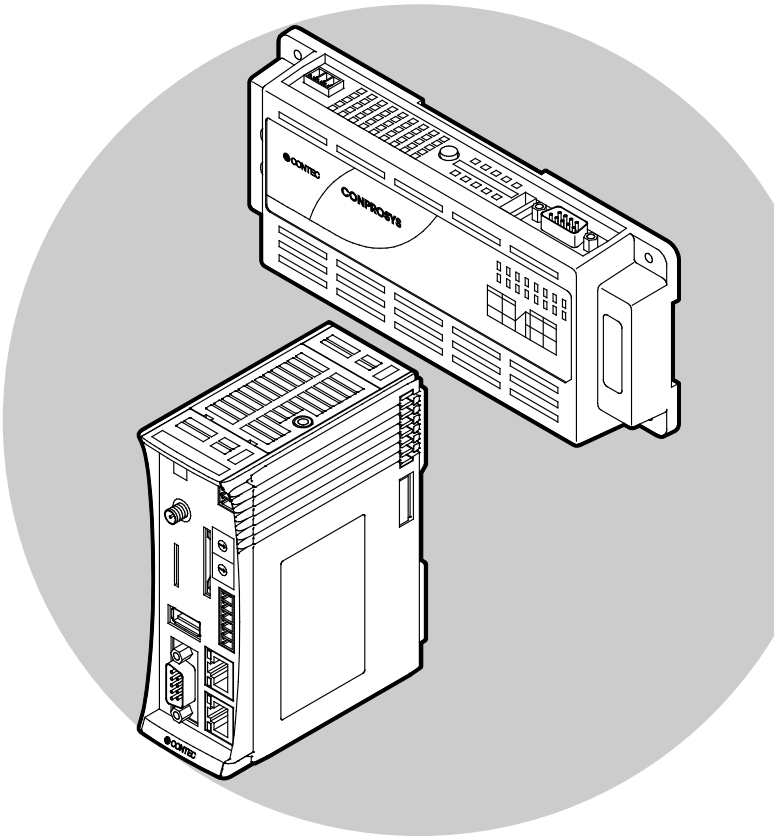
CONTEC

CONPROSYS®

用户手册

（交叉开发版）

CONPROSYS Linux SDK Ver. 1.5.0



目 录

CONTEC CO., LTD.

目录

目录

前言

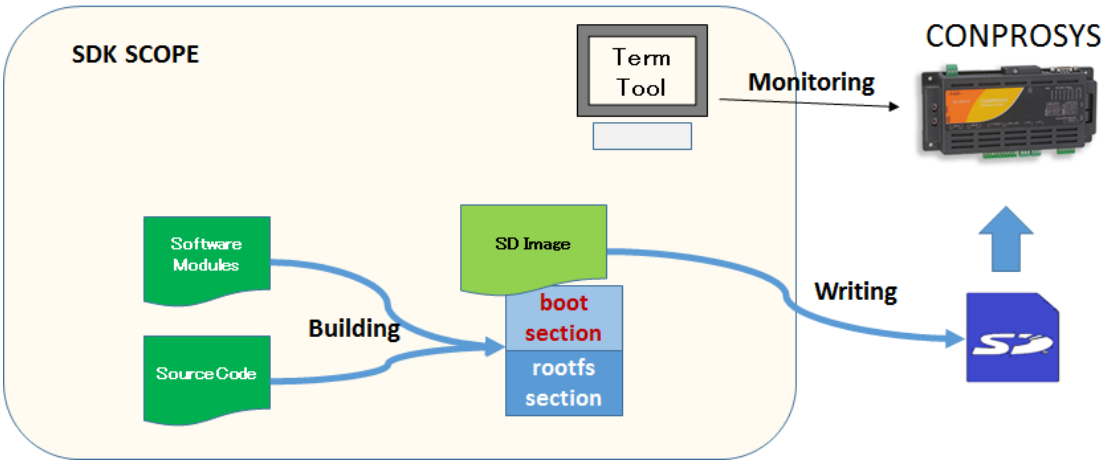
1. 概要

CONPROSYS Linux SDK(Software Development Kit) 是一个软件开发环境。交叉开发版在开发主机上开发在CONPROSYS上运行的软件。

本SDK包含以下内容。

- 用于生成CONPROSYS运行软件的开发主机上的工具
  (源代码(kernel、库、驱动程序等)和开发脚本等)
- 用于将CONPROSYS软件写入到SD卡的开发主机电脑上的工具
- 用于监视CONPROSYS上软件运行状况的工具
  (串行控制台等)

SDK范围(SCOPE)



本SDK在开发主机上，通过交叉开发生成CONPROSYS软件模块。

如果想在CONPROSYS上进行本机开发，请参阅《本机开发版SDK手册》。

用交叉开发板SDK中可以生成本机开发版SDK。有关详细信息，请参阅**《编译(P59)》**之后的章节。

2. 对应的CONPROSYS产品一览

本SDK对应的产品如下所示：

【精巧一体型　M2M控制器系列】

CPS-MC341-ADSCx系列	多功能I/O型
CPS-MC341G-ADSC1系列	多功能I/O + 3G(日本国内 / 全球) 型
CPS-MC341Q-ADSC1	多功能I/O + 920MHz段通信型
CPS-MC341-A1	模拟量输入输出型
CPS-MC341-DSx系列	数字量输入输出型
CPS-MC341-DS11	数字量输入输出型

【精巧一体型　M2M Gateway系列】

CPS-MG341-ADSC1系列	多功能I/O型
CPS-MG341G-ADSC1系列	多功能I/O + 3G型
CPS-MG341G5-ADSC1	多功能I/O + LTE型

【堆栈组合型　M2M控制器系列】

CPS-MCS341-DS1系列	CPU模块
CPS-MCS341G-DS1	CPU模块 + 3G型
CPS-MCS341G5-DS1	CPU模块 + LTE型
CPS-MCS341Q-DS1	CPU模块 + 920MHz段通信型

【堆栈组合型 M2M Gateway系列】

CPS-MGS341-DS1	CPU模块
CPS-MGS341G5-DS1	CPU模块 + LTE型

※M2M控制器系列搭载的HMI、VTC、OPC-UA、Modbus等功能，
　CONPROSYS Linux SDK未支持，需要另外安装软件。
※M2M Gateway系列搭载的HMI、VTC、OPC-UA、Modbus、PLC、CNC等功能，
　CONPROSYS Linux SDK未支持，需要另外安装软件。
※PAC系统系列、nano系列未对应。

3. 软件使用许可协议

本协议是客户与株式会社康泰克(以下简称“本公司”)之间，就本协议书关联的本公司软件程序(以下称为“本软件”)的使用许可达成协议。客户下载、安装或使用本软件，或使用已安装了本软件的机器，即表示客户同意本协议的各项条款，本协议随即成立并生效。在不同意本协议的情况下，不能下载、安装或使用本软件，或者使用安装了本软件的机器。

第1条（知识产权）

本软件和手册等附属的文档及其复制物(以下称为“本软件等”。)的著作权、专利权及其他知识产权属于本公司所有，客户除了本协议中明确授权的内容外，没有其他任何权利。

第2条（使用许可）

1. 本公司授予客户以使用本公司硬件为目的安装和使用本软件的非独占许可。
2. 客户仅可以紧急备份为目的而复制此软件，并仅限于使用本软件所需的最小份数。复制品具有与本公司提供的本软件相同的权利。
3. 客户可以将本公司提供的作为程序库的软件编入客户制作的软件中。

第3条（使用限制）

客户不得进行以下行为：

(1) 本协议规定以外的由本软件派生的软件的制作
(2) 本协议规定以外的本软件的复制
(3) 本软件的修改、改编、反编译、反汇编、及其他逆向工程
(4) 本软件的权利表示、商标等的删除或修改

第4条（免责）

1. 本公司对本软件不作任何保证。
2. 本公司对因下载、安装、使用或利用本软件后发生硬件或数据的故障，即使是因本软件引起或关与本软件有关而导致损害的场合，本司也一概不承担责任。对于复制软件、嵌入或修改了本软件以及使用或利用这些软件制作的软件也同样适用。

第5条（转让）

1. 客户只有满足了以下的全部规定的条件，才可以将本软件及本协议中所许可的客户权利转让给第三方。
   (1) 与本协议一起将本软件等全部转让给该第三方
   (2) 将下载了本软件的本公司的硬件产品整体转让给该第三方
   (3) 受让方同意本协议的条件
2. 根据前项的规定进行了本软件及权利转让的场合，受让方从接受转让时开始受本协议约束。

第6条（协议解除）

1. 如客户不遵守本协议的各项条款，本公司可以立即终止本协议，而不必向客户进行任何通知和督促。
2. 在本协议终止时，授权给客户的使用许可将全部作废。客户应立即停止使用本软件，卸载本软件，并删除所有复制品。

第7条（关于物理缺陷）

1. 如果存储本软件等的记录介质中存在妨碍使用本软件等的物理缺陷，本公司应在客户收到本软件等之日起 30 天内通过您购买的销售店更换记录介质。

第8条（关于软件程序的信息）

1. 有关本软件的各种信息和更新程序应在我们的网站上提供。
2. 上述信息和更新程序将根据本协议授予客户。客户可以根据需要自己判断使用这些信息和更新程序，但在这种情况下，该信息和更新程序也必须遵守本协议的条款。

第9条（出口管制）

1. 将本软件等带出外国时，客户必须遵守日本《外汇及对外贸易法》、《美国出口管理法》及其他国家的法令。
2. 客户不得将本软件等转让、出口或再出口给有可能用于核武器、生物化学武器设计、开发、制造或导弹设计、开发、制造的个人或法人。
3. 不得将本软件等转让、出口、再出口给下列各款规定的国家、地区、个人或法人。
   (1) 古巴、伊朗、伊拉克、利比亚、朝鲜
   (2) 基于出口贸易管理令的“外国用户列表”或美国商务部的“Denied Persons List”中记载的个人或法人
   (3) 日本国政府、美国政府及其他有关国家政府指定的国家、地区、个人或法人

第10条（适用法律）

本协议将按照日本国法律理解并解释。

第11条（管辖权的合意）

关于本协议或者本软件产生纠纷，需要提出诉讼等法律程序的情况下，双方同意大阪简易裁判所或者大阪地方裁判所作为双方同意的一审法院。

第12条（条款的分离）

即使本协议的一部分条款被视为无效或者丧失了法律强制力，也不会对其他条款产生影响，其他各条款仍然是有效的，在法律允许的范围内具有法律强制力。

为了安全使用

1. 注意记号的说明

在本书中，为了避免人身事故和机器的损坏，按如下符号提供有关的安全信息。
应认真理解内容，并安全操作机器。

⚠ 危险	表示【有可能导致人员死亡或重伤等严重后果，并且重要程度很高的内容】。
⚠ 警告	表示【有可能导致人员死亡或重伤等严重后果的内容】。
⚠ 注意	表示【有可能导致人员负伤或财产损失等后果的内容】。

2. 操作注意事项

⚠ 注意

- 本产品或本书为了增加功能、提高质量可能会更改规格，恕不另行通知，即使持续使用时，也请务必阅读本公司主页中的手册，确认内容。
- 请不要改造本产品。
  对于已改造的产品，本公司概不负责。
- 以本产品的使用为由而要求的损失、利润损失赔偿等，无论前提如何，本公司概不负责，敬请谅解。

3. 安全注意事项

连接网络时，应在考虑存在的安全风险的基础上，参考安全对策案例，正确地设置主机及相关网络设备。

1. 安全风险

- 系统因外部网络入侵而中断、数据损坏、信息窃取或感染恶意软件 *1。
- 入侵后以该机器为踏板，对外部网络的攻击。(从受害者变成加害者)
- 与外部网络连接相关的意外信息泄露。
- 这些事故的连带损害包括声誉损害、损害赔偿责任、信誉损失和机会损失。

*1…恶意软件（Malicious Software)：恶意程序。是执行用户不希望动作的程序

2. 安全措施示例

- 更改初始密码。(密码的设置方法，请参阅所使用的说明书、手册)
- 设置保密强度高的密码。

包括半角英文小写字母、大写字母、数字等，组成难以类推的组合。

- 定期更改密码。
- 停止（禁用）不需要的网络服务或不需要的功能。
- 限制网络连接设备的网络访问源。*2
- 限制网络连接设备的网络开放端口。 *2
- 使用封闭网络（如专用网络或 VPN*3）构建网络。

*2…有关设置方法，请与网络设备的制造商联系。

*3…VPN（虚拟专用网络)：是通过身份验证和加密保护通信路径，防止第三方进入的安全网络。

非法访问的手段和漏洞(安全漏洞)，不断出现，没有完美的防止手段。在理解了网络连接时常伴随着危险的同时，强烈推荐经常获取新的信息，进行安全对策。

开发环境

1. 开发所需的设备

- CONPROSYS主机(Linux)
- SDHC卡 (2Gbyte以上。SDXC未对应)
- USB串口转换电缆 (推荐电缆： FTDI产 TTL-232R-3V3-AJ)
- 网线

2. SDK规格

开发用主机P电脑 Linux Distribution:	Ubuntu 14.04 / 16.04 (64bit版) Desktop HDD必须有40Gbyte以上可用空间 可以运行sudo的管理员权限用户
目标Kernel version:	3.2.0
目标Distribution:	arm版 Ubuntu 14.04 (只是加载SD用)
交叉编译GCC version:	gcc 4.9 (Hardware float) / gcc 4.7 (Software float)

必要的Linux工具链：

apt, gcc-arm-linux-gnueabi, libncurses5-dev, gawk, u-boot-tools, openssh-server, samba, binutils-arm-linux-gnueabi, binutils-arm-linux-gnueabihf, xinetd, kpartx, gperf, bison, flex

※上面列出了运行本SDK所需的条件。
如果各个开发环境中有其他需要的软件包，请另行获取并安装。
(例：git, wget, subversion等)

3. SDK内容

- SDK文档
- 交叉编译器/工具链

  软件包（Debian/Ubuntu用）：

  gcc-arm-linux-gnueabihf-4.9.3, gcc-arm-linux-gnueabi-4.7, libncurses5-dev, gawk,

  u-boot-tools, openssh-server, samba, binutils-arm-linux-gnueabi,

  binutils-arm-linux-gnueabihf, xinetd, kpartx, gperf, bison, flex
- 编译工具
- 源代码

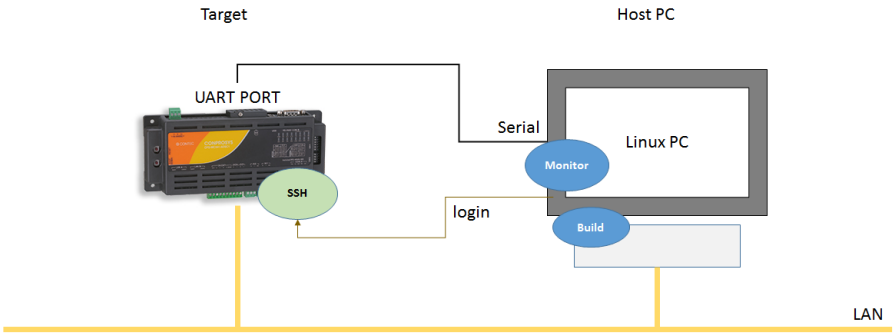
  u-boot, kernel, 应用示例, 库示例, 驱动程序示例
- 各个CONPROSYS产品的基本模块(u-boot, kernel, 设置等)

4. 开发环境构成

主机电脑(编译、调试用)和目标Conprosys的构成如下所示：

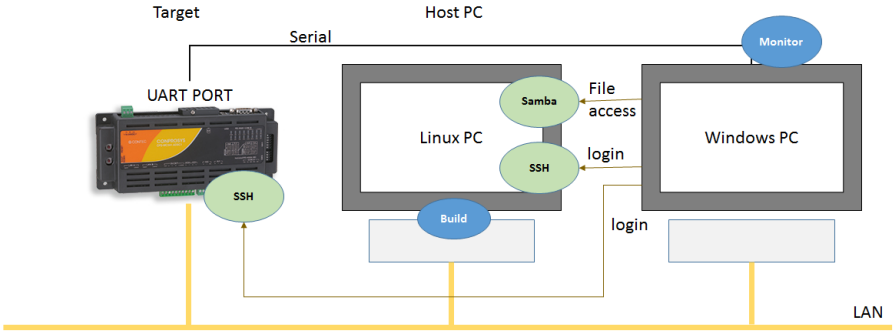
例1） 在1台开发主机上编译并对目标通过串口进行调试

用1台Linux电脑进行编译并通过串口调试



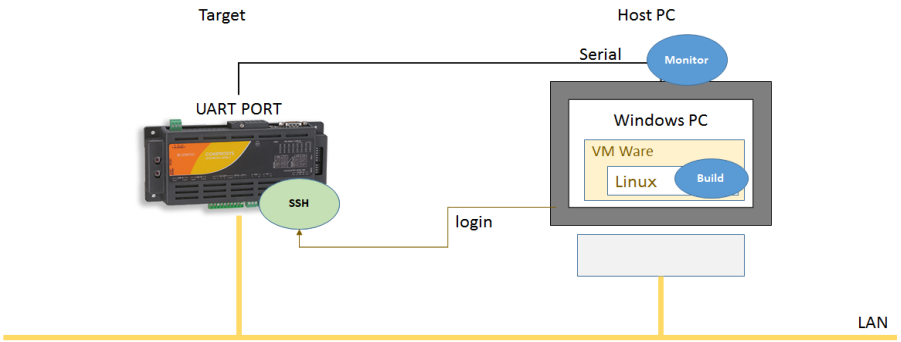
例2） 在开发主机上编译(或编辑源代码)，在另1台Windows电脑通过串口进行调试

用Linux电脑作为编译用，用Windows电脑通过串口对目标进行调试



例3） 在Windows系统中安装虚拟机(VM Ware，Virtual BOX等)，并在虚拟机中安装Linux系统，作为开发主机电脑使用

在一台电脑上，使用Linux(利用VM Ware)用于编译，使用Windows通过串口进行调试



5. SDK的安装

请准备好下载的SDK文件或DVD。

◆ 下载的为tgz文件：

1 解压下载的tgz文件

```
tar xvfz CPS_SDK_installer_xxxx.tgz[ -C 解压目标文件夹]
```

2 进入解压目标文件夹。
※当前已在解压文件夹时，本步骤省略。

◆ 下载的为iso文件：

1 加载下载的iso文件
加载的目标文件夹请任意作成。

```
sudo -E mount  -o loop CPSSDK_xxxx.iso 加载目文件夹
```

2 进入加载目标文件夹。

◆ 使用DVD时：

1 将DVD光盘插入主机开发电脑中。

2 插入的DVD光盘将自动加载，并进入到该加载目标文件夹。

1.SDK所需的工具链的安装

◆ 主机电脑连接互联网时

通过apt-get命令在ubuntu OS上安装以下工具链。

libncurses5-dev, gawk, u-boot-tools, openssh-server, samba, binutils-arm-linux-gnueabi, binutils-arm-linux-gnueabihf, xinetd, kpartx, gcc-4.7-arm-linux-gnueabi, gperf, bison

在安装工具链之前，请先更新apt-get的软件包列表。

apt-get的软件包列表更新命令：

```
sudo apt-get update
```

安装命令：

```
sudo apt-get install libncurses5-dev gawk u-boot-tools openssh-server samba \
  binutils-arm-linux-gnueabi binutils-arm-linux-gnueabihf xinetd kpartx gperf \
  bison flex
```

编译器安装CONPROSYS linux SDK附带的软件包。

安装命令：

```
cd Toolchain
sudo ./compiler_pkginstall.sh
cd ..
```

◆ 主机电脑未连接互联网时

CONPROSYS linux SDK提供了所需的工具链包。请进入到[Toolchain]文件夹，执行toolchain_pkginstall.sh。（按《**CONPROSYS linux SDK的安装 (P20)**》中的说明，用./install_sdk.sh也能安装。）

命令：

```
cd Toolchain
sudo ./toolchain_pkginstall.sh
cd ..
```

2.CONPROSYS linux SDK的安装

用以下命令开始安装SDK。

命令：

```
./install_sdk.sh [ -C 安装文件夹] [-t]
```

选项：

-C 安装文件夹

创建指定的安装文件夹，安装到该文件夹下。

-t

为SDK安装所需的交叉编译器等工具链。

如果指定此选项，则需要管理员密码才能将工具链安装到开发主机电脑上。

※ 如果未指定要安装的文件夹，则在当前文件夹下自动生成名为“CPS_SDK”的文件夹，并将其安装在该文件夹中。加载iso文件或DVD介质时，因无法在当前文件夹下创建文件夹，所以请务必指定安装目标文件夹。

※ 安装文件夹建议在登录用户的home文件夹下。

命令示例：

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

安装后的文件夹构成如下所示：

安装后的文件夹构成

```
~/
 |
 +-- [CPS_SDK]
       |
       +-- [Document]     文档
       |
       +-- [application]  应用源代码文件夹
       |
       +-- [base]         缺省文件保存文件夹
       |
       +-- [driver]       驱动程序源代码文件夹
       |
       +-- [kernel]       kernel源代码文件夹
       |
       +-- [lib]          库源代码文件夹
       |
       +-- [ramdisk]      ramdisk(NOR Flash用)作成文件夹
       |
       +-- [rootfs]       rootfs源代码文件夹
       |
       +-- [tools]        工具汇总
       |
       +-- [u-boot]       u-boot源代码文件夹
       |
       +-- [target]       Target文件夹
```

[Document]

是保存SDK文档文件的文件夹。

[application]

是保存应用源代码的文件夹。

[base]

是保存作为target基础的boot部分和rootfs部分的文件夹。

[tools]

是SDK的工具汇总。

[driver]

是保存驱动程序源代码的文件夹。

[kernel]

是保存kernel源代码的文件夹。

[lib]

是保存库源代码的文件夹。

[ramdisk]

是ramdisk的文件夹，内容和要安装在NOR Flash中的rootfs相同。

[rootfs]

是保存用于内置NOR Flash引导等的轻量版rootfs(Root File System)源代码的文件夹。

[u-boot]

是保存u-boot源代码的文件夹。

[target]

是为各个CONPROSYS产品生成用于SD卡启动部分的文件夹。configure.sh运行后，生成目标用文件夹，作为编译模块 （boot，kernel，driver，application） 的保存目标。

交叉开发环境设置

1. SD卡制作流程

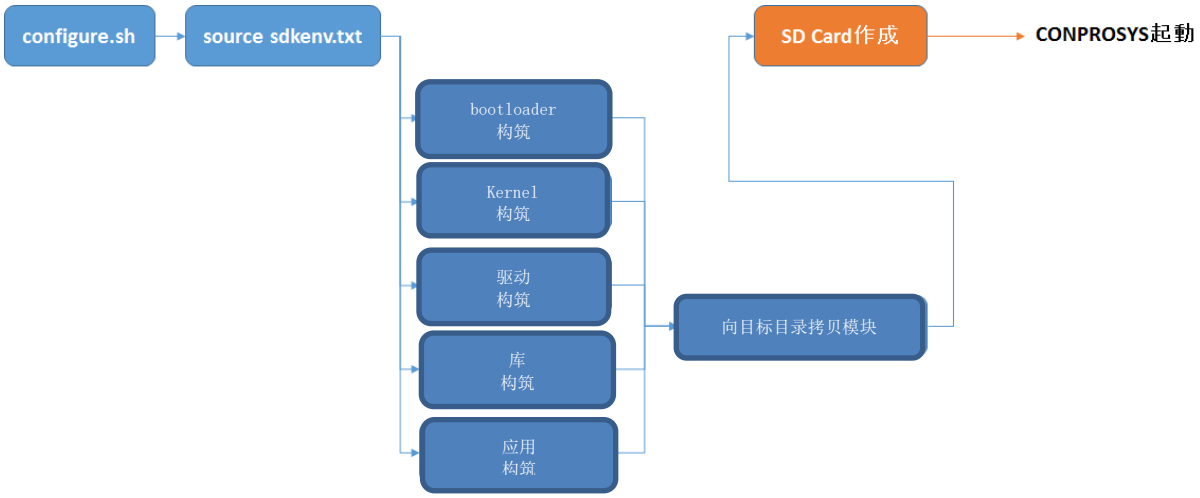
从模块的生成到启动SD卡制作的流程如下所示。

◆ 对于不进行交叉编译的目标，例如本机开发版CONPROSYS Linux SDK

本机开发版CONPROSYS Linux SDK的SD卡制作流程

◆ 对于进行交叉编译的目标

包含交叉编译的SD卡制作流程



关于SD卡的制作请参阅<启动用SD卡的制作 (P31)>。

关于各种编译请参阅<编译(P59)>。

2. 初始设置

在SDK安装目标文件夹下执行./configure.sh并执行初始配置以进行编译。通过执行此命令生成以下环境文件和文件夹。这个命令只需要在生成新的目标模块时才使用。

- 用于编译的环境设置文件(sdkenv.txt)
- Kernel的.config文件
- target 文件夹下的用于写入对应设备SD卡的文件系统
  （“boot”部分，“rootfs”部分）

命令：

```
./configure.sh
```

※中途为了执行root权限的命令可能需要密码。
请输入密码进行处理执行。

运行命令后便进入输入目标CONPROSYS信息的模式，请按照菜单输入相应的号码。

CONPROSYS Product:
用编号输入目标产品。

1) CPS-MC341-ADSCx　精巧一体型 多功能I/O型
CPS-MC341-ADSCx系列, CPS-MC341G-ADSC1系列(3G型),
CPS-MC341Q-ADSC1(920MHz段通信型),
CPS-MG341-ADSC1系列, CPS-MG341G-ADSC1系列(3G型),
CPS-MG341G5-ADSC1(LTE型)

2) CPS-MC341-Ax　精巧一体型 模拟量输入输出型
CPS-MC341-A1

3) CPS-MC341-DSx　精巧一体型 数字量输入输出型
CPS-MC341-DSx系列

4) CPS-MC341-DS1x　精巧一体型 数字量输入输出型(带USB口)
CPS-MC341-DS11

5) CPS-MxS341-DSx　堆栈组合型
CPS-MxS341-DS1系列,
CPS-MCS341G-DS1(3G型), CPS-MxS341G5-DS1(LTE型),
CPS-MCS341Q-DS1(920MHz频段通信型)

LAN type：

用编号输入网口的设置类型。

1) 1lan (HUB模式) (单网口) LAN A口和LAN B口使用一个网口设定，可作为HUB使用。

2) 2lan (双网口) LAN A口和LAN B口作为独立网口使用。

本机开发版CONPROSYS Linux SDK，LAN A口和电脑直接连接用于调试，LAN B口和互联网连接，此时请选择2lan。

Rootfs type：

用编号输入目标的rootfs类型。选项3是附带自编译程序的SDK，可以在CONPROSYS上开发软件。

1) light (busybox) 轻量版rootfs

2) Ubuntu 14.04 Ubuntu 14.04

3) Ubuntu 14.04 (include SDK) Ubuntu 14.04 附带SDK(本机开发版CONPROSYS Linux SDK)

Tool choice：

如果前项选择了轻量版rootfs，则可以选择搭载工具。请用编号输入要搭载的工具的类型。

1) Wireless tools, Apache 2.4, PHP5 Wireless工具, Apache 2.4, PHP5

0) None 不搭载工具

Cross compiler type：

如果前面选择了轻量版rootfs，则可用编号输入交叉编译器的类型。

1) gnueabi (default) arm交叉编译器

2) gnueabihf arm硬浮点对应交叉编译器

如果选择了Ubuntu14.04的rootfs时，则gnueabihf被自动选择。（不显示输入菜单）

运行configure.sh后，在target文件夹下生成target用的文件夹和基本boot部分以及在./configure中选择的rootfs部分的文件夹/文件。以下构成图是指定了CPS-MC341-ADSC系列 2 LAN type / Ubuntu14.04 with SDK时的文件夹示例。

target下的文件夹构成

```
~/
  |
  +-- [CPS_SDK]
        |
        +-- [target]                          Target文件夹
              +-- [CPS-MC341-ADSCX.2lan] CPS-MC341-ADSC(2LAN类型)用文件夹
                    +-- [boot]                boot部分
                    |      +-- MLO
                    |      +-- u-boot.img
                    |      +-- uEnv.txt
                    |      +-- uImage.
                    +-- [Ubuntu14.04_dev]      rootfs部分 (SD卡用)
                          +-- [bin]
                          +-- [boot]
                          +-- [dev]
                          +-- [etc]
                          +-- [home]
                          +-- [lib]
                          +-- [media]
                          +-- [mnt]
                          +-- [opt]
                          +-- [proc]
                          +-- [root]
                          +-- [sbin]
                          +-- [selinux]
                          +-- [srv]
                          +-- [sys]
                          +-- [tmp]
                          +-- [usr]
                          +-- [var]
```

3. 环境设置

在编译应用程序和内核等之前，请设置环境变量，用在SDK安装文件夹下由./configure.sh生成的sdkenv.txt进行设置。

命令：

```
source sdkenv.txt
```

请注意，如果没有设置此环境变量，则可能无法正常执行后述编译和Firmware写入方法等操作。

目标固件写入方法

1. 关于系统启动

目标CONPROSYS启动有下述方法：

- 从SD卡启动系统
- 从内置NOR FLASH启动系统

从SD启动系统时，请将用于SD启动的固件写入SD卡，插入SD卡并启动系统。

从内置NOR FLASH启动系统，可通过下述2个步骤：

1 制作用于向内置NOR FLASH安装系统的SD卡

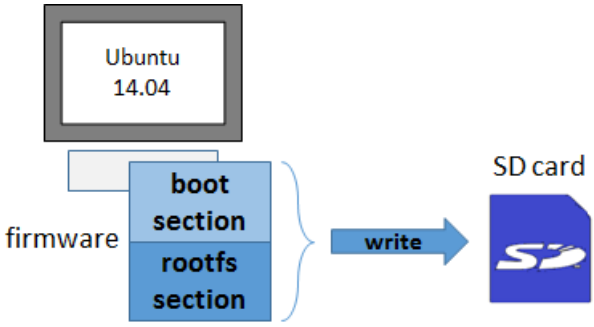
2 插入SD卡，由SD卡启动，向内置NOR FLASH安装系统

对应各种启动方法的Firmware写入方法如下所示。

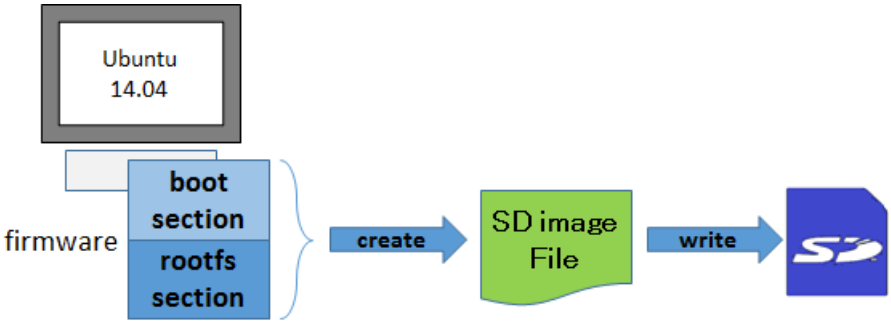
2. 启动用SD卡的制作

启动用的SD卡可按以下方法制作。

1） 向SD卡直接写入



2） 创建SD映像文件，用映像写入软件向SD卡写入



1. 向SD卡直接写入

1 在主机电脑上插入SD卡并确认能识别。
通过parted命令等检查SD卡在哪个文件系统中识别。
例）

```
sudo parted -l
```

SD卡如果已自动加载，请卸载该卡。

2 生成SD卡分区。
例）SD卡的设备是/dev/sdb时

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

用此命令生成以下两个分区。

- Boot分区　W95 FAT32 (LBA)
- Rootfs分区　ext3

3 加载SD卡。
请先生成加载上述生成的分区的目标文件夹。
下面是在/media下生成boot用和rootfs用文件夹的命令示例。

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

向这些加载目标加载上面生成的SD卡的分区。
下面是SD卡在/dev/sdb时的命令示例。

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

4 把在target下创建的boot文件夹和rootfs文件夹下的文件复制到SD卡。

[boot分区 (fat32)]

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

[rootfs分区 (ext3)]

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

5 同步复制到SD卡上的文件。

```
sync
sync
sync
```

使用sync命令同步之前，卸载并拔出了SD卡的情况下，文件可能无法正确写入SD卡中。 为了防止这种情况发生，请执行sync命令。

6 卸载SD卡，拔出SD卡。

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

2. 创建SD映像文件并用映像写入软件写入SD卡

1 创建SD映像文件。
用以下命令可创建映像文件。

命令：

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

选项：

-f filename

可指定输出的映像文件名，不指定时文件名为SD.img。

-s size

可指定输出的映像文件的大小，不指定时文件大小为2000Mbyte。

命令示例： 文件名为目标名称_ rootfs.img ，大小希望为4000 Mbyte的情况下

```
sudo -E ${CPS_SDK_ROOTDIR}/tools/mk2SDCardImage.sh ${CPS_SDK_ROOTFS} \
 -f ${CPS_SDK_TARGET_NAME}_${CPS_SDK_ROOTFS}.img -s 4000
```

2 把映像文件写入SD卡。

[Windows系统]

以使用Win32 Disk Imager写入SD卡的例子。

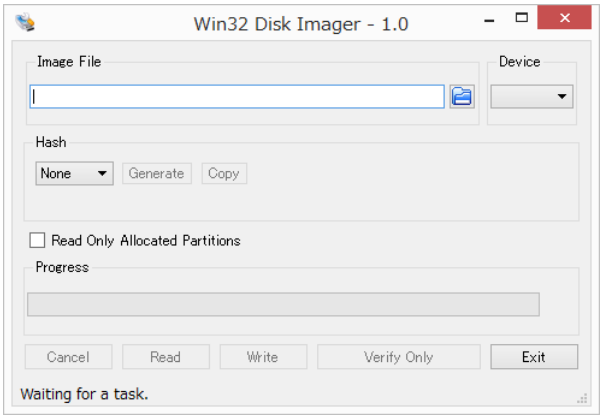
请预先从以下网站下载并安装Win32 Disk Imager安装程序到Windows 电脑。

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

i) 把SD卡插入Windows 电脑。

ii) 启动Win32 Disk Imager。

Win32 Disk Imager应用程序



iii) 选择写入的image文件。

确认Device栏中的驱动器为写入目标的SD卡，按Write按钮开始写入。

iv) 写入结束后会显示通知弹出框，请按OK按钮，拔出SD卡。

[Linux系统]

i) 如果SD卡已加载，则卸载。

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

ii) 用dd命令向SD卡写入映像文件。

```
sudo dd if=映像文件名称 of=/dev/sdb bs=1M
```

iii) 用sync命令同步。

```
sync
```

iv) 命令完成后请拔出SD卡。

3. 内置NOR FLASH启动用的安装SD卡的制作

制作内置NOR FLASH启动用的安装SD卡之前需要作以下准备。

- 创建用于内置NOR FLASH启动安装的rootfs部分
- 编译自定义的bootloader和应用程序，以及向安装用的rootfs部分复制

1. 创建用于内置NOR FLASH启动安装的rootfs部分

创建用于基础内置NOR FLASH安装的rootfs(InstallerForFlash)。

命令：

```
./create_FlashInstaller.sh
```

在执行此命令后，在目标文件夹下生成名为InstallerForFlash的文件夹。以下是在CPS-MC341-ADSC系列2 LAN type编译环境下生成的文件夹示例。

生成用于NOR Flash安装的rootfs时的文件夹构成

```
~/
 |
 +-- [CPS_SDK]
       |
       +-- [target]                          Target文件夹
             +-- [CPS-MC341-ADSCX.21an]
                   +-- [boot]                boot部分
                   +-- [Ubuntu14.04_dev]     rootfs部分 (SD卡用)
                   +-- [rootfsForInstall]    rootfs部分 (NOR Flash安装用)
                         |
                         +-- [InstallToMTD] NOR Flash安装文件文件夹
                         |       +-- MLO.byteswap   NOR Flash版引导装载器
                         |       +-- u-boot.img     NOR Flash版u-boot
                         |       +-- ramdisk.xz     ramdisk
                         |       +-- uImage         kernel
                         |       +-- mtd5.tgz       应用程序数据
                         +-- [bin]
                         +-- [dev]
                         +-- [etc]
                         +-- [home]
                         +-- [lib]
                         +-- [libexec]
                         +-- linuxrc
                         +-- [mnt]
                         +-- [proc]
                         +-- [sbin]
                         +-- [sys]
                         +-- [tmp]
                         +-- [usr]
                         +-- [var]
```

应用程序数据 mtd5.tgz

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

2. 向内置NOR FLASH安装用rootfs部分的复制

如果有自定义编译的bootloader和应用软件等，将其复制到安装用rootfs(InstallerForFlash)的安装文件夹(/InstallToMTD)中。

请在复制之前确认是否有由编译生成的文件，然后进行复制。

[bootloader]

复制由<**内置NOR FLASH启动用的编译 (P61)**>编译的MLO.byteswap u-boot.img。关于编译方法请参阅<**内置NOR FLASH启动用的编译 (P61)**>内容。

命令：

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

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

[kernel]

命令：

```
cd ${CPS_SDK_ROOTDIR}/kernel/arch/arm/boot/
```

```
cp -p uImage ${CPS_SDK_INSTALL_FULLDIR}/InstallerForFlash/InstallToMTD/\
uImage.${CPS_SDK_BOARD_NAME}.${CPS_SDK_LAN_TYPE}
```

[ramdisk]

命令：

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

```
make install
```

本SDK附属的安装程序(Shell Script)是将这四个文件(MLO.byteswap，u-boot.img，uImage，ramdisk.xz)和设备专用应用程序数据(mtd5.tgz)安装到NOR FLASH。

没有安装u-boot选项，如果想安装，请编辑下面的文件。

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

在该文件中，作为参考，以注释掉的状态记载了以下内容。

mtd2: 通过dd复制

请参考此内容自定义安装方法。

关于NOR FLASH的容量等的详细信息，请参阅<**内置NOR FLASH内存配置(P89)**>。

3. 内置NOR FLASH安装用SD卡的制作(向SD卡直接写入)

内置NOR FLASH安装用的SD卡按以下步骤制作。

基本的制作步骤和SD启动用的相同，但rootfs的复制源不同。（该rootfs的复制源为InstallerForFlash）

1 在主机电脑上插入SD卡并使其识别。
可以通过parted命令等检查SD卡在哪个文件系统中识别。
例）

```
sudo parted -l
```

2 生成SD卡分区。
例）SD卡的设备是/dev/sdb时

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

用此命令生成以下两个分区。

- Boot分区　W95 FAT32 (LBA)
- rootfs分区　ext3

3 加载SD卡。
预先生成加载上述生成的分区的目标文件夹。
下面是在/media下生成boot用和rootfs用文件夹的命令示例。

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

向这些加载目标加载上面生成的SD卡的分区。

下面是SD卡在/dev/sdb时的命令示例。

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

4 把在target下创建的boot文件夹和rootfs文件夹下的文件复制到SD卡。

[boot分区 (fat32)]

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

[rootfs分区 (ext3)]

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

5 同步复制到SD卡上的文件。

```
sync
sync
sync
```

使用sync命令同步之前，卸载并拔出了SD卡的情况下，文件可能无法正确写入SD卡上。 为了防止这种情况发生，请执行sync命令。

6 卸载SD卡，拔出SD卡。

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

4. 内置NOR FLASH安装用SD卡的制作(创建SD映像文件)

1 创建SD映像文件。
用以下命令可用创建映像文件。

命令：

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

选项：

-f filename

可指定输出的映像文件名，不指定时文件名为SD.img。

-s size

可指定输出的映像文件的大小。不指定时文件大小为2000Mbyte。

命令示例： 文件名为目标名称_InstallerForFlash.img”时

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

2 把映像文件写入SD卡。

向SD卡写入映像文件的方法请参阅<**创建SD映像文件并用映像写入软件写入SD卡(P34)**>。

4. 向内置NOR FLASH的安装

在目标CONPROSYS上启用SD boot并启动在<内置NOR FLASH安装用SD卡的制作(向SD卡直接写入)(P39)>或<内置NOR FLASH安装用SD卡的制作(创建SD映像文件)(P41)>中制作的SD卡，自动开始写入NOR FLASH。

关于启动方法，请参阅<从SD卡启动 (P44)>内容。

安装过程中ST1(绿色)、ST2(红色)和Power(绿色)LED闪烁，正常结束后ST1(绿色)和Power(绿色)LED将保持点亮。

目标动作确认

1. 目标启动方法

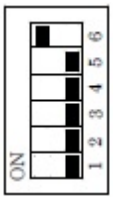
1. 从SD卡启动

请确保目标CONPROSYS的DIP SW的SD boot模式都已启用。

◆ 精巧一体型(CPS-Mx341-xxx)时（包括Gateway系列）

DIP SW1的6号开关为ON (SD boot模式有效)

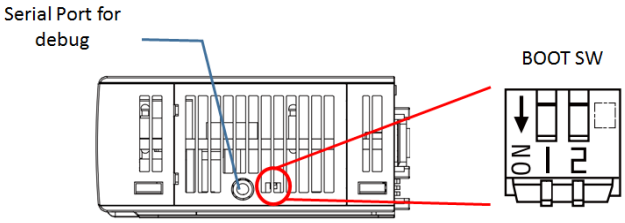
精巧一体型BOOT SW设置



◆ 堆栈组合型(CPS-MxS341-xxx)时

在调试用串口(3.5Φ迷你插孔)旁边的BOOT SW(框体中)的2号开关为ON (SD boot模式有效)

堆栈组合型BOOT SW设置



插入在<**启动用SD卡的制作 (P31)**>中制作的SD卡，接通主机电源。

※SD卡没插时从内置NOR FLASH启动。

2. 从内置NOR FLASH启动

请确认<**从SD卡启动 (P44)**>中记载的SD boot模式已禁用，然后接通主机电源。

2. 使用串口电缆连接登录

通过用串行电缆从主机电脑连接到CONPROSYS的专用串行端口(3.5Φ迷你插孔)，可以从控制台登录CONPROSYS。
串行设置如下：

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

建议主机电脑和CONPROSYS使用以下的USB串口转换电缆连接。
请根据主机电脑的操作系统版本S下载正确的驱动程序。

- FTDI产　TTL-232R-3V3-AJ
  驱动程序提供URL:　http://www.ftdichip.com/Drivers/VCP.htm

缺省登录/密码如下所示：

登录:	conprosys
密码:	contec

※在可连接外部网络的环境下，根据<**安全注意事项 (P12)**>，请务必进行密码变更。

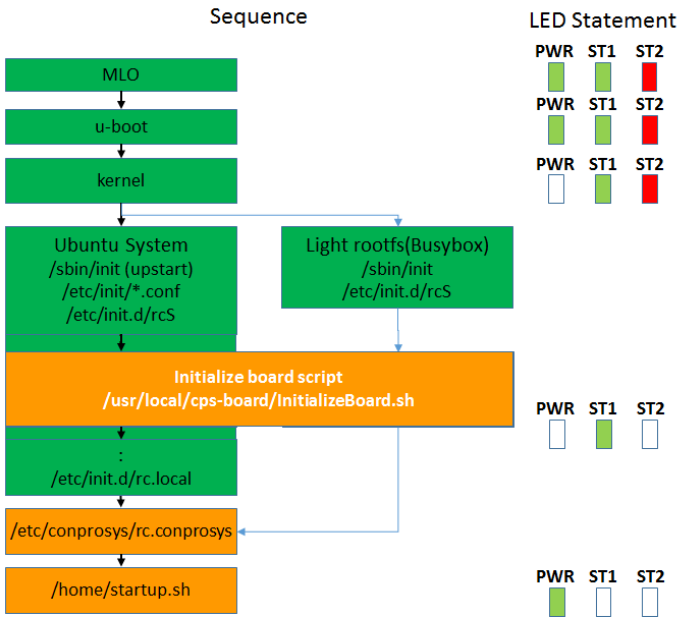
3. 通过ssh连接登录

如果主机电脑和CONPROSYS是在同一网络环境下运行时，可以使用ssh登录到CONPROSYS。

4. 目标的启动顺序

在本SDK中目标的启动将按照以下顺序执行。

启动顺序



请通过编辑下面的脚本文件来设置在目标启动时要执行的命令。

[SD卡引导时]

```
/home/startup.sh
```

[内置NOR FLASH引导时]

```
/mnt/mtd/startup.sh
```

在目标系统上，文件系统已设置为READ ONLY模式，要编辑这些文件，需将其置于WRITE许可模式后进行，编辑结束后请返回READ ONLY模式。

[SD卡引导时]

WRITE许可模式

```
sudo rommode rw
```

READ ONLY模式

```
sudo rommode ro
```

[内置NOR FLASH引导时]

WRITE许可模式

```
sudo rommode_mtd rw
```

READ ONLY模式

```
sudo rommode_mtd ro
```

5. 目标的网络设置

本SDK目标的默认网络设置如下：

[默认设置]

LAN A (eth0)： 10.1.1.101

LAN B (eth1)： DHCP （仅设置2 LAN Type时）

要变更网络设置时，请匹配各个rootfs进行编辑。

如果希望更改网络设置，请以root权限编辑目标上的以下文件。

/etc/conprosys/config.ini

LAN设置

项目名	设置内容
eth0_dhcp	设置LAN A(eth0)的DHCP有效/无效。 有效: enabled 无效: disabled
eth0_ipaddr	设置LAN A(eth0)的IP地址。
eth0_netmask	设置LAN A(eth0)的Netmask。
eth0_gateway	设置LAN A(eth0)的网关地址。
eth0_dns1	设置LAN A(eth0)的DNS服务器地址。
eth1_dhcp	设置LAN B(eth1)的DHCP有效/无效。 有效: enabled 无效: disabled
eth1_ipaddr	设置LAN B(eth1) 的IP地址。
eth1_netmask	设置LAN B(eth1)的Netmask。
eth1_gateway	设置LAN B(eth1)的网关地址。
eth1_dns1	设置LAN B(eth1) 的DNS服务器地址。
ntp_addr	设置NTP服务器。
host_name	设置主机名。 初始状态下没有项目，因此设置为下述主机名。 模块名＋MAC地址的后3个字节

3G/LTE设置 （仅3G/LTE型）

项目名	设置内容
m3g_connect	设置3G/LTE连接的有效/无效。 有效: enabled 无效: disabled
m3g_apn	设置通信服务提供商提供的APN。
m3g_user	设置通信服务提供商提供的用户ID。
m3g_passwd	设置通信服务提供商提供的密码。
m3g_auth	设置通信服务提供商提供的下记认证类型。 None PAP CHAP

无线LAN设置

项目名	设置内容
wlan_dhcp	设置无线LAN(wlan0)的DHCP有效/无效。 有效: enabled 无效: disabled
wlan_ipaddr	设置无线LAN(wlan0)的IP地址。
wlan_netmask	设置无线LAN(wlan0)的Netmask。
wlan_gateway	设置无线LAN(wlan0)的网关地址。
wlan_dns1	设置无线LAN(wlan0)的DNS服务器地址。
wlan_essid	设置无线LAN(wlan0)的SSID。
wlan_encrypt	从下记中选择设置无线LAN(wlan0)的加密方式。 [设置项目] 没有加密: 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自动: wpawpa2psk-auto
wlan_key	设置无线LAN(wlan0)的加密密钥。

※通过连接支持USB的USB无线LAN适配器可使用无线LAN。

服务启动设置

项目名	设置内容
srv_ssh	设置SSH服务器的启动。 enabled: 有效 disabled: 无效
srv_ftp※	设置FTP服务器的启动。 enabled: 有效 disabled: 无效
srv_samba※	设置SAMBA服务器的启动。 enabled: 有效 disabled: 无效

※Ubuntu14.04（无SDK）时，FTP/samba服务器需要另外的软件包。

另外，对于轻量版rootfs，samba服务器不可用。

路由功能设置

项目名	设置内容
router	设置路由功能。 enabled: 有效 disabled: 无效
wan_if	设置WAN接口。 3G: eth2 LTE: ppp0 Wireless LAN: wlan0 LAN A: eth0 LAN B: eth1
dhcp_server	设置DHCP服务器的启动。 enabled: 有效 disabled: 无效
dhcp_server_lan_if	设置DHCP服务器的LAN接口。 Wireless LAN: wlan0 LAN A: eth0 LAN B: eth1
dhcp_server_top_addr	设置DHCP起始地址。
dhcp_server_alloc_num	设置DHCP地址分配数。

※Ubuntu14.04（无SDK）时，DHCP服务器需要另外的软件包。

PPPoE功能设置

项目名	设置内容
pppoe	设置PPPoE功能。 enabled: 有效 disabled: 无效
pppoe_if	设置PPPoE接口。 LAN A: eth0 LAN B: eth1
pppoe_user	设置PPPoE的用户名。
pppoe_password	设置PPPoE的密码。
pppoe_dns	设置PPPoE的DNS服务器。
pppoe_firewall	设置PPPoE的防火墙。 NONE: 0 STANDALONE: 1 MASQUERADE: 2

※PPPoE在Ubuntu14.04 include SDK时可设置。
其他rootfs时，需要另外的PPPoE软件。

静态路由功能设置

项目名	设置内容
static_route	设置静态路由功能。 enabled: 有效 disabled: 無効
st_route_addr_1	设置静态路由的目标IP地址。
st_route_gw_1	设置静态路由的网关地址。
st_route_mask_1	设置静态路由的子网掩码。
st_route_if_1	设置静态路由的接口。
:	
:	
:	
st_route_addr_32	设置静态路由的目标IP地址。
st_route_gw_32	设置静态路由的网关地址。
st_route_mask_32	设置静态路由的子网掩码。
st_route_if_32	设置静态路由的接口。

※项目名的数字代表设置No.。(最大到32)

端口转发功能设置

项目名	设置内容
port_forward	设置端口转发功能。 enabled: 有效 disabled: 无效
port_fw_sif_1	设置端口转发输入接口。
port_fw_sport_1	设置端口转发输入端口。
port_fw_daddr_1	设置端口转发目标IP地址。
port_fw_dport_1	设置端口转发目标端口。
:	:
port_fw_sif_32	设置端口转发输入接口。
port_fw_sport_32	设置端口转发输入端口。
port_fw_daddr_32	设置端口转发目标IP地址。
port_fw_dport_32	设置端口转发目标端口。

※项目名的数字代表设置No.。(最大到32)

IP过滤器功能设置

项目名	设置内容
ipfilter	设置IP过滤器功能。 enabled: 有效 disabled: 无效
ipfilter_kind_1	设置过滤器类型。 许可： ACCEPT 废弃： DROP
ipfilter_proto_1	设置协议。 TCP, UDP, ICMP, ALL
ipfilter_saddr_1	设置发送源IP地址。
ipfilter_sport_1	设置发送源端口。
ipfilter_daddr_1	设置发送目标IP地址。
ipfilter_dport_1	设置发送目标端口。
:	:
ipfilter_kind_64	设置过滤器类型。 许可： ACCEPT 废弃： DROP
ipfilter_proto_64	设置协议。 TCP, UDP, ICMP, ALL
ipfilter_saddr_64	设置发送源IP地址。
ipfilter_sport_64	设置发送源端口。
ipfilter_daddr_64	设置发送目标IP地址。
ipfilter_dport_64	设置发送目标端口。

※项目名的数字代表设置No.。(最大到64)

Ubuntu14.04 include SDK的情况下，可从电脑等的Web浏览器，经由LAN与CONPROSYS连接，进行网络设置。 详情请参阅<**Web Setup功能 (P55)**>。

6. 驱动程序的启动方法

需要手动启动的驱动程序或在<**6-3. CPS-MxS341系列驱动程序的编译**>中编译的驱动程序可以通过modprobe命令启动。通过串口连接登录目标Conprosys，或通过ssh登录执行命令。

命令示例： can驱动程序(d_can_platform)的启动

```
modprobe d_can_platform
```

可以使用以下命令确定驱动程序是否已启动。

命令：

```
lsmod
```

如果希望自动启动驱动程序，请参考<**目标的启动顺序(P47)**>向startup.sh文件添加启动驱动程序的命令。

7. Web Setup功能

Rootfs类型指定成以下内容时，搭载Web Setup功能。

Ubuntu 14.04 (include SDK)　Ubuntu 14.04 附带SDK(本机开发版CONPROSYS Linux SDK)

light (busybox)　轻量版rootfs

※light(busybox)在装载工具中包含Apache2，PHP5时,会配备Web Setup功能。

Web Setup功能包括网络和时间等设定、系统信息和网络等状态显示功能等。 通过从电脑等的Web浏览器直接访问CONPROSYS的IP地址，可显示CONPROSYS的设置画面。

例： 初始设置时，将电脑连接到LAN A端口进行确认

http://10.1.1.101/

登录： admin

密码： password

Web设置画面

Web Setup具有以下功能：

1. 设置菜单

可进行以下设置：

设置菜单项目

设置种类	设置内容	初始值	备注
系统	主机名	(无设置)	没有设置时，设置为下述主机名。 模块名＋MAC地址的后3个字节
网络	有线LAN A	10.1.1.101(固定IP)	
	有线LAN B	DHCP	
	3G/LTE网络		仅限3G/LTE模块搭载机型
	无线LAN	DHCP	仅在与兼容USB无线适配器连接时
时间	NTP服务器	(无设置)	
	手动设置		
服务启动	SSH服务器	系统启动时：有效	
	FTP服务器	系统启动时：无效	
	SAMBA服务器	系统启动时：无效	
路由功能	路由功能	系统启动时：无效	
	WAN接口		
	DHCP服务器功能	系统启动时：无效	
	静态路由设置	系统启动时：无效	设置最大数：32
	端口转发设置	系统启动时：无效	设置最大数：32
IP过滤器	IP过滤器设置	系统启动时：无效	设置最大数：64

2. 状态菜单

可确认以下状态:

状态菜单项目

项目	内容
系统	显示主机名、序列号、分发/内核信息、磁盘/内存使用量等。
网络	显示IP地址、MAC地址、RX/TX字节等。
路由功能	显示路由表和NAT表。
IP过滤器	显示IP过滤器的设置信息。
Log	显示日志，如syslog。

3. 维护菜单

可进行以下维护处理：

维护菜单项目

项目	内容
密码	可以更改访问Web设置画面的密码。
设置文件	可备份和恢复设置文件。
缺省设置	可以返回到出厂时的默认设置。
Ping	为了确认网络是否联通，可以进行Ping。

4. 结束菜单

可进行以下处理。

结束菜单项目

项目	内容
保存和再启动	保存设置项目，再启动。
保存和关机	保存设置项目，关机(系统停止)。
保存	保存设置项目。
再启动	再启动。不保存设置项目，再启动时返回到设置前的状态。
关机	关机(系统停止)。不保存设置项目，关机时返回到设置前的状态。

有关如何使用Web Setup功能的详细信息，请参阅Web菜单中的<帮助>。

Web设定项目通过下述文件进行管理。

设置文件：

/etc/conprosys/config.ini

出厂时设置文件：

/etc/conprosys/config_def.ini

Web文件由以下文件夹管理。

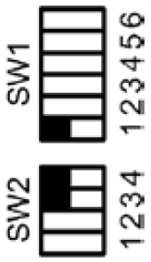
Web内容文件夹：

[Ubuntu 14.04 (include SDK)]
/var/www/html/
[轻量版rootfs]
/opt/htdocs/

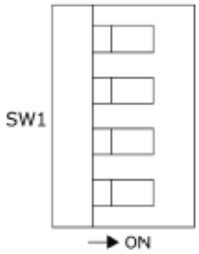
8. 使用DIP开关的初始化相关设置

设置CONPROSYS的DIP开关，可以在上电后以初始化LAN A的IP地址启动CONPROSYS，或将CONPROSYS的设置恢复为出厂默认设置。

精巧一体型



堆栈组合型



DIP开关的设置方法

SW设置	操作
仅将SW1的2号开关设置为ON	上电时，以出厂时的IP地址设置启动CONPROSYS。 用户/密码、组保持原来的设置。 在Web画面中，可确认当前IP地址设置、用户/密码设置。
将SW1的2号、3号开关均设置为ON	上电时，CONPROSYS将恢复为按出厂时默认设置。 恢复完成后， PWR和ST1的LED会闪烁。请在确认闪烁后，将2号和3号开关恢复为OFF，然后重新启动。

编译

1. 编译步骤

为了生成目标CONPROSYS的运行环境，根据需要进行以下操作。

1 编译的初始设置（configure.sh）
运行./configure.sh，生成基础的目标执行环境。
此外，本机开发版的CONPROSYS Linux SDK也可以通过此编译的初始配置来生成运行环境。

2 编译的环境设置（source sdkenv.txt）
设置环境变量以执行编译和生成SD卡。
登录后，在执行编译前请务必进行该设置。

3 Bootloader的编译
如想更改电源ON后的启动程序，则进行本编译。
通常不需要进行。

4 kernel的编译
如果想更改Linux kernel，则进行本编译。
请在对未默认设置的kernel功能或驱动程序进行添加/更改/删除时进行。
kernel如果不更改，则不需要执行此操作。

5 编译示例驱动程序
如果想更改示例驱动程序，则进行本编译。
如果不更改示例驱动程序，则不需要编译。

6 编译示例库
如果想更改示例库，则进行本编译。
如果不更改示例库，则不需要编译。

7 编译示例应用程序
如果想在目标上运行示例应用程序，则进行本编译。
如果不使用示例应用程序或在CONPROSYS上进行本机开发，则无需编译。

2. 目标的bootloader的编译

bootloader的编译可以在u-boot的文件夹中进行。一般不需要进行编译，但在bootloader的源代码发生变更或编译器选项发生变更时请编译。 u-boot有SD卡启动用和内置NOR FLASH启动用2种。

1. SD卡启动用的编译

移动到u-boot文件夹：

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

编译命令：

```
make am335x_evm
```

编译生成的模块(MLO，u-boot.img)通过以下命令复制到目标boot文件夹中。

命令：

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

要删除编译时生成的对象文件等时，执行以下命令。

命令：

```
make distclean
```

2. 内置NOR FLASH启动用的编译

从内置NOR FLASH启动的bootloader模块与用SD卡启动的bootloader模块不同。此bootloader模块通过以下方法编译。

进入u-boot文件夹：

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

编译命令：

```
make am335x_evm_spiboot
```

编译生成的模块(MLO.byteswap，u-boot.img)复制到内置NOR FLASH安装的rootfs(/InstallToMTD)。

※ 内置NOR FLASH用于安装的rootfs的制作方法请参阅<**向内置NOR FLASH安装用rootfs部分的复制 (P38)**>。

命令：

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

3. 目标的kernel的编译

Kernel的配置/编译可以在kernel的文件夹中完成。

一般不需要进行编译，但在kernel选项更改或安装不支持的USB等设备驱动程序时请编译。

移动到kernel文件夹：

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

配置命令：

```
make menuconfig
```

编译命令：

```
make uImage
```

编译生成的模块(uImage)通过以下命令复制到目标boot文件夹中。

命令：

```
cp -p arch/arm/boot/uImage ${CPS_SDK_INSTALL_FULLDIR}/boot/uImage.${CPS_SDK_BOARD_NAME}
```

内核编译后，编译驱动程序模块，并在以下命令示例中将其复制到目标rootfs文件夹中。

编译命令：

```
make modules
```

安装命令示例：

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

※驱动程序模块安装在rootfs文件夹下的以下文件夹中。

lib/modules/3.2.0.CONPROSYS/

如果要安装到其他rootfs文件夹，请更改INSTALL_MOD_PATH的内容。

要删除编译时生成的对象文件等时，执行以下命令。

命令：

```
make clean
```

如果kernel的配置信息损坏或想要恢复到初始状态时，请执行以下命令。

命令：

```
make distclean
```

```
make ${CPS_SDK_TARGET_NAME}_defconfig
```

执行上述命令后，请执行配置和编译。

4. CPS-MxS341系列驱动程序的编译

下述SDK的CPS-MxS341系列驱动程序的源代码包含在driver文件夹中。

- cps-driver　　(CPS-MxS341用系统驱动程序)
- cpsaio　　(CPS-MxS341用AIO驱动程序)
- cpsdio　　(CPS-MxS341用DIO驱动程序)
- 8250_cpscom　　(CPS-MxS341用COM驱动程序)
- cpsssi　　(CPS-MxS341用SSI驱动程序)
- cpscnt　　(CPS-MxS341用CNT驱动程序)
- cps_iolib　　(CPS-MxS341用IO通用访问驱动程序)

一般不需要进行编译，但在示例驱动程序发生更改时请编译。

要编译驱动程序，需要kernel编译结果的源代码，因此请事先执行kernel的编译。(<**目标的kernel的编译 (P62)**>:参照)。

在各个文件夹下，可以通过以下命令编译。

命令：

```
make
```

驱动程序编译后，按以下命令示例将其复制到目标rootfs文件夹中。

命令示例：

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

※驱动程序模块安装在目标rootfs下的以下文件夹中。

lib/modules/3.2.0.CONPROSYS/extra

如果要将其安装到其他rootfs文件夹，请更改INSTALL_MOD_PATH的内容。

注意）包含的各驱动软件并不是每一个都支持所有的产品的。
请在确认适用设备的基础上进行操作。

5. 目标的示例库的编译

SDK包含了以下共享库(Shared Object)的源代码。

- libCpsEeprom (EEPROM数据访问模块库)
- libCpsAio (CPS-MxS341用AIO库)
- libCpsDio (CPS-MxS341用DIO库)
- libCpsSsi (CPS-MxS341用SSI库)
- libCpsCnt (CPS-MxS341用CNT库)
- libconexio (CPS-MC341Q-ADSC1用920MHz模块库)
- SerialFunc (CPS-MC341Q-ADSC1用串行模块库)

一般不需要进行编译，但在示例库发生更改时请编译。

在各文件夹下，可以通过以下命令编译。

命令：

```
make
```

库编译后，使用以下命令将其复制到目标rootfs文件夹中。

命令：

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

※库模块安装在目标rootfs下的以下文件夹中。

usr/local/lib

如果要安装到其他rootfs文件夹，请更改TARGET_ROOTFS的内容。

创建库时，请参考makefile或源代码。

注意1） 包含的库软件并不是每一个都支持所有的产品的。
请在确认适用设备的基础上进行操作。

注意2） 包含的每个示例库软件都需要对应的驱动软件。
请参考下面的依存关系。

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. 目标的示例应用程序的编译

SDK在下面的文件夹中包含了示例应用程序的源代码。

${CPS_SDK_ROOTDIR}/application/sample/

每个目标CONPROSYS都有对应的示例应用程序。(参见<**示例程序对应表**>) 请用于目标CONPRYS的动作确认或编写应用程序时参考。要编译示例程序，在各个文件夹中运行make命令会生成可执行文件。

例：定时器的示例程序

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

示例程序对应表

示例程序	文件夹 application/sample/	CPS-MC341-ADSCx CPS-MC341G-ADSC1	CPS-MC341Q-ADSC1	CPS-MC341-Ax	CPS-MC341-DSx	CPS-MC341-DS1x	CPS-MCS341-DSx CPS-MGS341-DS1 CPS-MCS341G-DS1	CPS-MCS341Q-DS1
TCP/IP 服务器/客户端	socket	○	○	○	○	○	○	○
定时器	timer	○	○	○	○	○	○	○
EEPROM数据读取	getEepromData	○	○	○	○	○	○	○
CAN发送/接收测试	can			△	△			
RS-485通信(精巧一体型用)	RS485	○	○	△	○	○		
DI/DO, AI控制(多功能型用)	mc341_io	○	○					
AI/AO控制(精巧一体型用)	mc341-ax_aio			○				
AI/AO控制(堆栈组合型用)	mcs341_aio						○	○
DI/DO控制(精巧一体型用)	spitest	○	○		○	○		
http控制(DIO) (精巧一体型用)	http_post	○	○		○	○		
DI/DO控制(堆栈组合型用)	mcs341_dio						○	○
SSI控制(堆栈组合型用)	mcs341_ssi						○	○
COM控制(堆栈组合型用)	mcs341_com						○	○
CNT控制(堆栈组合型用)	mcs341_cnt						○	○
System控制(堆栈组合型用)	mcs341_system						○	○
iolib控制(堆栈组合型用)	mcs341_iolib						○	○
920MHz发送/接收测试	conexio_CMM920		○					

○：对应　△：部分机型对应　空栏：未对应

不是每一个示例应用程序都支持所有CONPROSYS产品的。

即使是对应的设备，由于设备端口等的不同，也有不支持的程序，请在确认程序后进行编译和测试。

另外，对于需要驱动程序/库的示例软件的编译，请先编译这些驱动程序/库，然后再编译示例应用程序。

关于设备端口，请参阅<**设备I/F(P76)**>。

编写应用软件时，请参考这些makefile或源代码。

7. 轻量版rootfs的编译

本SDK包含了小型轻量版linux的rootfs源代码，用于内置NOR FLASH等小容量的启动设备。Rootfs的内容如下：

busybox, glibc, dropbear (轻量版SSH服务器/客户端), iptables, sudo

可在rootfs的文件夹下编译。

移动到light文件夹：

```
cd ${CPS_SDK_ROOTDIR}/rootfs/light
```

编译命令：

```
make
```

可以通过以下命令在目标文件夹下的rootfs中创建rootfs文件系统。

命令：

```
make install
```

另外，在Makefile中的环境变量EXPORTDIR指定的文件夹中，也可以进行编译。

命令示例： 如果希望在内置NOR FLASH启动用目标文件夹下编译rootfs时

```
make install EXPORTDIR=${CPS_SDK_ROOTDIR}/ramdisk/export
```

8. 内置NOR FLASH启动用ramdisk.xz的编译

内置NOR FLASH的rootfs压缩并存储在名为ramdisk.xz的文件中。

当轻量版rootfs的编译成功完成时，可以使用下面的命令从rootfs生成ramdisk.xz。

移动到ramdisk文件夹：

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

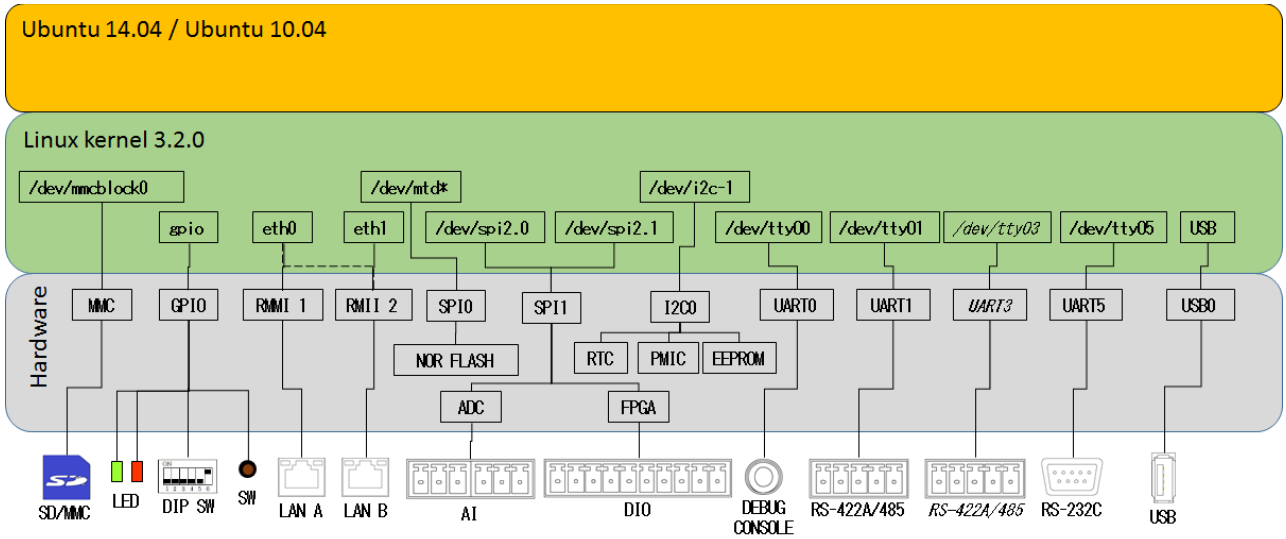
编译命令：

```
make
```

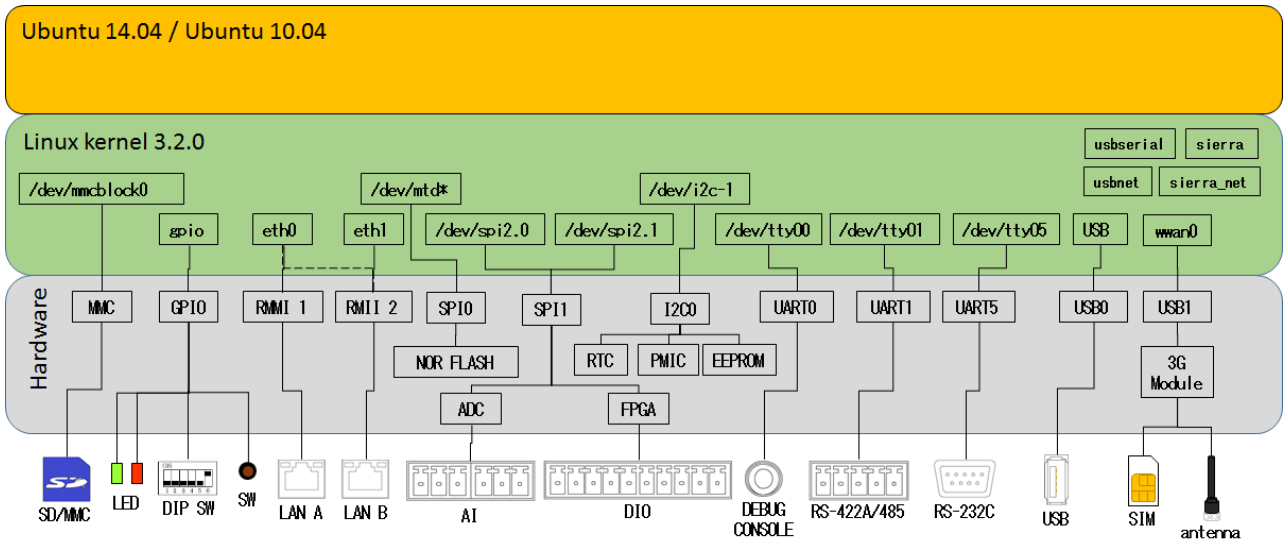
附录

1. 结构图

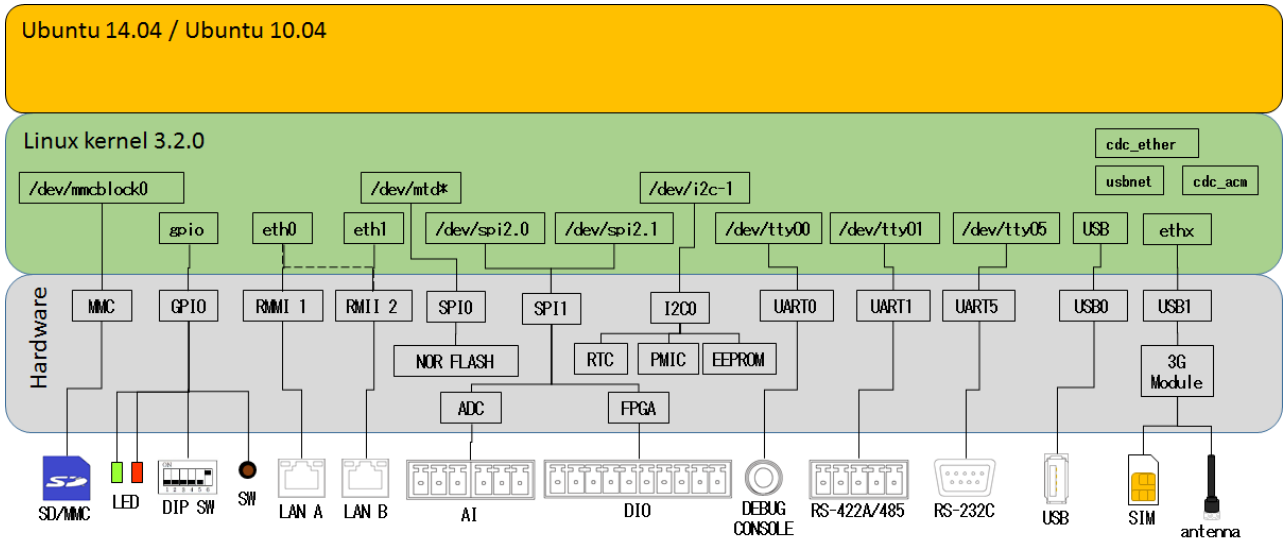
CPS-Mx341-ADSCx系列结构图（斜体字为可选项）



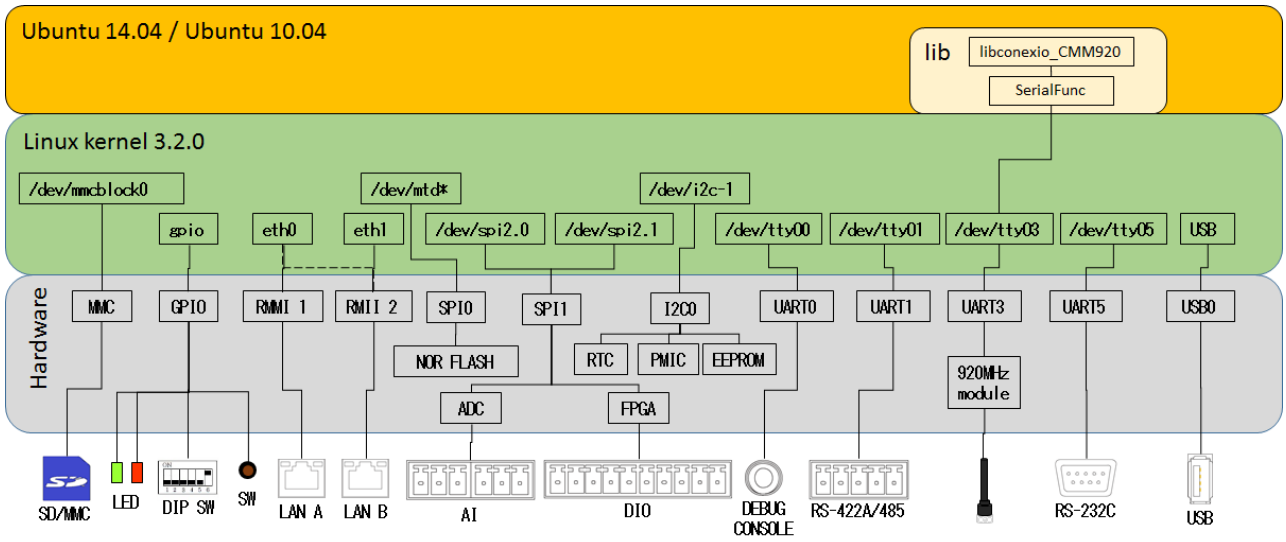
CPS-Mx341G-ADSC1（日本国内专用）结构图



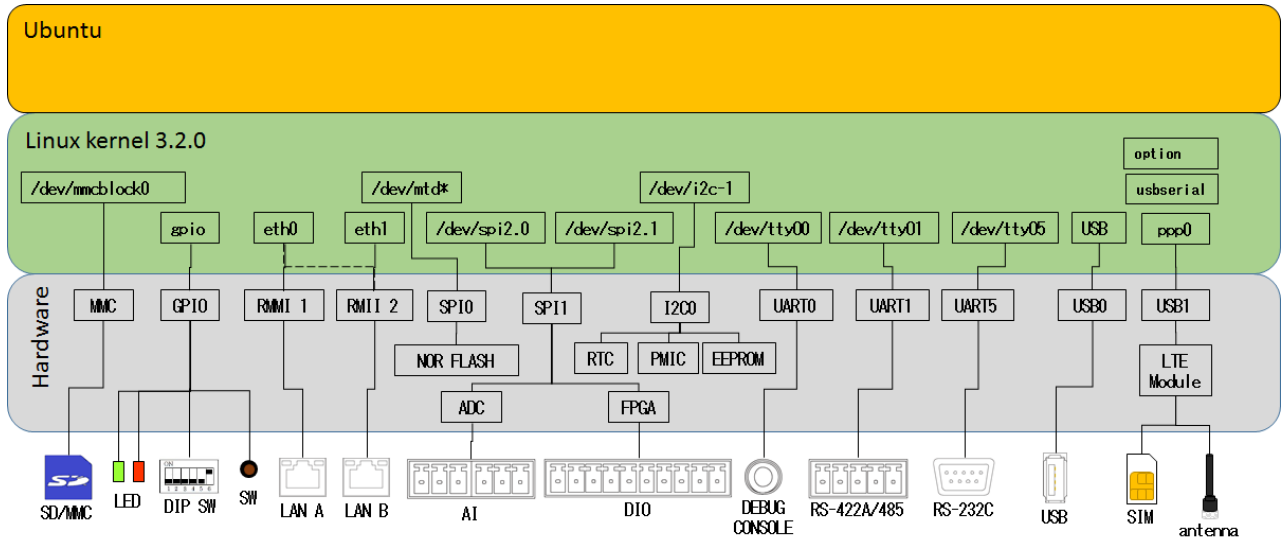
CPS-Mx341G-ADSC1（全球型）结构图



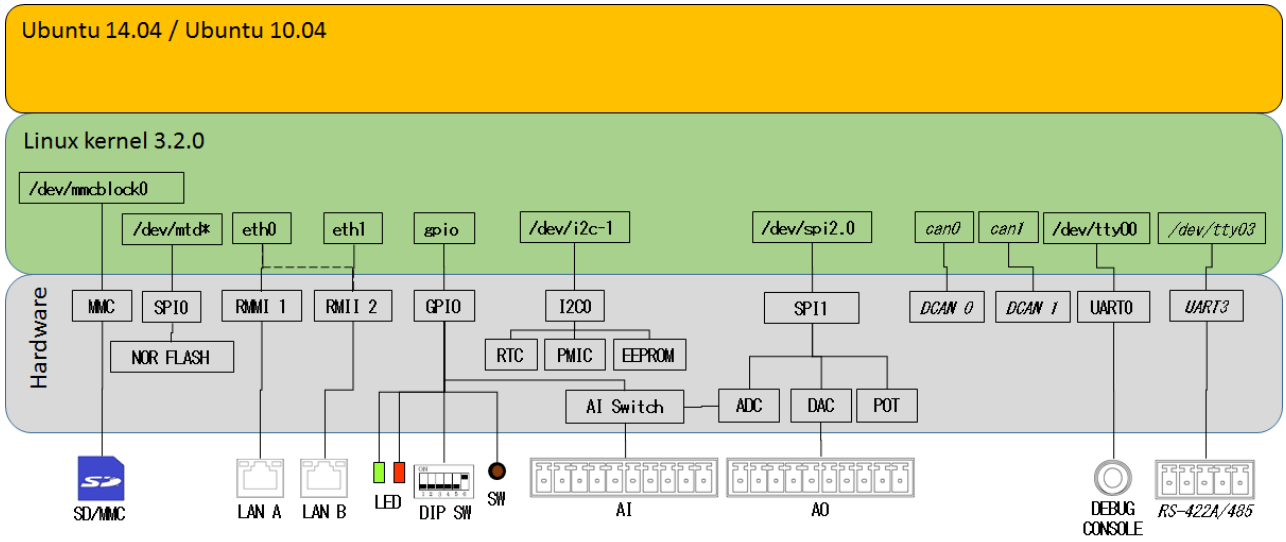
CPS-MC341Q-ADSC1结构图



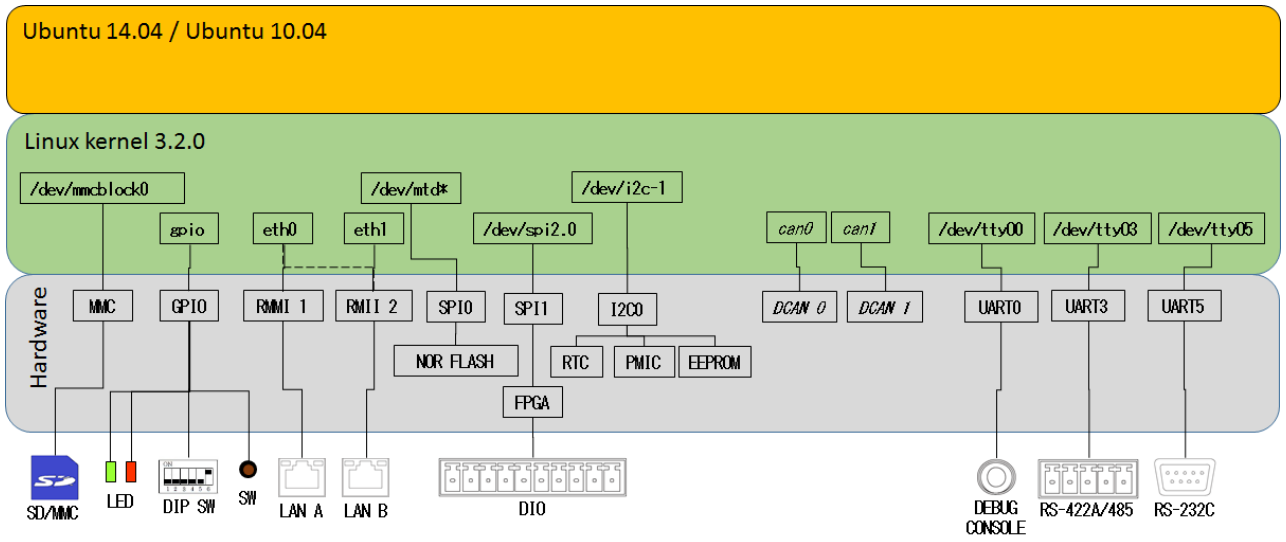
CPS-MG341G5-ADSC1结构图



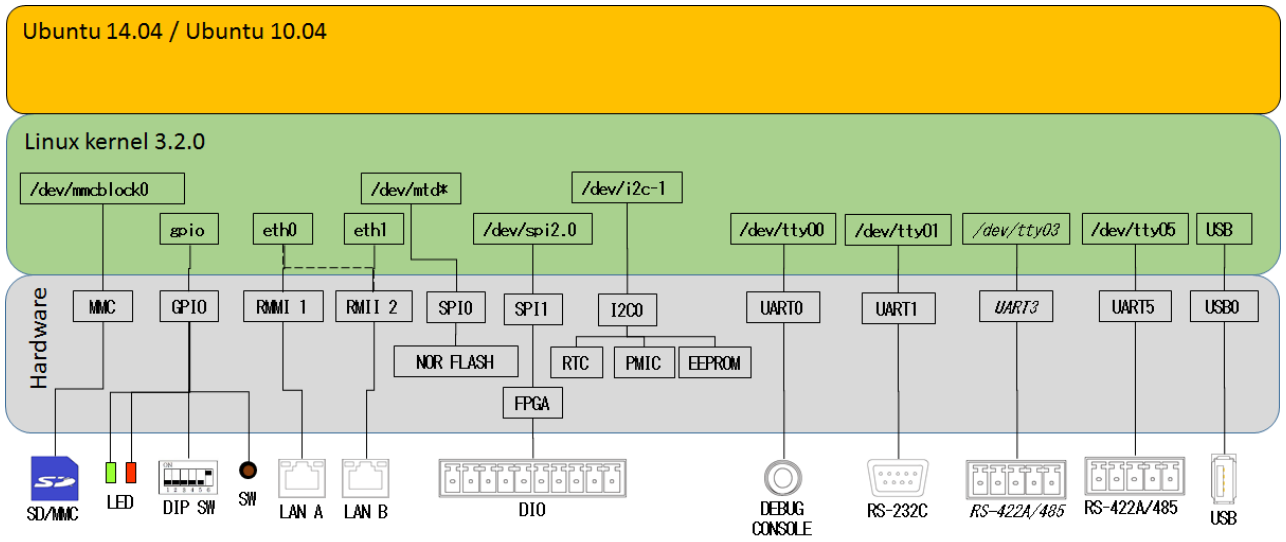
CPS-MC341-Ax系列结构图（斜体字为可选项）



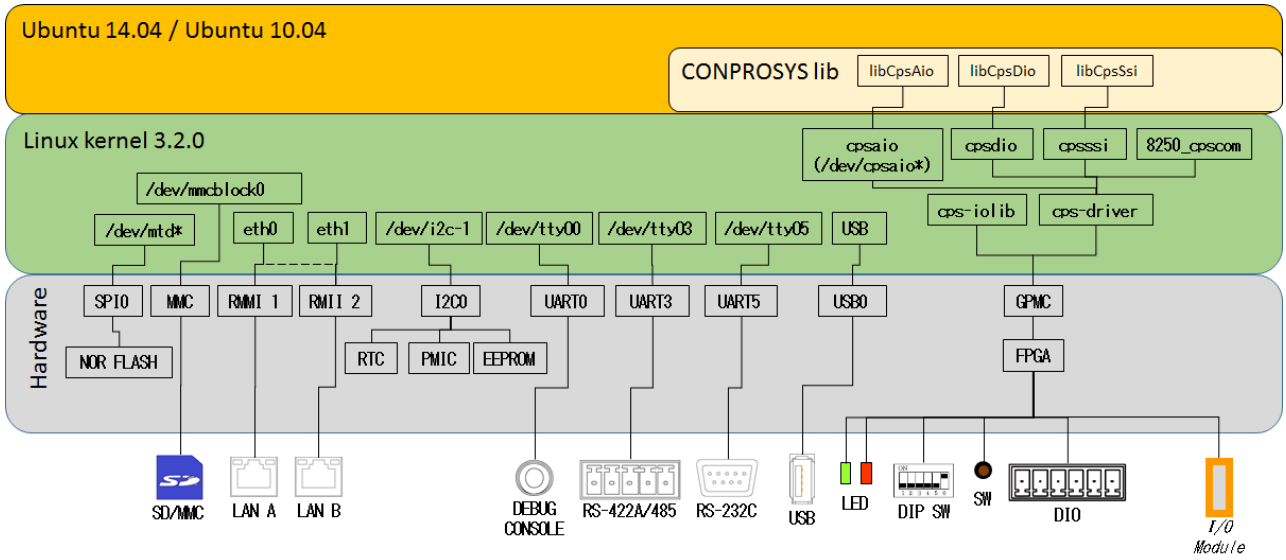
CPS-MC341-DSx系列结构图（斜体字为可选项）



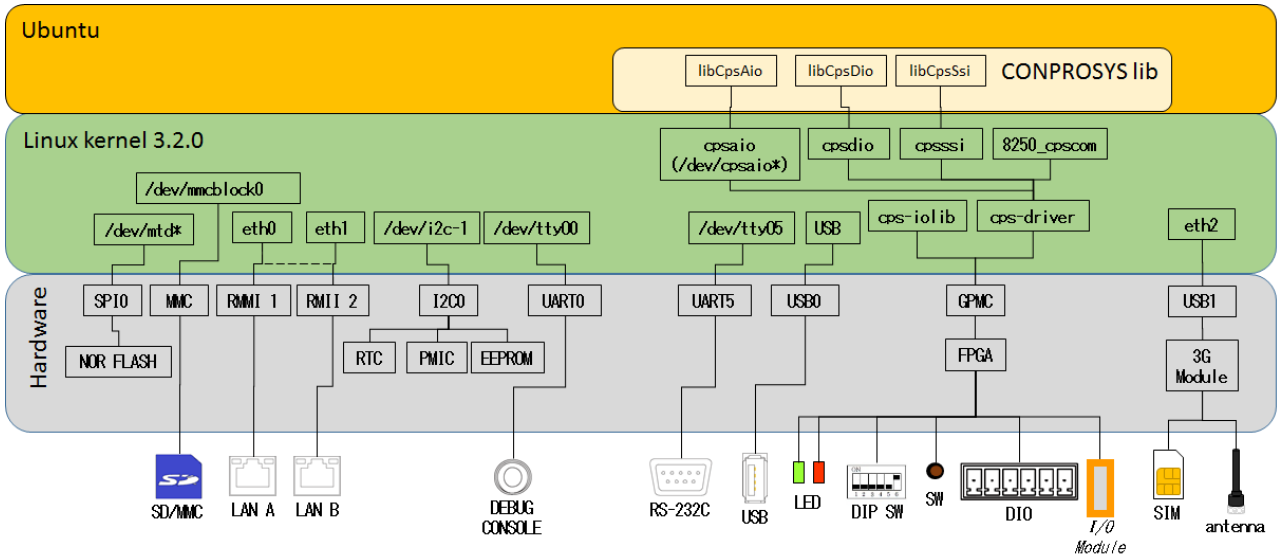
CPS-MC341-DS1x系列结构图（斜体字为可选项）



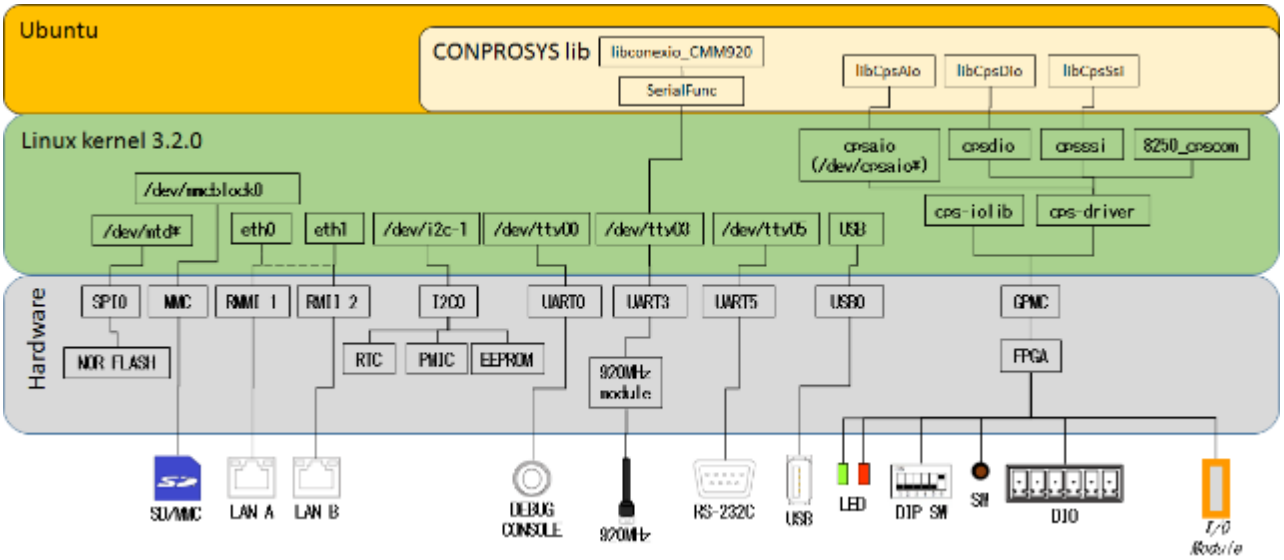
CPS-MxS341-DSx系列结构图（斜体字为可选项）



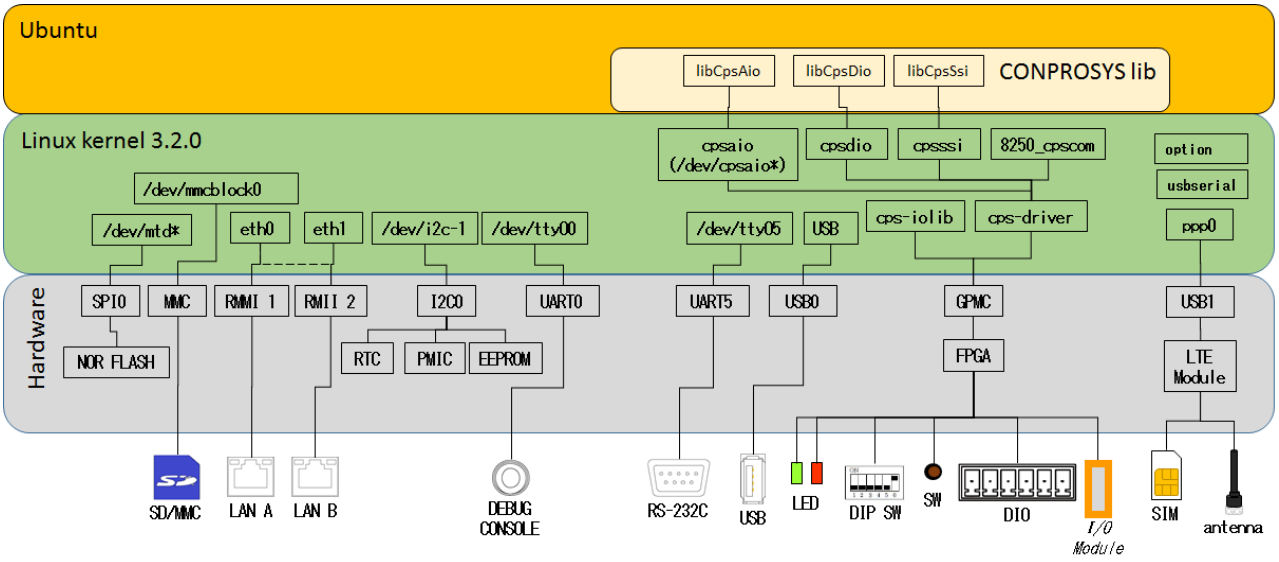
CPS-MCS341G-DS1系列结构图（斜体字为可选项）



CPS-MCS341Q-DS1系列结构图（斜体字为可选项）



CPS-MxS341G5-DS1系列结构图（斜体字为可选项）



2. 设备I/F

CONPROSYS具有的设备I/F可以在Linux上访问，如下表所示。

请注意，根据设备的不同端口可能会有所不同。

UART控制设备

型号	/dev/tty01	/dev/tty02	/dev/tty03	/dev/tty04	/dev/tty05
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-MC341-A1	-	-	-	-	-
CPS-MC341-DS1	-	-	-	-	RS-422A/485 (COM A)
CPS-MC341-DS2	(CAN用)※1	-	-	-	RS-422A/485 (COM A)
CPS-MC341-DS11	RS-232C (COM A)	-	-	-	RS-422A/485 (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

※1 为CAN端口预留。控制请通过Network进行。

SPI控制设备

型号	/dev/spidev2.0	/dev/spidev2.1	/dev/spidev2.2
CPS-MC341-ADSCx CPS-MC341G-ADSC1 CPS-MG341G5-ADSC1 CPS-MC341Q-ADSC1	AI (ADC / CLK=6MHz)	DIO (FPGA / CLK=24MHz)	-
CPS-MC341-A1	AI (ADC / CLK=6MHz)	AO (DAC / CLK=20MHz)	Potentiometers (CLK=25MHz)
CPS-MC341-DSx	DIO (FPGA / CLK=24MHz)	-	-
CPS-MC341-DS11	DIO (FPGA / CLK=24MHz)	-	-
CPS-MCS341-DS1 CPS-MGS341-DS1 CPS-MCS341G-DS1 CPS-MCS341G5-DS1 CPS-MGS341G5-DS1 CPS-MCS341Q-DS1	-	-	-

括号内为连接设备和SPI控制MAX时钟值

GPIO控制设备（LED系）

型号	GPIO 26	GPIO 27	GPIO 67	GPIO 128	GPIO 129
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)	LTE Green (Out)	LTE Red (Out)
CPS-MC341-A1	ST1 Green (Out)	ST2 Red (Out)	Power (Out)	-	-
CPS-MC341-DSx	ST1 Green (Out)	ST2 Red (Out)	Power (Out)	-	-
CPS-MC341-DS11	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	-	-	-	LTE Green (Out)	LTE Red (Out)

GPIO控制设备（Switch系）

型号	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-MC341-A1	DIP SW1-2 (In)	DIP SW1-3 (In)	DIP SW1-4 (In)	Shutdown SW (In)	-
CPS-MC341-DSx	DIP SW1-2 (In)	DIP SW1-3 (In)	DIP SW1-4 (In)	Shutdown SW (In)	-
CPS-MC341-DS11	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控制设备（Input Switch控制系）

型号	GPIO 39	GPIO 44	GPIO 45	GPIO 46	GPIO 47	GPIO 100
CPS-MC341-ADSCx CPS-MC341G-ADSC1 CPS-MG341G5-ADSC1 CPS-MC341Q-ADSC1	-	-	-	-	-	-
CPS-MC341-A1	DAC LDACB (Out)	AI switches A0 (Out)	AI switches A1 (Out)	AI switches A2 (Out)	AO Switch (Out)	Potentiometers $\overline{CS}$ (Out)
CPS-MC341-DSx	-	-	-	-	-	-
CPS-MC341-DS11	-	-	-	--	-	-
CPS-MCS341-DS1 CPS-MGS341-DS1 CPS-MCS341G-DS1	-	-	-		-	-

型号	GPIO 39	GPIO 44	GPIO 45	GPIO 46	GPIO 47	GPIO 100
CPS-MCS341G5-DS1 CPS-MGS341G5-DS1 CPS-MCS341Q-DS1						

GPIO控制设备 (Board控制系)

型号	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-MC341-A1	-	-	-	-	Power RESET (Out)
CPS-MC341-DSx	-	-	-	-	Power RESET (Out)
CPS-MC341-DS11	-	-	-	-	Power RESET (Out)
CPS-MCS341-DS1 CPS-MGS341-DS1 CPS-MCS341G-DS1 CPS-MCS341G5-DS1 CPS-MGS341G5-DS1 CPS-MCS341Q-DS1	-	-	-	-	Power RESET (Out)

括号内容表示输入输出方向。

USB-Serial控制设备

型号	/dev/ttyUSB0	/dev/ttyUSB1	/dev/ttyUSB2	/dev/ttyUSB3	/dev/ttyUSB3
CPS-MC341-ADSCx CPS-MC341Q-ADSC1	Optional Serial Device				
CPS-MC341G-ADSC1 (日本国内专用)	Sierra USB modem	Sierra USB modem	Sierra USB modem	Sierra USB modem	Optional Serial device
CPS-MC341G-ADSC1 (全球型)	Optional Serial device				
CPS-MG341G5-ADSC1	Quectel USB modem	Quectel USB modem	Quectel USB modem	Quectel USB modem	Optional Serial device
CPS-MC341-A1					
CPS-MC341-DSx					
CPS-MC341-DS11	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

精巧一体型 ADC / DAC / FPGA(DIO) 使用设备

型号	设备	厂家	设备型号	控制端口
CPS-MC341-ADSC1 CPS-MC341-ADSC2 CPS-MC341G-ADSC1 CPS-MC341Q-ADSC1 CPS-MG341G5-ADSC1	ADC	Analog Devices	ADC7327	/dev/spidev2.0
	FPGA (DIO)	Lattice Semiconductor	LCMXO2-640HC-4TG1001	/dev/spidev2.1
CPS-MC341-A1	ADC	Texas Instruments	ADS8326IDGKR	/dev/spidev2.0
	AI Multiplexers	Analog Devices	ADG508FBRNZ	A0: GPIO 44 A1: GPIO 45 A2: GPIO 46
	DAC	Texas Instruments	DAC161S055CISQ	/dev/spidev2.1 LDACB: GPIO 39
	AO Switch	Toshiba	SSM3J135TU	Gate: GPIO 47
	Potentiometers	Analog Devices	AD5206BRUZ10	/dev/spidev2.2 $\overline{CS}$: GPIO 100
CPS-MC341-DSx	FPGA (DIO)	Lattice Semiconductor	LCMXO2-640HC-4TG1001	/dev/spidev2.0
CPS-MC341-DS11	FPGA (DIO)	Lattice Semiconductor	LCMXO2-640HC-4TG1001	/dev/spidev2.0

关于AIO设备控制详情，请根据上述信息获取各设备制造商的数据表并参考。有关DIO设备控制(FPGA)的信息，请参阅<FPGA I/O分布图(P81)>部分。

堆栈组合型 FPGA使用设备

型号	设备	厂家	设备型号	控制端口
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

有关设备控制(FPGA)的信息，请参阅<FPGA I/O分布图(P81)>部分。

堆栈组合型COM设备

型号	/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

堆栈组合型AIO 控制设备

型号	/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

堆栈组合型DIO 控制设备

型号	/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

堆栈组合型SSI 控制设备

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

堆栈组合型FPGA 控制设备

型号	/dev/cps-iolib
CPS-MxS341-DSx CPS-MCS341G-DS1 CPS-MCS341Q-DS1 CPS-MCS341G5-DS1 CPS-MGS341G5-DS1	GPMC

Network设备

Network Category	eth0	eth1	eth2	can0	can1	wwan0	ppp0
1 LAN(Hub Mode) Type	LAN A/B	–	–	–	–	–	–
2 LAN Type	LAN A	LAN B	–	–	–	–	–
CAN搭载型 1 LAN(Hub Mode) Type	LAN A/B	–	–	CAN※	CAN※	–	–
CAN搭载型 2 LAN(Hub Mode) Type	LAN A	LAN B	–	CAN※	CAN※	–	–
3G搭载型(日本国内型) 1 LAN(Hub Mode) Type	LAN A/B	–	–	–	–	3G	–
3G搭载型(日本国内型) 2 LAN Type	LAN A	LAN B	–	–	–	3G	–
3G搭載全球型 1 LAN(Hub Mode) Type	LAN A/B	3G	–	–	–	–	–
3G搭載全球型 2 LAN Type	LAN A	LAN B	3G	–	–	–	–
LTE搭載型 1 LAN Type	LAN A/B	–	–	–	–	–	LTE
LTE搭載型 2 LAN Type	LAN A	LAN B	–	–	–	–	LTE

※对于CAN搭载型，请安装CAN驱动程序(d_can_platform)进行控制。

3. FPGA I/O分布图

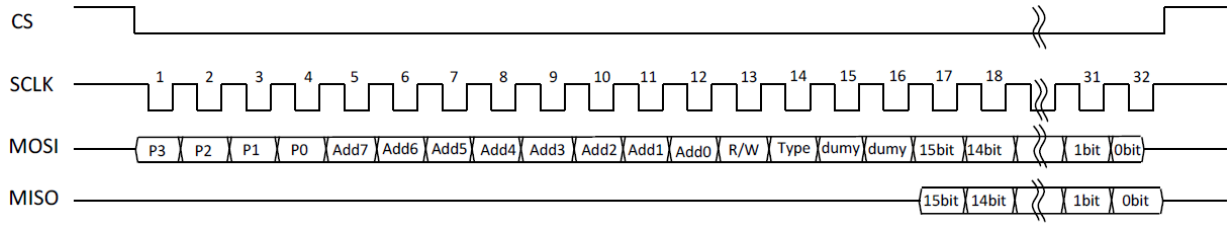
1.[精巧一体型 CPS-Mx341-ADSCx / DSx系列]

厂家 : Lattice Semiconductor

设备型号 : LCMXO2-640HC-4TG1001

接口 : SPI

SPI信号时序



MOSI : 在SCLK的下降沿，从站锁存信号

MISO : 在SCLK的上升沿输出从站信号，在SCLK的下降沿，主机锁存信号

SPI信号格式

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 : 固定 0

按Byte存取时，将数据后对齐，按16位数据进行收发。

例： 向Page = 0h、Address=12h中Write 00AAh时

0x0 12 C 00AA

Products Category

Products Category	Function	Register Page	适用机种
01h	数字量输入输出部分	0h	CPS-MC341-ADSCx, CPS-MC341-DSx
02h	模拟量输入部分	1h	CPS-MC341-ADSCx
03h	计数器部分	2h	CPS-MC341-ADSCx

数字量输入输出部分端口分布 (Page 0h)

Address	Read/Write种别	内容
00h - 01h	R	系统预留区域
02h - 03h	R	系统预留区域
04h - 0Ch	R	未使用
0Eh - 0Fh	R	系统预留区域
10h - 11h	R	数字量输入端口
12h - 13h	R/W	数字量输出端口
14h - 17h	R	未使用
18h - 19h	R/W	数字滤波器设定时间
1Ah - 1Fh	R	未使用
1Ch - 1Dh	R/W	内置电源 ON/OFF※
1Eh - 1Fh	R	未使用
20h - 21h	R/W	系统预留区域
22h - 23h	R	未使用
24h - 25h	R/W	系统预留区域
26h - FFh	R	未使用

※ 仅CPS-MC341-ADSC1-931对应

模拟量输入部分端口分布 (Page 1h)

Address	Read/Write种别	内容
00h - 01h	R	系统预留区域
02h - 03h	R	系统预留区域
04h - 27h	R	未使用
28h - 29h	R/W	模拟量输入部分
2Ah - FFh	R	未使用

计数器输入输出部分端口分布 (Page 2h)

Address	Read/Write种别	内容
00h - 01h	R	系统预留区域
02h - 03h	R	系统预留区域
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	系统预留区域
22h - 23h	W	系统预留区域
24h - 25h	R/W	系统预留区域
26h - 27h	R/W	系统预留区域
2Ah - FFh	R	未使用

数字量输入端口 (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

此端口可获取数字量输入端子的值。如果设置了数字滤波器，则获取通过滤波器之后的值。

※CPS-MC341-ADSCx系列只有DI0 - DI3有效。

数字量输出端口 (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

该端口可设置数字量输出端子的值或者获取设定值。

※CPS-MC341-ADSCx系列只有DO0 - DO1有效。

数字滤波器设置时间 (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

该端口可设置适用于数字输入端子的数字滤波器的值，也可获取设置的值。设定值适用于所有输入端子。有关设定值请参见<数字滤波器设置项目>。

数字滤波器设置项目

设置项目	名称	意义	设置项目	初始值
ST4～0	数字滤波器设定时间	设置数字滤波器的时间。	0：滤波功能未使用	0 [滤波功能未使用]
			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	

内置电源 ON/OFF 设定端口 (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

该端口可设置数字量输入端口用内置电源的有效(ON)/无效(OFF)。

通过Read这个端口，可以检查设置状态。有关设定值请参见<表 内置电源 ON/OFF设置>。

内置电源 ON/OFF设置

设置项目	名称	意义	设置项目	初始值
PWEn	内置电源有效	让内置电源有效(ON)。	0：无效(OFF) 1：有效(ON)	0 [无效]

模拟量输入端口 (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

此端口可获取模拟输入通道的值。如果需要通道间隔离功能，请不要同时ON两个开关。如果同时ON时，通道间隔离的功能将失效。

计数器数据读取端口 (Page 2h / Address 10h - 13h) 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

此端口可读取锁存的计数器数据。

读取数据在<计数器读取通道设置端口(Page 2h / Address 10h) W>中设置。

计数器读取通道设置端口 (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

此端口可选择要在计数器数据读取端口中读取的通道。

计数器数据的读取在<计数器数据读取端口(Page 2h / Address 10h - 13h) R>中进行。

计数器读取设置

设置项目	名称	意义	设置项目	初始值
Sel0	计数器读取 通道	设置从计数器数据读取端口读取的 通道。	0: Channel 0 1: Channel 1	0 [Channel 0]

计数器数据锁存设置端口 (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

将[1]写入此端口会锁存计数器数据。从计数器数据读取端口读取此处锁存的计数值。

计数器有效通道设置端口 (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

本端口写入时可设置计数器通道为有效或无效，读取时读出设置的状态。

计数器指令端口 (Page 2h / Address 18h) 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						

此端口是用于执行以下指令代码的操作指令端口。

指令代码一览：

08h: Ch0计数器方式 (Write)

09h: ch1计数器方式 (Write)

18h: Ch0比较寄存器0 (Write)

19h: Ch1比较寄存器0 (Write)

20h: Ch0比较寄存器1 (Write)

21h: Ch1比较寄存器1 (Write)

38h: 计数一致状态确认/清除 (Read/Write)

3Ah: 进位状态确认/清除(Read/Write)

3Dh: 软件清零 (Write)

使用Write指令时，向数据地址端口(Page 2h / 1Ch - 1Fh)设置数据。使用Read指令时，从数据地址端口(Page 2h / 1Ch - 1Fh)读取数据。控制了指令端口后，也请控制数据地址端口。关于各指令代码的数据地址端口的格式，请参照<计数器输入输出部分端口映射 (Page 2)> ~ <内置电源 ON/OFF设定端口 (Page 0h / Address 1Ch - 1Dh) R/W>。

Ch0 / Ch1计数器方式 (计数器指令代码：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

设置计数器的动作方式。设定是按每个输入通道进行的。

Ch0 / Ch1比较寄存器0 (计数器指令代码：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							

向Ch0 - Ch1的计数值比较寄存器0设置数据。

Ch0 / Ch1比较寄存器1 (计数器指令代码：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							

向Ch0 - Ch1的计数值比较寄存器1设置数据。

计数一致状态确认 / 清除 (计数器指令代码：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时，条件成立的位为1。

在Write时，通过将对应位设置为1来重置。

进位状态确认 / 清除 (计数器指令代码：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时，条件成立的位为1。

在Write时，通过将对应位设置为1来重置。

软件清零 (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

在Write时，通过将对应位设置为1来重置。

2.[堆栈组合型 CPS-MxS341-DSx系列]

厂家 : Lattice Semiconductor

设备型号 : LCMXO2-7000HC-4FTG256I

接口 : GPMC

端口映射

Address	Read/Write种别	内容
0000h - 0001h	R	系统预留区域
0002h	R	旋钮开关
0003h	R	DIP SW
0004h	R	连接设备台数
0005h	R/W	系统预留区域
0006h - 0007h	R/W	LED控制
0008h - 000Bh	R/W	系统预留区域
000Ch - 000Dh	R/W	系统预留区域
000Eh - 000Fh	R	未使用
0010h - 005Fh	R	堆栈设备信息
0060h - 0063h	R/W	DIO控制寄存器
0064h - 0065h	R/W	UART控制寄存器
0066h - 00FFh	R/W	未使用
0100h - 01FFh	R	设备0
0200h - 02FFh	R	设备1
: :		: :
1F00h - 1FFFh	R	设备30
2000h - 20FFh	R	设备31

4. 内置NOR FLASH内存配置

CONPROSYS具有32Mbyte内置NOR FLASH内存。

内存配置与<**向内置NOR FLASH安装用rootfs部分的复制(P38)**>项中所示的安装文件之间的关系如下所示。

NOR FLASH内存配置

Address	dev	内存大小	用途	安装文件
0000000h - 001FFFFh	mtd0	131,072 byte	主引导用的	MLO.byteswap
0020000h - 009FFFFh	mtd1	524,288 byte	u-boot用	u-boot.img
00A0000h - 00DFFFFh	mtd2	262,144 byte	u-boot选用	※1
00E0000h - 043FFFFh	mtd3	3,538,944 byte	内核用	uImage
0440000h - 0DBFFFFh	mtd4	9,961,472 byte	ramdisk用	ramdisk.xz
0DC0000h - 1FFFFFFF	mtd5	19,136,512 byte	应用区域	mtd5.tgz※2

※1 本SDK的默认安装工具将处理注释掉了。

※2 用展开的文件向mtd5安装。

5. 精巧一体型系列 LED/DIP Switch/Switch 控制

精巧一体型的LED可以通过GPIO端口按下表所示进行控制。

精巧一体型LED控制

LED种类	控制设备	端口No	端口属性	控制方法 (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

精巧一体型的Switch可以从GPIO端口读取下表所示的内容。

精巧一体型Switch控制

LED种类	控制设备	端口No	端口属性	控制方法 (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. 堆栈组合型系列 DIO/LED/DIP Switch/Switch控制

堆栈组合型的DIO / LED / DIP Switch / Switch 可由CONPROSYS的以下文件夹中的文件控制。

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

文件的功能和使用方式如<**堆栈组合型系列 DIO/LED/DIP Switch/Switch控制 (P91)**>所示。

堆栈组合型DIO / LED / DIP Switch / Switch控制

文件	控制设备	功能
	使用方法	
dio0_direction	DIO	DI/DO的切换设置
	b0(DIO0) - b3(DIO3) 如果是0设置为DI,如果是1设置为DO 设置举例: 将DIO0和DIO1设置为DI、DIO2和DIO3设置为DO b3:1, b2:1, b1:0, b0:0 → cH <Command> echo 0xc > /sys/bus/platform/drivers/cps-driver/dio0_direction 设置读取举例: <Command> cat /sys/bus/platform/drivers/cps-driver/dio0_direction	
dio0_do_value	DO	DO值设置
	设置举例: 将DO0和DO2设置为1、DO1和DO3设置为0 b3:0, b2:1, b1:0, b0:1 → 5H <Command> echo 0x5 > /sys/bus/platform/drivers/cps-driver/dio0_do_value 设置读取举例: <Command> cat /sys/bus/platform/drivers/cps-driver/dio0_do_value	
dio0_di_value	DI	DI值读取
	<Command> cat /sys/bus/platform/drivers/cps-driver/dio0_di_value	
id	旋钮开关	旋钮开关值读取
	<Command> cat /sys/bus/platform/drivers/cps-driver/id	
led_status1	Status1 LED	Status1 LED On/Off设置
	设置举例: Status1 LED をOn <Command> echo 1 > /sys/bus/platform/drivers/cps-driver/led_status1 设置读取举例: <Command> cat /sys/bus/platform/drivers/cps-driver/led_status1	
led_status2	Status2 LED	Status2 LED On/Off设置
	设置举例: Status2 LED をOff <Command> echo 0 > /sys/bus/platform/drivers/cps-driver/led_status2 设置读取举例: <Command> cat /sys/bus/platform/drivers/cps-driver/led_status2	
led_error	Error LED	Error LED On/Off设置
	设置举例: Error LED をOn <Command> echo 1 > /sys/bus/platform/drivers/cps-driver/led_error 设置读取举例: <Command> cat /sys/bus/platform/drivers/cps-driver/ switch	
switch	DIP Switch	DIP Switch值读取
	<Command> cat /sys/bus/platform/drivers/cps-driver/switch	

堆栈组合型的LED/Switch等，也可以使用本SDK应用示例程序中的CPS-MxS341-DSx系列用的iolib控制示例程序进行访问。

堆栈组合型的LED可以根据下表所示的FPGA的I/O映射地址从GPMC端口进行控制。

堆栈组合型LED控制

寄存器	D7	D6	D5	D4	D3	D2	D1	D0
0006h	-	-	-	-	ERR	ST2	ST1	Power
					R/W	R/W	R/W	R/W
					On: 1 Off: 0	On: 1 Off: 0	On: 1 Off: 0	On: 1 Off: 0

命令举例： 让Power、ST1、 ST2点亮

```
gpmc_testd -w1 0006 06
```

命令举例： 获取LED的状态

```
gpmc_testd -r1 0006
```

堆栈组合型的Switch可以从GPMC端口，根据下表所示的FPGA的I/O映射地址读取。

堆栈组合型Switch控制

寄存器	D7	D6	D5	D4	D3	D2	D1	D0
0002h	旋钮开关 H				旋钮开关 L			
0003h	DIP SW1-4	DIP SW1-3	DIP SW1-2	DIP SW1-1	-	-	-	-
	On: 1 Off: 0	On: 1 Off: 0	On: 1 Off: 0	On: 1 Off: 0	-	-	-	-

命令举例： 获取旋钮开关的状态

```
gpmc_testd -r1 0002
```

7. 选项板控制

在以下型号中，主机内置了3G/LTE/920 Hz通信选项板。

【精巧一体型　M2M控制器系列】

CPS-MC341G-ADSC1系列	多功能I/O + 3G(日本国内/全球)型
CPS-MC341Q-ADSC1	多功能I/O + 920MHz段通信型

【精巧一体型　M2M Gateway系列】

CPS-MG341G-ADSC1系列	多功能I/O + 3G(日本国内)型
CPS-MG341G5-ADSC1	多功能I/O + LTE型

【堆栈组合型　M2M控制器系列】

CPS-MCS341G-DS1	CPU模块 + 3G(日本国内)型
CPS-MCS341G5-DS1	CPU模块 + LTE型
CPS-MCS341Q-DS1	CPU模块 + 920MHz段通信型

【堆栈组合型　M2M Gateway系列】

CPS-MGS341G5-DS1	CPU模块 + LTE型

这些型号允许控制选项板的电源。

选项板控制

功能	控制方法 (linux shell)
选项板电源On※	/usr/local/cps-board/PowerOnOptionBoard.sh
选项板电源Off※	/usr/local/cps-board/PowerOffOptionBoard.sh
选项板检测	/usr/local/cps-board/DetectOptionBoard.sh [结束状态] 0:　选项板启动中 1:　选项板未检测到

※需要root权限。如果要在控制台中运行，请使用sudo命令执行。

3G/LTE型可以控制连接/断开、SIM检查、RSSI获取等。

3G/LTE控制

功能	控制方法 (linux shell)
连接※1	/usr/local/cps-board/mobile/start_mobile.sh
断开※1	/usr/local/cps-board/mobile/stop_mobile.sh
3G/LTE模块复位※1	/usr/local/cps-board/mobile/reset_mobile.sh
SIM检查	/usr/local/cps-board/mobile/checkSIM_mobile.sh [结束状态] 0: 有SIM 表示“Detect SIM” 1: 无SIM 表示“Not Detect”
RSSI获取	/usr/local/cps-board/mobile/checkSIM_mobile.sh [结束状态] 0: 成功 表示RSSI值(dbm) 1: 失败
RSRP获取(只有LTE型)	/usr/local/cps-board/mobile/getRSRP.sh [结束状态] 0: 成功 表示RSRP值(dbm) 1: 失败
选项板的LED控制※2	/usr/local/cps-board/mobile/ctrl_LED.sh param [param] 0: All off 1: Green On Red Off 2: Green OffRed On 3: Green On Red On [结束状态] 0: 成功 1: 失败

※1 需要root权限。如果要在控制台中运行，请使用sudo命令执行。

※2 对于CPS-MC341G-ADSC1-111以及CPS-MG341G-ADSC1-111型号，因为是3G模块控制，所以不能进行LED控制。

8. 目标搭载应用程序

主要搭载的应用程序

Application	Light rootfs NOR Flash version	Light rootfs SD version	Ubuntu 14.04	Ubuntu 14.04 with SDK
busybox	1.31.1	1.31.1	-	-
apt-utils	-		1.0.1	1.0.1
binutils	-		-	2.24-5
ncurses	-		-	5.9
apache	2.4.29※1	2.4.29※1	-	2.4.7-1
ssh server/client	dropbear 2019.78	dropbear 2019.78	open-ssh 6.6	open-ssh 6.6
NTP client	(busybox)	(busybox)	ntpdate 4.2.6	ntpdate 4.2.6
DHCP server	(busybox)	(busybox)	Udhcpd 1.21.0-1	isc-dhcp-server 4.2.4-7
DHCP client	(busybox)	(busybox)	isc-dhcp-client 4.2.4-7	isc-dhcp-client 4.2.4-7
Samba server	-	-	-	4.3.11
Samba client	-	-	-	4.3.11
Nfs Server	-	-	-	-
Nfs Client	-	-	-	-
gcc / g++	-	-	-	4.9.4-2
cmake	-	-	-	3.2.2-2
autoconf	-	-		2.69-6
automake	-	-		1.14.1-2
perl	-	-	5.18.2	5.18.2-2
python	-	-	3.4.3-1	3.4.3-1
php5	5.6.34※1	5.6.34※1	-	5.5.9
curl	7.59.0※1	7.59.0※1	7.35.0-1	7.35.0-1
wget	(busybox)	(busybox)	1.15	1.15-1
ftp server	(busybox)	(busybox)	-	vsftpd 3.0.2
ftp client	(busybox)	(busybox)	-	0.17
tftp server	(busybox)	(busybox)	-	-
tftp client	(busybox)	(busybox)	-	-
mail	(busybox)	(busybox)	-	-
iperf	-	-	-	-
minicom	-	-	-	-
ppp	2.4.7	2.4.7	2.4.5-5.1	2.4.5-5.1
pppoe	-	-	-	3.8-3
iptables	1.8.4	1.8.4	1.4.21-1	1.4.21-1
Wireless tool	29※1	29※1	30~pre9-8	30~pre9-8
wpa_supplicant	2.7※1	2.7※1	2.1-0	2.1-0
Open SSL	1.0.2n※1	1.0.2n※1	1.0.1f-1	1.0.1f-1
sudo	1.8.31p1	1.8.31p1	1.8.9p5-1	1.8.9p5-1
gdb	-	-	-	8.2

※1 选择项

修订履历

修订日期	修订内容
2021年11月	初版
2022年2月	Ver 1.4.3 - 在"目标动作确认 - 7.Web Setup功能 - 2.状态菜单"中增加以下项目 路由功能、IP过滤器、Log - "附录 - 2.设备I/F" 在堆栈组合型AIO控制设备中添加以下设备 CPS-AO-1604ALI - "附录 - 2.设备I/F" 在堆栈组合型DIO控制设备中添加以下设备 CPS-DI-16L、CPS-DI-16RL、CPS-DO-16L、CPS-DO-16RL、CPS-RRY-4PCC
2024年11月	Ver 1.5.0 -追加支持机型 CPS-MGS341-DS1 CPS-MGS341G5-DS1 - 添加支持的堆栈I/O CPS-SSI-4C

- 关于本书的内容，虽然已经做了仔细的确认，如有发现不妥之处或内容遗漏等情况，请联系经销商或技术支持中心。
- CONPROSYS是CONTEC CO.,LTD.的注册商标。其它书中使用的公司名称和产品名称一般是各公司的商标或注册商标。

CONTEC CO., LTD.

3-9-31, Himesato, Nishiyodogawa-ku, Osaka 555-0025, Japan

https://www.contec.com/

CONPROSYS Linux SDK用户手册(交叉开发版)

NA08630 (LXAJ433) 11222024_rev3 [11262021] November 2024 Edition