC341Bxx_1lan.dtb MC341Bxx_2lan.dtb
00E0000h - 043FFFFh	mtd3	3,538,944 byte	カーネル用	uImage
0440000h - 0DBFFFFh	mtd4	9,961,472 byte	ramdisk用	ramdisk.xz
0DC0000h - 1FFFFFFF	mtd5	19,136,512 byte	アプリケーション領域	mtd5.tgz※2

※1 DeviceTreeファイルが格納されています。

※2 mtd5へは、ファイルを展開した状態でインストールします。

5. コンパクトタイプシリーズ LED/DIP Switch/Switch制御

コンパクトタイプのLEDは次の表に示すものをGPIOポートで制御することができます。

コンパクトタイプLED制御

LED種別	制御デバイス	ポートNo	ポート属性	制御方法 (linux shell)
Power	GPIO	67	Out	On : /usr/local/bin/gpio_out.sh 67 0 Off : /usr/local/bin/gpio_out.sh 67 1
ST1	GPIO	26	Out	On : /usr/local/bin/gpio_out.sh 26 0 Off : /usr/local/bin/gpio_out.sh 26 1
ST2	GPIO	27	Out	On : /usr/local/bin/gpio_out.sh 27 0 Off : /usr/local/bin/gpio_out.sh 27 1

コンパクトタイプのSwitchは次の表に示すものをGPIOポートから読み出すことができます。

コンパクトタイプSwitch制御

LED種別	制御デバイス	ポートNo	ポート属性	制御方法 (linux shell)
DIP SW1-2	GPIO	32	In	/usr/local/bin/gpio_in.sh 32 On=0, Off=1
DIP SW1-3	GPIO	33	In	/usr/local/bin/gpio_in.sh 33 On=0, Off=1
DIP SW1-4	GPIO	34	In	/usr/local/bin/gpio_in.sh 34 On=0, Off=1
Shutdown SW	GPIO	35	In	/usr/local/bin/gpio_in.sh 35 Press(On)=0, Release(Off)=1

6. スタックタイプシリーズ DIO/LED/DIP Switch/Switch制御

スタックタイプのDIO / LED / DIP Switch / Switch は、CONPROSYS上の下記ディレクトリ下にあるファイルによって制御することができます。

/sys/bus/platform/drivers/cps-driver

各ファイルの機能と使用方法を『**スタックタイプDIO / LED / DIP Switch / Switch制御 (P89)**』に示します。

スタックタイプDIO / LED / DIP Switch / Switch制御

ファイル	制御デバイス	機能
	使用方法	
dio0_direction	DIO	DI/DOの切り換え設定
	b0(DIO0) - b3(DIO3)を0ならDI、1ならDOに設定 設定例： DIO0とDIO1をDI、DIO2とDIO3をDOに設定 b3:1, b2:1, b1:0, b0:0 → cH <Command> echo 0xc > /sys/bus/platform/drivers/cps-driver/dio0_direction 設定読み出し例： <Command> cat /sys/bus/platform/drivers/cps-driver/dio0_direction	
dio0_do_value	DO	DO値設定
	設定例： DO0とDO2を1、DO1とDO3を0に設定 b3:0, b2:1, b1:0, b0:1 → 5H <Command> echo 0x5 > /sys/bus/platform/drivers/cps-driver/dio0_do_value 設定読み出し例： <Command> cat /sys/bus/platform/drivers/cps-driver/dio0_do_value	
dio0_di_value	DI	DI値読み出し
	<Command> cat /sys/bus/platform/drivers/cps-driver/dio0_di_value	
id	ロータリースイッチ	ロータリースイッチ値読み出し
	<Command> cat /sys/bus/platform/drivers/cps-driver/rotary_id	
led_status1	Status1 LED	Status1 LED On/Off設定
	設定例： Status1 LED をOn <Command> echo 1 > /sys/bus/platform/drivers/cps-driver/led_status1 設定読み出し例： <Command> cat /sys/bus/platform/drivers/cps-driver/led_status1	
led_status2	Status2 LED	Status2 LED On/Off設定
	設定例： Status2 LED をOff <Command> echo 0 > /sys/bus/platform/drivers/cps-driver/led_status2 設定読み出し例： <Command> cat /sys/bus/platform/drivers/cps-driver/led_status2	
led_error	Error LED	Error LED On/Off設定
	設定例： Error LED をOn <Command> echo 1 > /sys/bus/platform/drivers/cps-driver/led_error 設定読み出し例： <Command> cat /sys/bus/platform/drivers/cps-driver/ switch	
switch	DIP Switch	DIP Switch値読み出し
	<Command> cat /sys/bus/platform/drivers/cps-driver/switch	

7. オプションボード制御

下記モデルにおいては、3G/LTE/920Hz通信のオプションボードが本体に内蔵されています。

【コンパクトタイプ　M2Mコントローラシリーズ】

CPS-MC341G-ADSC1シリーズ　マルチI/O + 3G(日本国内 / グローバル)モデル
CPS-MC341Q-ADSC1　マルチI/O + 920MHz帯通信モデル

【コンパクトタイプ　M2M Gatewayシリーズ】

CPS-MG341G-ADSC1シリーズ　マルチI/O + 3G(日本国内)モデル
CPS-MG341G5-ADSC1　マルチI/O + LTEモデル

【スタックタイプ　M2Mコントローラシリーズ】

CPS-MCS341G-DS1　CPUモジュール + 3G(日本国内)モデル
CPS-MCS341G5-DS1　CPUモジュール + LTEモデル
CPS-MCS341Q-DS1　CPUモジュール + 920MHz帯通信モデル

【スタックタイプ　M2M Gatewayシリーズ】

CPS-MGS341G5-DS1　CPUモジュール + LTEモデル

これらのモデルは、オプションボードの電源を制御することができます。

オプションボード制御

機能	制御方法 (linux shell)
オプションボード電源On※	/usr/local/cps-board/PowerOnOptionBoard.sh
オプションボード電源Off※	/usr/local/cps-board/PowerOffOptionBoard.sh
オプションボード検知	/usr/local/cps-board/DetectOptionBoard.sh [終了ステータス] 0: オプションボード起動中 1: オプションボード未検知

※ root権限が必要です。コンソールで実行する場合はsudoコマンドを用いて実行してください。

3G/LTEモデルは、接続/切断、SIMチェック、RSSI取得等を制御することができます。

3G/LTE制御

機能	制御方法 (linux shell)
接続※1	/usr/local/cps-board/mobile/start_mobile.sh
切断※1	/usr/local/cps-board/mobile/stop_mobile.sh
3G/LTEモジュールリセット※1	/usr/local/cps-board/mobile/reset_mobile.sh
SIMチェック	/usr/local/cps-board/mobile/checkSIM_mobile.sh [終了ステータス] 0： SIMあり “Detect SIM”表示 1： SIMなし “Not Detect”表示
RSSI取得	/usr/local/cps-board/mobile/checkSIM_mobile.sh [終了ステータス] 0： 成功 RSSI値(dbm)表示 1： 失敗
RSRP取得(LTEモデルのみ)	/usr/local/cps-board/mobile/getRSRP.sh [終了ステータス] 0： 成功 RSRP値(dbm)表示 1： 失敗
オプションボードのLED制御※2	/usr/local/cps-board/mobile/ctrl_LED.sh param [param] 0: All off 1: Green On Red Off 2: Green Off Red On 3: Green On Red On [終了ステータス] 0: 成功 1: 失敗

※1 root権限が必要です。コンソールで実行する場合はsudoコマンドを用いて実行してください。

※2 CPS-MC341G-ADSC1-111およびCPS-MG341G-ADSC1-111のモデルについて、3Gモジュールが制御するためLED制御を行うことはできません。

8. ターゲット搭載アプリケーション

主な搭載アプリケーション

Application	Light rootfs NOR Flash version	Light rootfs SD version	Ubuntu20.04 with SDK
busybox	1.31.1	1.31.1	-
apt-utils	-		2.0.6
binutils	-		2.34
ncurses	-		6.2
apache	2.4.29※1	2.4.29※1	2.4.41
ssh server/client	dropbear 2019.78	dropbear 2019.78	OpenSSH_8.2p1
NTP client	(busybox)	(busybox)	ntpdate 4.2.8p12
DHCP server	(busybox)	(busybox)	isc-dhcp-server 4.4.1
DHCP client	(busybox)	(busybox)	isc-dhcp-client 4.4.1
Samba server	-	-	4.13.14-Ubuntu
Samba client	-	-	4.13.14-Ubuntu
Nfs Server	-	-	-
Nfs Client	-	-	-
gcc / g++	-	-	9.3.0
cmake	-	-	3.16.3
autoconf	-	-	2.69
automake	-	-	1.16.1
perl	-	-	v5.30.0
python	-	-	3.8.10
php	5.6.34※1	5.6.34※1	7.4.3
curl	7.59.0※1	7.59.0※1	7.68.0
wget	(busybox)	(busybox)	1.20.3
ftp server	(busybox)	(busybox)	vsftpd 3.0.3
ftp client	(busybox)	(busybox)	-
tftp server	(busybox)	(busybox)	-
tftp client	(busybox)	(busybox)	-
mail	(busybox)	(busybox)	-
iperf	-	-	-
minicom	-	-	-
ppp	2.4.7	2.4.7	2.4.7
pppoe	-	-	3.12
iptables	1.8.4	1.8.4	v1.8.4
Wireless tool	29※1	29※1	30
wpa_supplicant	2.7※1	2.7※1	v2.9
Open SSL	1.0.2n※1	1.0.2n※1	1.1.1f
sudo	1.8.31p1	1.8.31p1	1.8.31
gdb	-	-	9.2

※1 オプション選択

改訂履歴

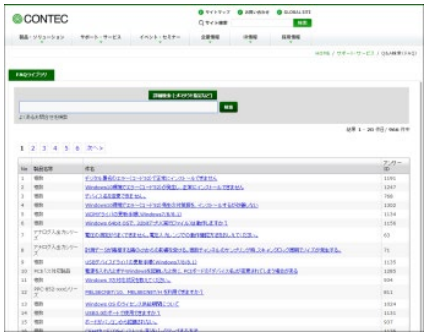
改訂日	改訂内容
2022年2月	初版
2023年9月	Ver 2.1.0 - 対応機種追加 CPS-MCS341-DSxシリーズ（スタックモデル） - 対応I/Oモジュール追加（スタックモデル用） CPS-COM シリーズ CPS-AI シリーズ CPS-AO シリーズ CPS-DIO シリーズ CPS-DI シリーズ CPS-DO シリーズ CPS-RRY シリーズ CPS-SSI シリーズ
2024年11月	Ver 2.2.0 - 対応機種追加 CPS-MGS341-DS1 CPS-MGS341G5-DS1 - 対応I/Oモジュール追加（スタックモデル用)] CPS-SSI-4C

よくあるご質問（FAQ検索）

FAQライブラリ

https://www.contec.com/jp/tsc/

お客さまからよく寄せられるお問い合わせ内容を「Q&A」形式でご覧いただけます。
製品やサービスに関する疑問やお困りごとの解決にお役立てください。



株式会社コンテック　〒555-0025　大阪市西淀川区姫里3-9-31

https://www.contec.com/

CONPROSYS Linux SDKユーザーズマニュアル(クロスビルド版)

NA08763 (LXAU772)　[11222024]　　2024年11月改訂