RZ/N2L

工业以太网协议自动检测

介绍

如今，各种工业以太网协议通常需要生产和交付具有不同软件风格的相同硬件，因为只有一个应用程序的单一方法很难处理。

本应用笔记介绍一种适用于 RZ/N2L 的简易引导加载程序解决方案，该解决方案可以检测所使用的工业以太网协议，并在运行期间将适当的应用程序从闪存加载到 RAM。灵活使用内部系统 RAM、外部 SDRAM 或外部超 RAM 支持不同的 RAM 消耗和执行速度要求。现有的以太网协议应用程序只需要对启动代码和链接器脚本进行少量更改。

下面列出了工业协议自动检测样本程序的主要功能。

- 在启动期间，将读取 RZ/N2L-RSK 板端口的传入以太网数据包，以找出所使用的以太网协议。以太网类型和源 MAC 地址用于此目的。
- 根据检测到的协议，相应的应用程序从闪存复制到 RAM 并从那里执行。

目标设备

RZ/N2L

将本应用笔记中的示例程序应用于另一台微型计算机时，请根据目标微型计算机的规格修改程序，并对修改后的程序进行广泛评估。

参考文件

[1] r01an6434ej0106-rzt2-rzn2-fsp-getting-started.pdf RZ/T2、RZ/N2 灵活软件包入门，版本 1.06

[2] r20ut4984eg0103-rskplus-rzn2l-v1-um.pdf
RZ/N2 luser 用户手册瑞萨初学者工具包+版

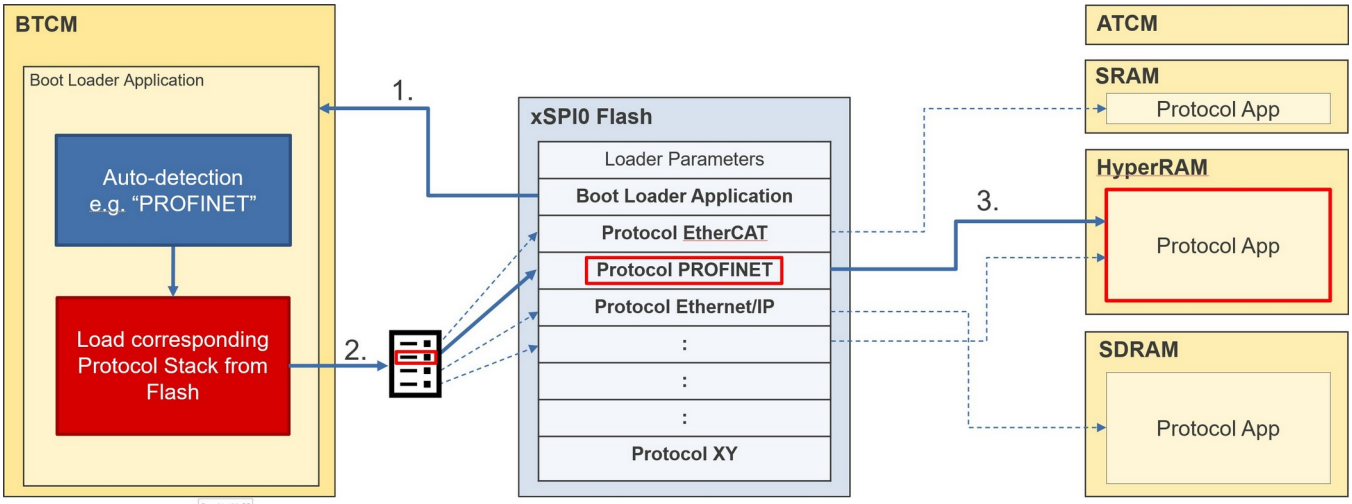
[3] 以太猫示例应用程序，快速入门指南:以太猫从属软件，r01an6178ej0120-rzn2l-ecat.pdf

内容

1. 概观

1.1 通过引导加载程序加载协议应用程序

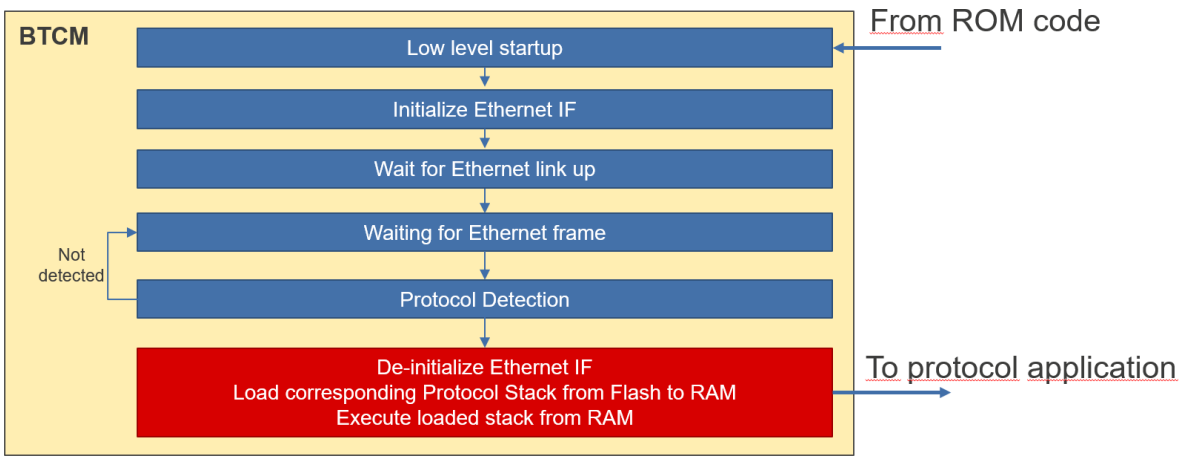
ROM 中的启动代码根据也存储在闪存中的加载程序参数将引导加载程序应用程序从闪存复制到内部 RAM。启动引导加载程序。工业以太网协议检测如下文详细描述的那样执行。根据结果，引导加载程序将相应的代码和数据从闪存复制到 RAM，并最终从该位置启动应用程序。



用户必须定义 RAM 的类型并指定每个协议应用的位置。引导加载程序应用程序中的一个表保存了闪存源地址、RAM 目的地址和应用程序的大小。根据应用程序的链接程序脚本，在链接程序过程中会自动填充所需的值。

1.2 以太网协议自动检测

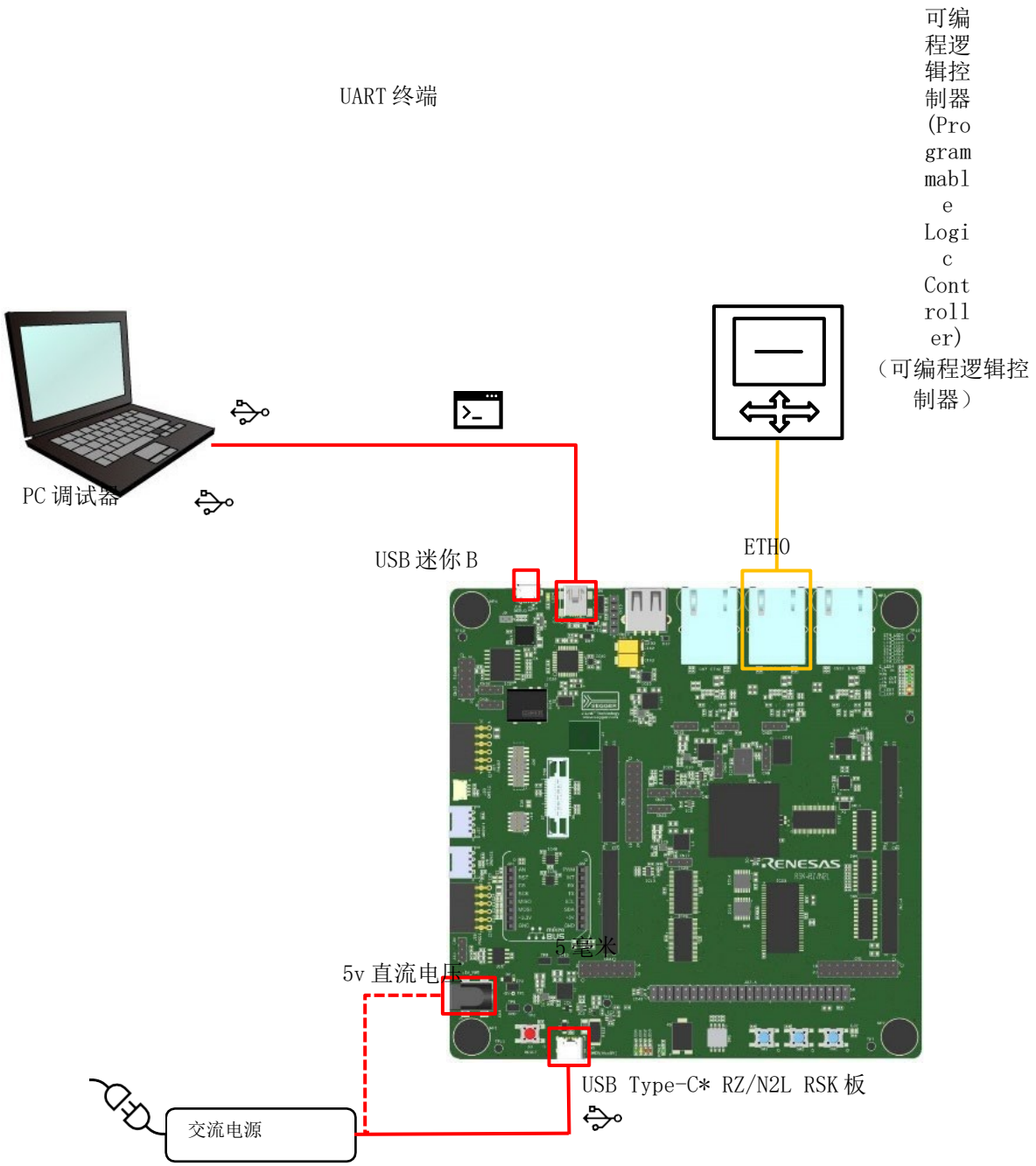
在低级启动之后，引导加载程序初始化硬件以打开以太网接口。该软件评估传入以太网数据包的以太网类型字段，以检测网络中使用的协议。如果接收到一定数量的协议帧，引导加载程序会将相应的应用程序从闪存复制到 RAM 并开始执行。以前的硬件初始化被还原。



2. 规格和操作环境

2.1 系统概况

下图显示了 RZ/N2L RSK 板与 PC、PCL 和电源的连接。



2.2 操作环境

本应用笔记涵盖的示例代码适用于以下硬件和软件环境。

表 1.1 要求

项目	小贩	描述
板	瑞萨电子	瑞萨初学者工具包+适用于 RZ/N2L（RTK 9 rz N2 l 0s 00000 be）
集成驱动电子设备	瑞萨电子	• e 工作室 2023-04 • RZ/N2L 灵活软件包（FSP）1 . 2 . 0 版 • ARM 工具链 9.3.1.20200408
仿真器	塞格	硬件:J-Link（板载）软件:J-Link V7.80b
软件 PLC	贝克霍夫自动化 CODESYS 发展有限公司	TwinCAT3 CODESYS V3.5

2.3 电路板设置

本节介绍该工业以太网协议自动检测应用用例所需的 DIP 开关和跳线设置。引导模式设置为 xspi 0（x1 引导串行闪存）。
有关主板的详细信息，请参考 RZ/N2L RSK 主板用户手册和原理图。

DIP 开关 SW4

一	2	3	四	5	6	七	8
在…上	在…上	在…上	在…上	离开	离开	在…上	离开

SW8

一	2	3	四	5	6	七	8	9	10
离开	在…上	离开	在…上	离开	离开	离开	离开	离开	离开

SW11

一	2	3	四	5	6	七	8	9	10
在…上	离开	离开	离开	离开	离开	离开	离开	离开	离开

针织套衫

表 1.2 瑞萨初学者工具包+用于 RZ/N2L 跳线设置

跳线号	环境
CN8	短 2-3
CN17	短 1-3
CN20	短 1-2
CN21	短 1-2
CN22	短 1-2
CN24	短 2-3
CN25	短 1-2
CN27	短 1-2
CN29	短 1-2
CN31	短 1-2
CN32	短 1-2
J9	打开

2.4 串行终端

调试输出串行终端的设置如下:

- 速度 115200 波特
- 数据 8 位
- 无奇偶校验
- 1 个停止位
- 无流量控制

2.5 软件元素

该软件由基于 RZ/N2L FSP 的多个软件元素组成，如下所示。
有关单个软件包的详细说明，请参考软件包文档。

2.5.1. 引导加载程序软件

RZ/N2L 集团分离加载程序和应用程序项目的示例 1.1.0 版

此外，引导加载程序初始化以太网接口并检查传入以太网帧的协议。根据协议，相应的应用程序二进制文件从 SPI 闪存复制到指定的 RAM 位置，并从该位置启动。

2.5.2. 工业协议自动检测程序

自动检测程序是一个基于 RZ/N2L FSP 的简单程序。该程序能够通过读取传入数据包的以太类型来检测一些市场领先的工业以太网协议（PROFINET、EtherCAT、以太网/IP）。

2.5.3. 以太猫

瑞萨 RZ/N2L 集团 EtherCAT 示例程序包 1.2.0 版修改应用于 startup.c 和链接器脚

本中的低级代码。

2.5.4. PROFINET

港口工业自动化有限公司的 GOAL Profinet 演示

（https://www.port.de).RZ/N2L:2020 年 4 月 2 日（更新日期:2023 年 9 月 29 日）

https://portgmbh . atlassian . net/wiki/spaces/GOAL r/pages/639762433/GOAL+-+瑞萨+RZ+N2L-RSKGOAL Profinet 演示

该演示是一个全功能协议栈，但有 1 小时的时间限制。选定的演

示代码:“01_simple_io”

修改应用于 startup.c 和 system.c 中的低级代码以及链接器脚本 fsp_ram_execution.ld .原始文件的扩展名为。fsp120 或。goal。

2.5.5. 以太网/IP

港口工业自动化有限公司的目标以太网/IP 演示（https://www.port.de).RZ/N2L:2020

年 4 月 2 日（更新日期:2023 年 9 月 29 日）

https://portgmbh . atlassian . net/wiki/spaces/GOAL r/pages/639762433/GOAL+-+瑞萨+RZ+N2L-RSK 目标以太网/IP 演示

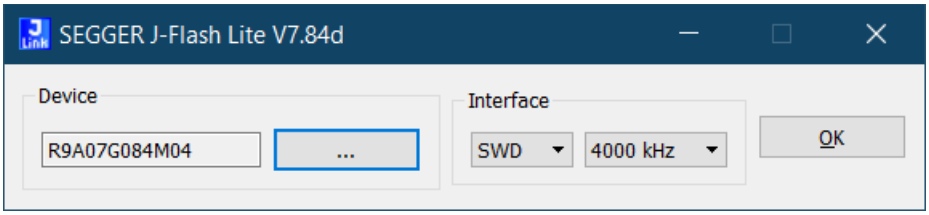
该演示是一个全功能协议栈，但有 1 小时的时间限制。选定的演

示代码:“10_led_demo”

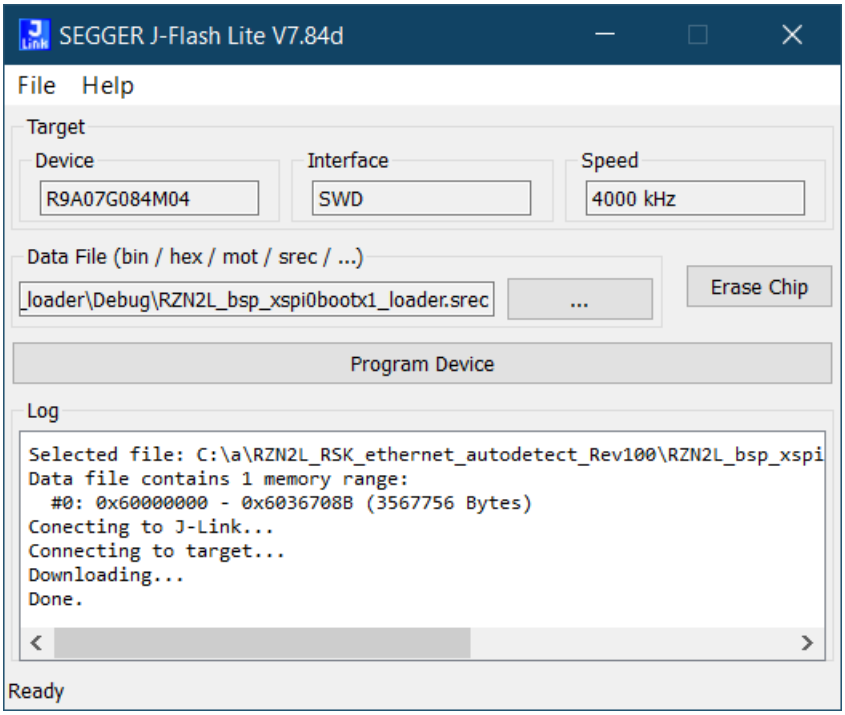
修改应用于 startup.c 和 system.c 中的低级代码以及链接器脚本 fsp_ram_execution.ld .原始文件的扩展名为。fsp120 或。goal。

3. 以二进制形式运行示例应用程序

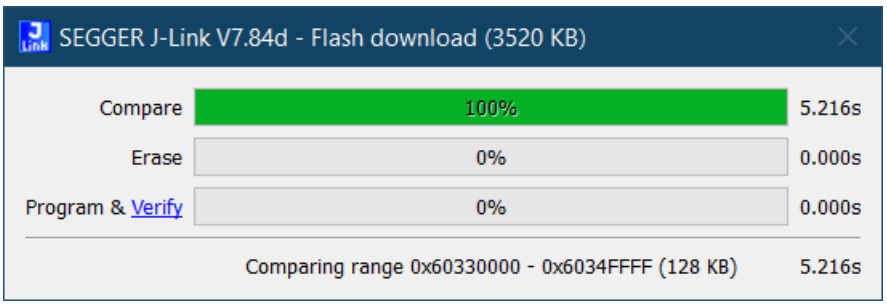
打开 SEGGER J-Flash Lite 工具，并选择 RZ/N2L 设备“R9A07G084M04”：



在文件夹 RZN2L _ RSK _以太网_自动检测_ rev 100 \ RZN2L _ bsp _ xspi 0 bootx 1 _ loader \ Debug 中选择整个应用程序 RZN2L _ bsp _ xspi 0 bootx 1 _ loader . srec 的 s 记录二进制文件。
然后按“程序设备”：



在比较、擦除和闪烁期间，将出现以下窗口：

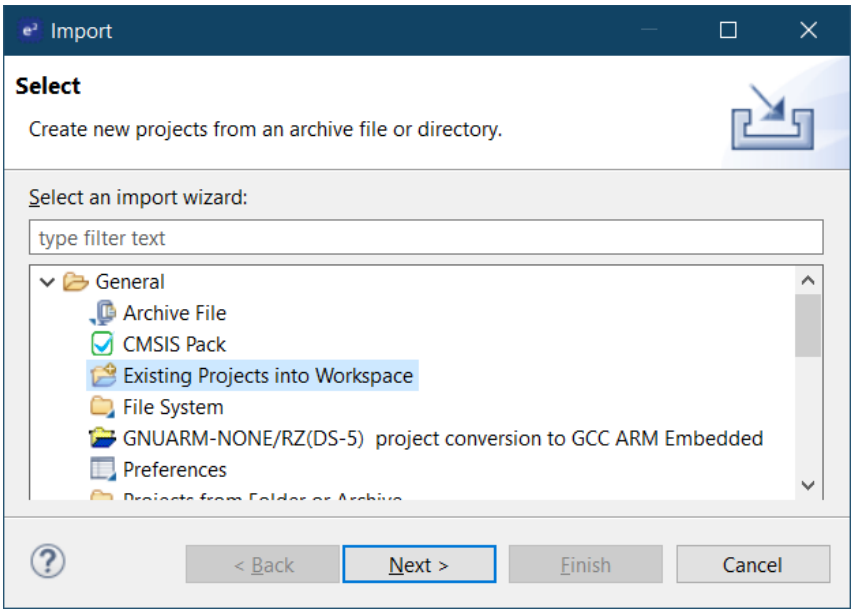


继续下一章 5.

4. 从源代码运行示例应用程序

4.1 软件安装

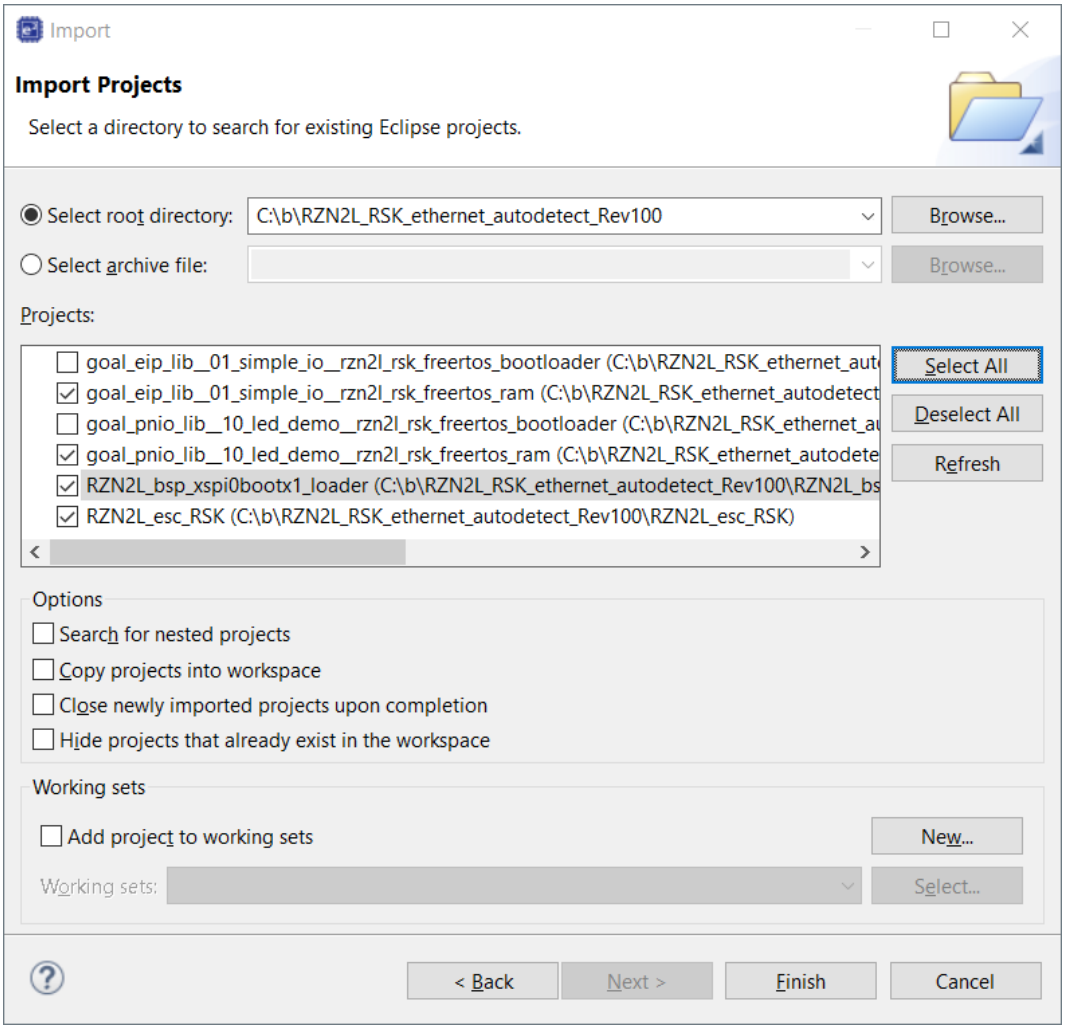
1. 创建一个新文件夹。此文件夹的路径应该很短，以免超过 Windows 操作系统的路径长度限制。
2. 将包 RZN2L _ RSK _以太网_自动检测_Rev100.zip 解压到文件夹中。
3. 启动 e 工作室。
4. 选择创建的文件夹（1）作为新工作区。
5. 选择【文件】-【导入】-【将现有项目导入工作区】。按下【下一步】。



6. 选择【选择根目录】，单击【浏览】按钮，并从（1）中选择文件夹。

7. 选择以下四个项目：

- RZN2L_bsp_xspi0boot x1_loader
- RZN2L _ esc
- goal _ EIP _ lib 01 _ simple _ io _ _ rzn2l _ freertos _ ram
- goal _ pnio _ lib _ _ 10 _ led _ demo _ _ rzn2l _ rsk _ freer tos _ ram



8. 单击“完成”按钮。
9. FSP 文件已经创建并可用。不需要调用 FSP 智能配置器。

注意:如果 FSP 文件将再次使用 FSP 智能配置器生成，请在此之前复制链接器脚本和文件 startup.c。对这些文件进行了修改以支持加载程序方法。生成的文件可能会在生成过程中被覆盖。

4.2 EtherCAT 源文件

由于许可条件的限制，不可能在此包中提供 Beckhoff 的 EtherCAT 源代码。请参考【3】第 3.3 章“生成从堆栈代码”如何生成所需的源代码。
必须将生成的文件复制到文件夹中。\ RZN2L _ ESC _ RSK \ Src \ ether cat \ beckhoff \ Src。

4.3 构建应用程序

链接器将协议应用程序的二进制文件添加到引导加载器应用程序。因此，必须首先构建协议应用程序。

1. 首先构建三个以太网协议应用程序。
   选择每个应用程序，从 e studio 菜单中选择【项目】 - 【全部构建】。

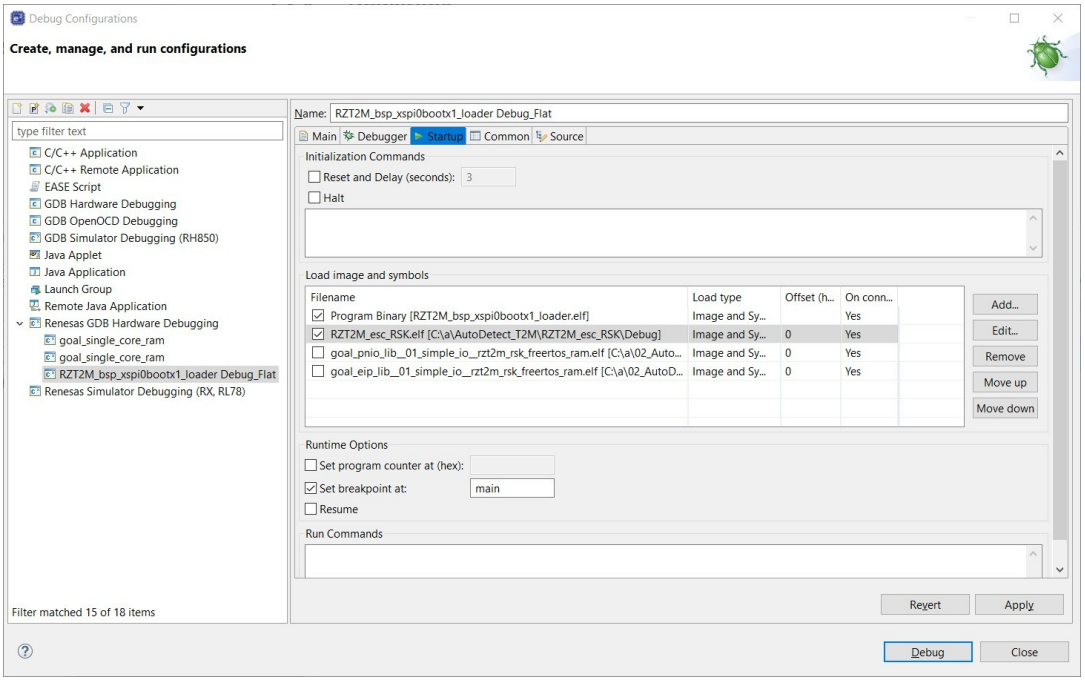
2. 在成功构建这些应用程序之后，构建加载器应用程序。

4.4 调试应用程序

对于源代码调试，调试器必须将源代码映射到当前执行的二进制文件。通常，这种映射在运行时是固定的。然而，这里在运行时选择加载的以太网协议应用。源代码因检测到的以太网协议而异。必须指示调试器应该调试哪个应用程序。

这可以在 e studio 对话框【调试-配置】 - 【瑞萨 GDB 硬件调试】 - 【RZN2L _ bsp _ xspi 0 bootx 1 _ loader _ Debug _ Flat】 - 【启动】中选择。

一次仅启用【加载图像和符号】的一个应用程序进行调试:



在项目浏览器中选择项目【RZN2L_bsp_xspi0bootx1_loader】作为活动项目，例如双击。（由于缺少初始化，直接启动其中一个应用程序将失败。）

使用【运行】 - 【调试】开始调试。

应用程序二进制文件被下载到 RZ/N2L RSK 板并启动。断点将在以下位置命中 system_init 和 main 函数。在这种情况下，请按【运行】 - 【恢复】继续。

5. 测试

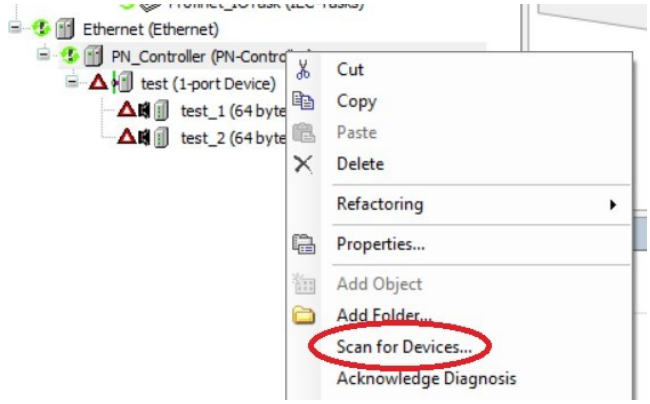
如第章所示，将电路板连接到您的电脑 2.1. 启动 TeraTerm 等终端软件，并按照第章中的说明设置接口参数 2.4.

安装自动化和控制软件，如 TwinCAT（支持 EtherCAT）或 CodeSys（支持 EtherCAT、PROFINET 和以太网/IP），以检查连接并控制电路板。请参考相应的示例应用程序和相关文档，了解如何设置 TwinCAT 和 Codesys 以使用原始示例应用程序。

按照上述文档中的说明，启用 TwinCAT/CodeSys 的操作模式。在 TwinCAT 中，这意味着“在配置模式下重启 TwinCAT 系统”。在 CodeSYS 中，这意味着启动网关、PLC 并执行登录和启动。

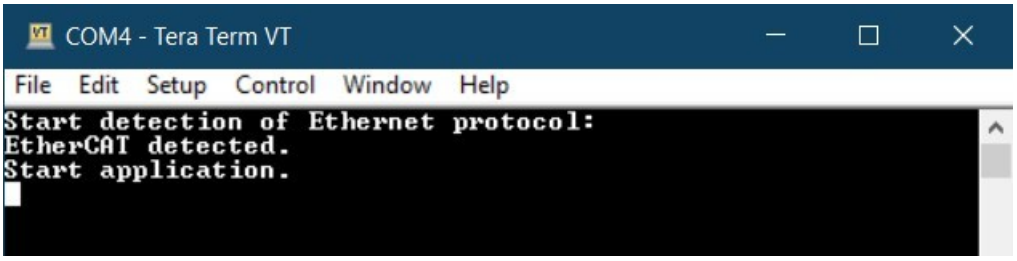
通过调试器或从闪存启动自动检测软件。串行终端应显示输出“开始检测以太网协议”。

要触发目标协议数据包的生成，可能需要停止并启动工具。在 CodeSys 中调用“扫描设备”很有帮助:



如果自动检测软件已经接收到足够的一种协议的帧，则给出找到的协议，
例如“检测到 EtherCAT”。相应的应用程序从闪存复制到 RAM 并启动。相应地，“启动应用程序”被打印到终端。

例如，对 EtherCAT 的成功检测应该如下所示:



注意:

以太网/IP 使用普通 IP 数据包的以太类型值。这意味着软件无法直接区分以太网/IP 和其他流量。因此，如果使用 Rockwell 设备，则可以使用 MAC 供应商 ID（也称为“组织唯一标识符”（OID））作为第二个标准。

在使用 PC 的测试环境中，MAC 具有任意的 OID，因此自动检测软件禁用了对 MAC 的检查。可以在加载器应用程序的 ead_app.c 中启用和配置对罗克韦尔或专用 OID 的检查。

如果不考虑 MAC 地址，典型的 PC 会连续发送各种被解释为以太网/IP 的帧。存在以太网/IP 被错误检测的风险。为了克服这个问题，成功检测以太网/IP 的要求是 25 帧，中间没有任何 EtherCAT 或 Profilnet 帧，但在极少数情况下，自动检测可能会因检测到错误的协议而失败。

5.1 使用以太网测试仪

要使用以太网测试仪测试以太网协议的自动检测功能，必须按如下方式准备要发送的帧:

草案	以太类型	源 MAC 地址
以太猫	0x88A4	（不在意）
PROFINET	0x8892	（不在意）
以太网/IP	0x0800	BC:F6:85:xx:xx:xx（罗克韦尔 OID 公司，如果启用检查）
Powerlink	0x88AB	（不在意）

5.2 无以太网流量测试

为了在没有以太网流量的情况下测试以太网协议应用程序的加载，可以注释掉检测，并且可以在加载器应用程序的 hal_entry.c 中的函数 hal_entry（）中将协议设置为固定值。

例如，在不检测的情况下加载 PROFINET，代码应该如下所示:

```
/*启用以太网并检测协议*/
//printf（“开始检测以太网协议:\ n“）;
//detected _ protocol = EAD _ main（）;

/*选择专用应用程序进行测试*/
//detected _ PROTOCOL = e _ PROTOCOL _ ether cat;
detected _ PROTOCOL = e _ PROTOCOL _ ProFi net;
//detected _ PROTOCOL = e _协议_ ENIP;
printf（“以太网协议%d 已显式设置。\n“，检测到的_协议）;
```

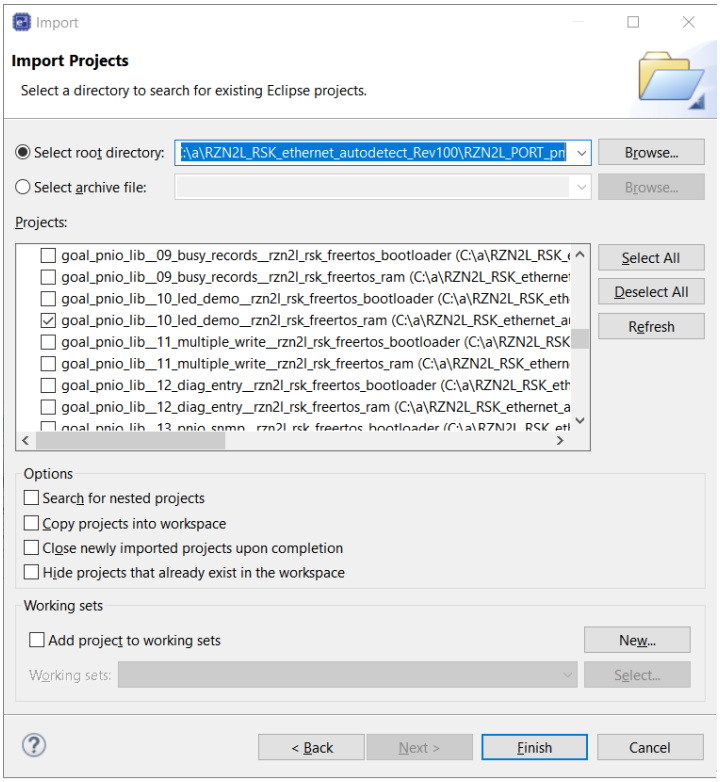
6. 综合

本章介绍如何将应用程序添加到加载程序中。作为示例，使用了 port GmbH 的 PROFINET 堆栈。

6.1 将示例应用程序导入工作区

在自动检测应用程序的现有工作区中创建一个新的子文件夹，例如 PROFINET 应用程序的“RZN2L_PORT_pn”。将提供的 zip 文件的内容提取到该文件夹中。

单击“文件”-“导入”，并在“常规”部分选择“将现有项目导入工作区”。输入创建的文件夹作为根文件夹，并取消选择所有找到的项目，除了一个示例的应用程序，例如“10_led_demo”。（不需要加载程序）。



6.2 生成 FSP 文件

在文件夹 projects \ goal _ pnio _ lib \ 10 _ led _ demo \ e2studio \ renesas \ rzn2l _ rsk _ freer tos \ ram \ script 中制作现有链接器脚本“fsp_ram_execution.ld”的副本。此文件包含一些改编内容，在使用智能配置器生成文件时将被默认脚本覆盖。

在项目浏览器中双击 configuration.xml 打开智能配置器。点击右上角的按钮生成文件。

用副本替换生成的链接器脚本。

6.3 设置 e studio 环境

要将应用程序包含到加载程序中，类型必须是纯二进制。在 e studio 中，选择【项目】-》【C/c++项目设置】-》【Cross ARM GNU Create Flash Image】-》【常规】。选择“输出文件格式”为“原始二进制”。

6.4 改编低级代码

必须调整文件夹《app》\ e2studio \ RENESAS \ rzn21 _ rsk _ freer tos \ ram \ rzn \ FSP \ src \ bsp \ CMS is \ Device \ RENESAS \ Source 中的文件 startup.c 和 system.c，以适应启动加载程序已经完成的初始化。现有的文件可以作为参考。

生成应用程序项目。

6.5 将应用程序添加到加载程序项目中

创建一个。S 文件，将应用程序的二进制文件包含到加载程序中。

```
。section . app 2 _ IMAGE _ QSPI _ FLASH _ section, “ax“, %progbits
。incbin“../../RZN2L _ PORT _ pn/projects/goal _ pnio _ lib/10 _ led _
demo/e2studio/renesas/RZN2L _ rsk _ freer tos/ram/单核/goal _ pnio _ lib 10 _ led _ demo RZN2L _
rsk _ freer tos _ ram . bin “
```

6.6 加载程序的链接器脚本

在链接器脚本 fsp_xspi0_boot_loader.ld 中添加指令以指定闪存和 RAM 中的位置:

```
。app 2 _ IMAGE _ SYSRAM 0x 30000000:AT（0x 30000000）
{
    APP2_IMAGE_SYSRAM_start =。;
    KEEP（*（app 2 _ IMAGE _ SYSRAM））
}


。app 2 _ IMAGE _ QSPI _ FLASH _ section 0x 60200000:AT（0x 60200000）
{
    app 2 _ IMAGE _ QSPI _ FLASH _ section _ start =。; 保持（。/src/FLASH _ section _ pn . o（（app
    2 _ IMAGE _ QSPI _ FLASH _ section））app 2 _ IMAGE _ QSPI _ FLASH _ section _ end =。;
}
app 2 _ IMAGE _ QSPI _ FLASH _ section _ size = SIZEOF（。app 2 _ IMAGE _ QSPI _ FLASH _ section）;
```

6.7 将新应用程序添加到加载器表中

链接器脚本的符号在源代码中用于定义源地址、目的地址和应用程序二进制文件的大小。Add in loader_table.c:

```
extern uint 32 _ t app 2 _ IMAGE _ QSPI _ FLASH _ section _ start;
extern uint 32 _ t app 2 _ IMAGE _ QSPI _ FLASH _ section _ size;

extern uint 32 _ t app 2 _ IMAGE _ SYSRAM _ start;

#定义应用程序 2 _ PRG _ FLASH _ addr（& app 2 _ IMAGE _ QSPI _ FLASH _ section _ start）
#定义应用产品 2 _ PRG _ start _ addr（& app 2 _ IMAGE _ SYSRAM _ start）
#定义应用程序 2 _ PRG _ size（& app 2 _ IMAGE _ QSPI _ FLASH _ section _ size）

const LOADER _ TABLE TABLE【TABLE _ ENTRY _ NUM】BSP _ PLACE _ IN _ SECTION（“CPU 0 _ LOADER _
TABLE“）
{
  {（uint32_t *）应用程序 1_prg_flash_addr,（uint32_t *）应用程序 1_prg_start_addr,（uint 32
          _ t）应用程序 1_prg_size,（uint 32 _ t）TABLE _ ENABLE },
  {（uint32_t *）应用程序 2_prg_flash_addr,（uint32_t *）应用程序 2_prg_start_addr,（uint 32
          _ t）应用程序 2_prg_size,（uint 32 _ t）TABLE _ ENABLE },
  {（uint 32 _ t *）TABLE _ INVALID _ VALUE,（uint 32 _ t *）TABLE _
          INVALID _ VALUE,（uint 32 _ t）TABLE _ INVALID _ VALUE,（uint
          32 _ t）TABLE _ DISABLE },
```

```
  {（uint32_t *）表_无效值，（uint32_t *）表_无效值，（uint32_t）表_无效
          值，（uint 32 _ t）表_禁用}
};
```

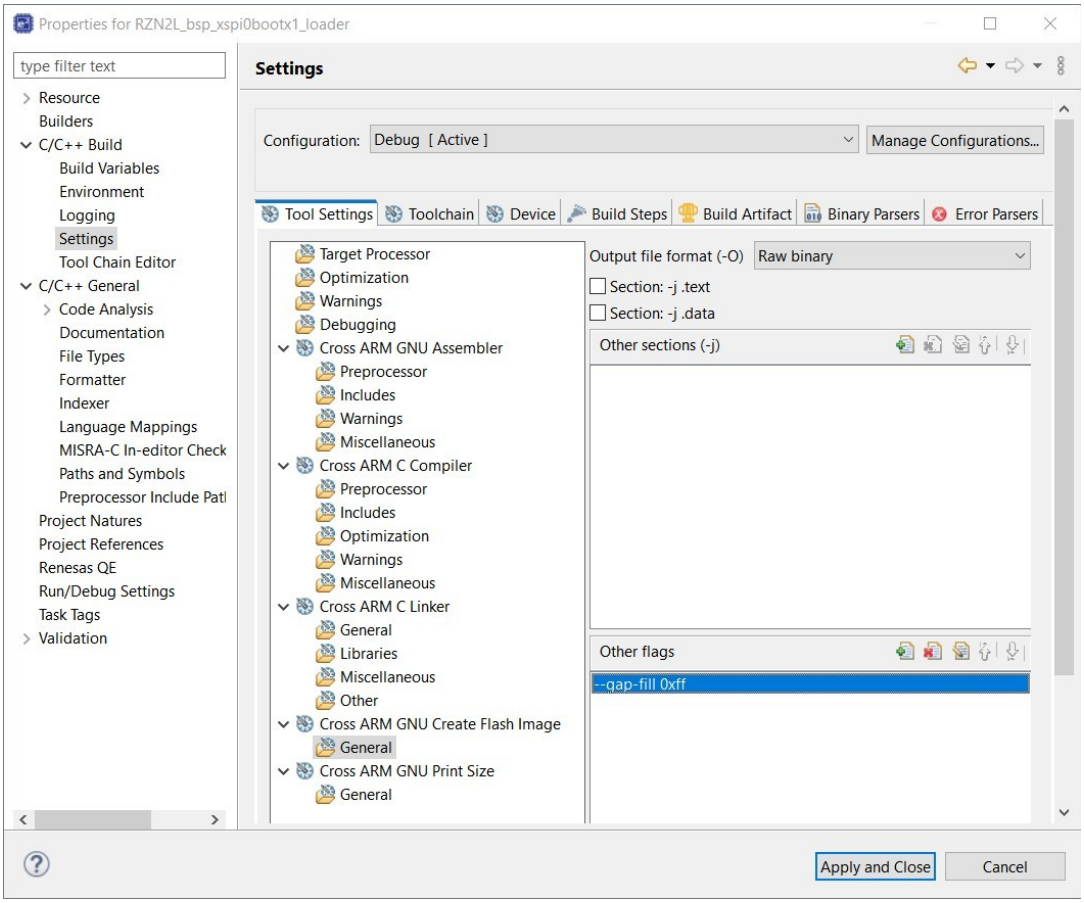
6.8 e 工作室

为了构建和调试，e studio 中需要进行两项更改。

6.8.1. 输出原始二进制文件

链接器将编译后的应用程序文件包含到加载器应用程序中需要二进制格式。

在应用程序的项目中，将输出格式更改为原始二进制。该二进制文件包含在。加载程序的 s 文件。



6.8.2. 为应用程序加载符号

在加载程序的项目中，转到菜单【运行】-【调试配置】。为新应用程序添加一个条目，并从工作区中选择 ELF 文件。

7. 修订历史

修订本	日期	描述
1.00	2023 年 12 月 18 日	第一版发行。