Mailbox Client IP サンプル説明資料

株式会社 マクニカ

Agenda

1. 準備
2. ハードウェア構成
3. フローチャート
4. サンプル使用時注意点
5. サンプル動作説明

準備

MACNICA

準備

- **動作環境**
  - インテル® Quartus® Prime 開発ソフトウェア: v21.3 Pro Edition
  - デバイス: インテル® Stratix® 10 FPGA
  - ボード:インテル® Stratix® 10 SX SoC Development Kit

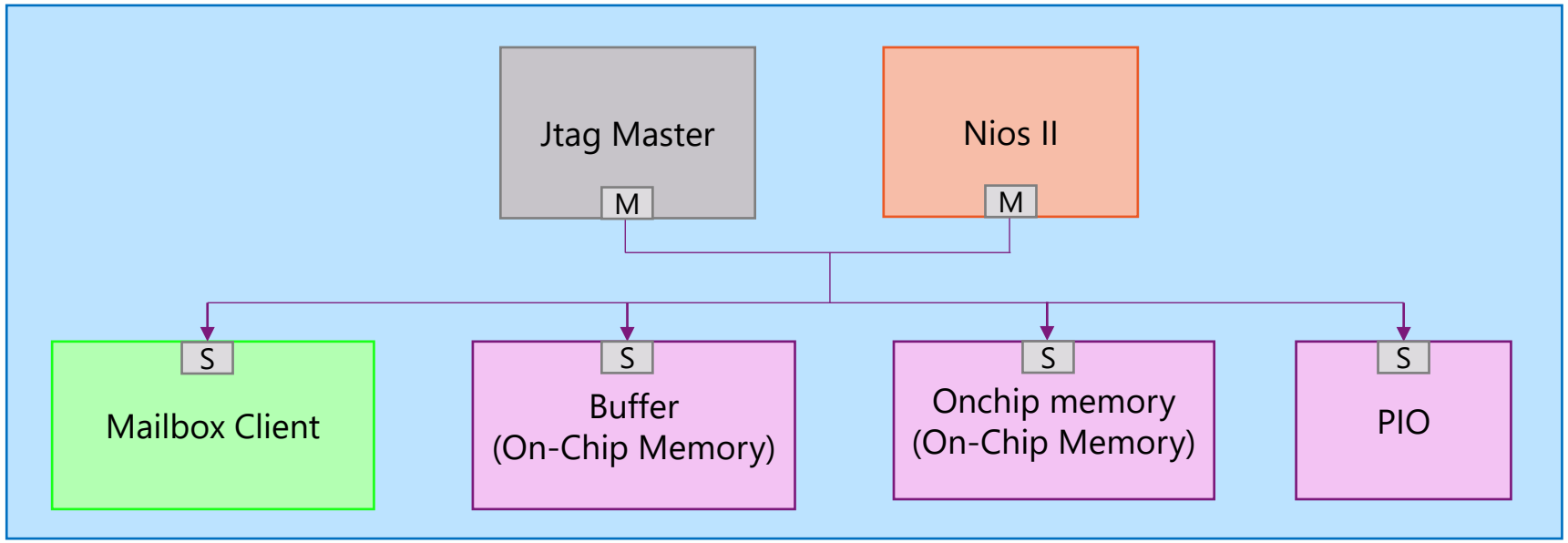
- **提供物**
  - mb_hw.qar
    - ハードウェアデザイン
  - mb_sample.c
    - C 言語ソースコード

ハードウェア構成

MACNICA

ハードウェア構成

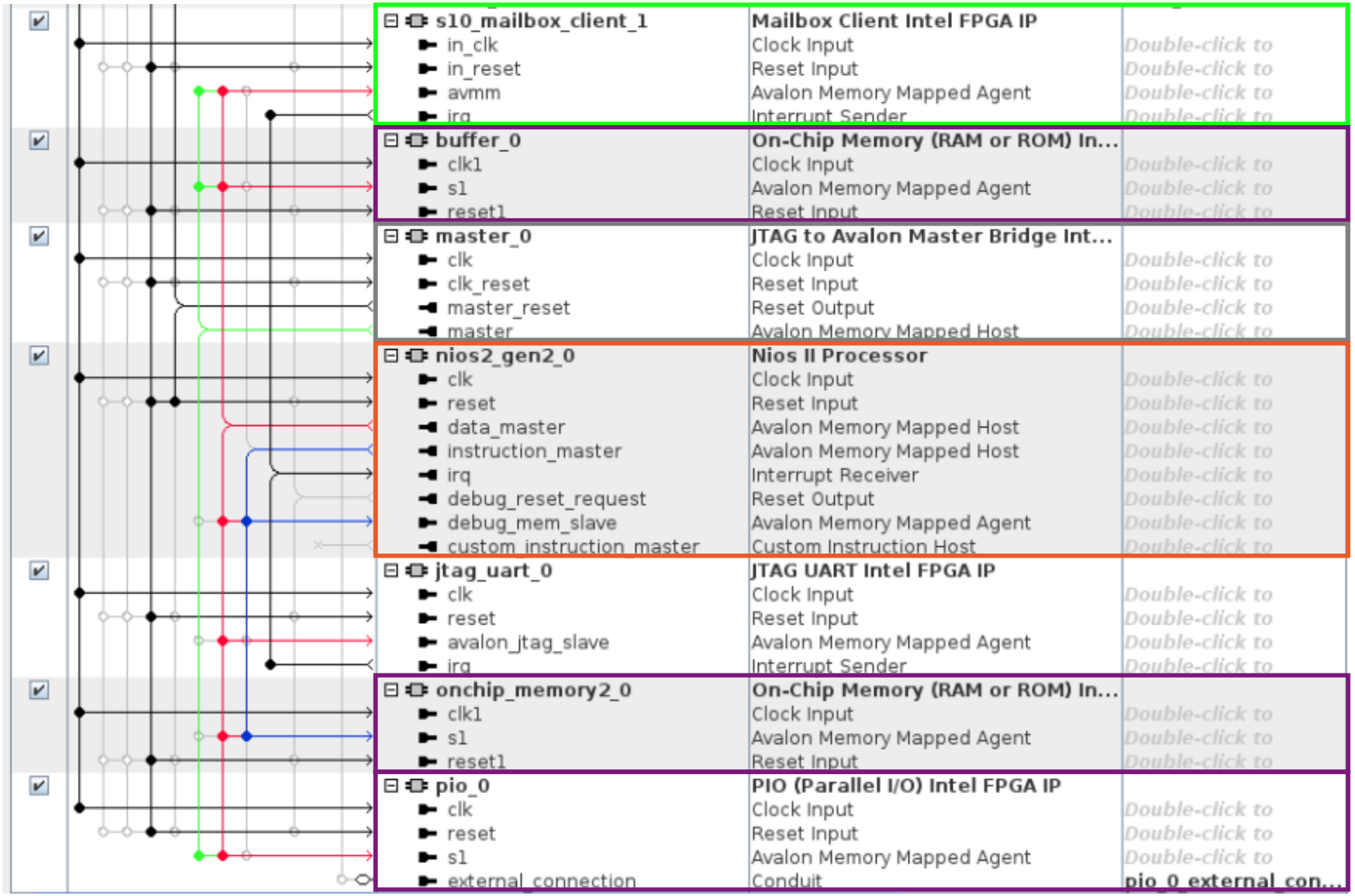
- ブロック図



  - 主要ブロック役割
    - Nios II: Slave の制御
    - Mailbox Client IP: SDM へのアクセス
    - Buffer: リード/ライトデータを一時的に格納するバッファー
    - Onchip memory: Nios II のワークスペース

ハードウェア構成

- Platform Designer



Mailbox Client

Buffer

Jtag Master

Nios II

On Chip RAM

PIO

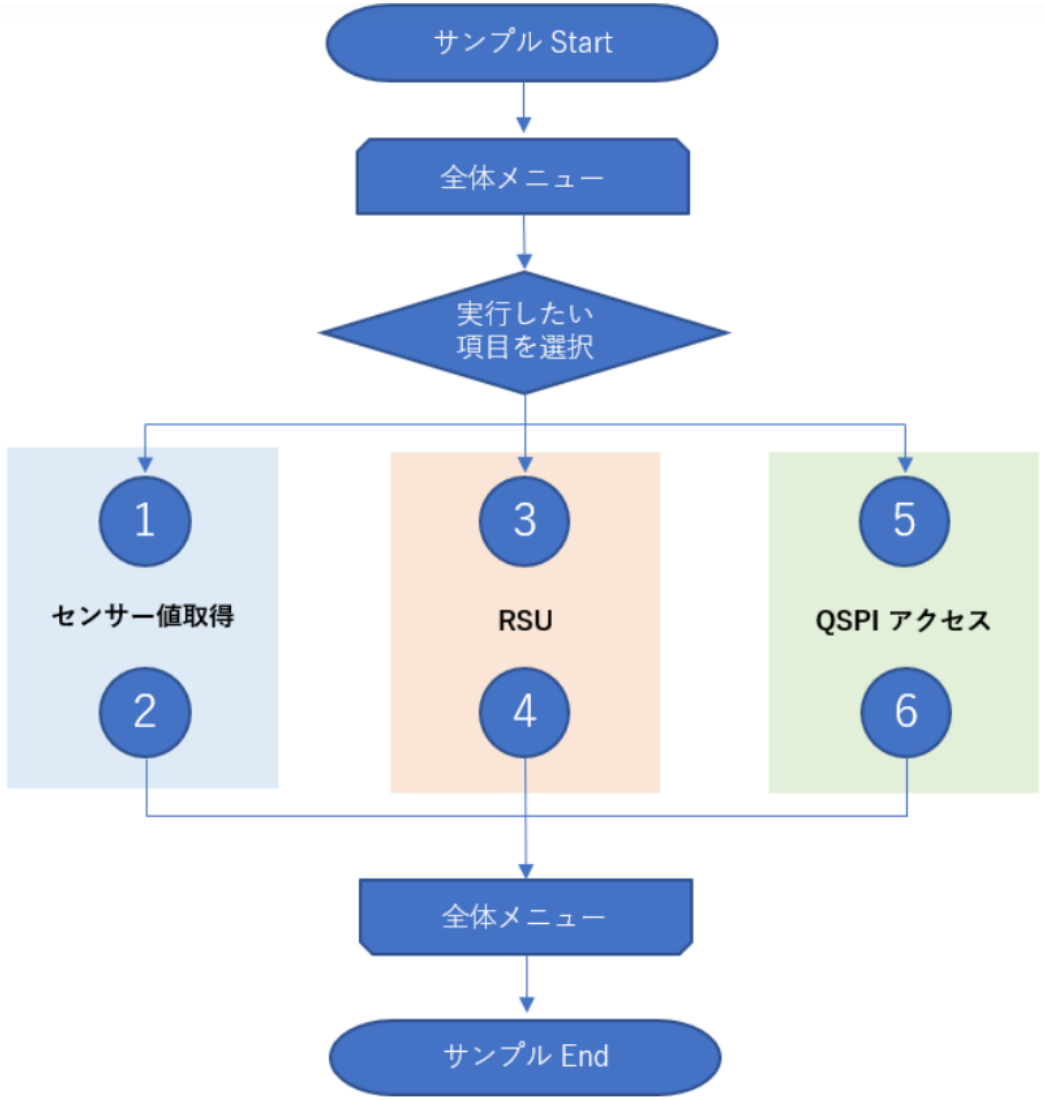
フローチャート

MACNICA

フローチャート

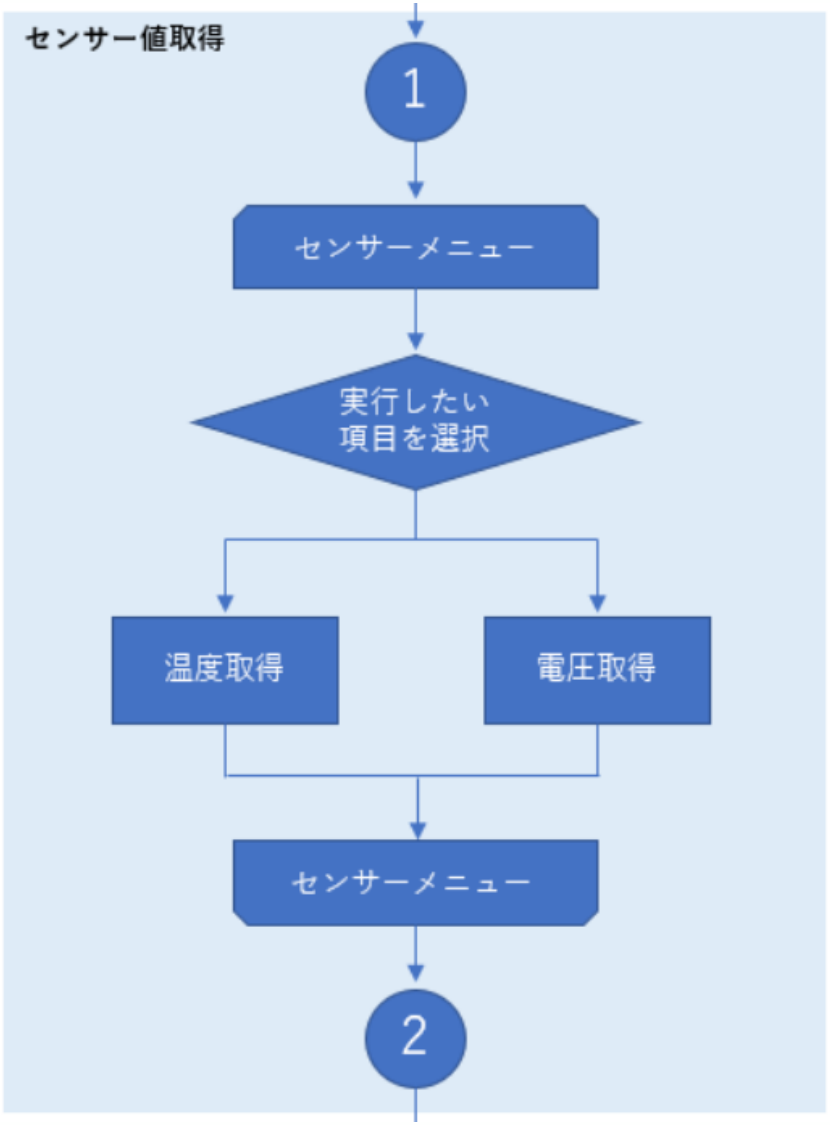
- **サンプル全体のフローチャート**
  - サンプルを実行すると、
    下記メニューが表示されます
    - センサー値取得
    - RSU
    - QSPI アクセス

※センサー値取得/RSU/QSPI アクセスの、
それぞれのフローチャートは次のページ以降に記載



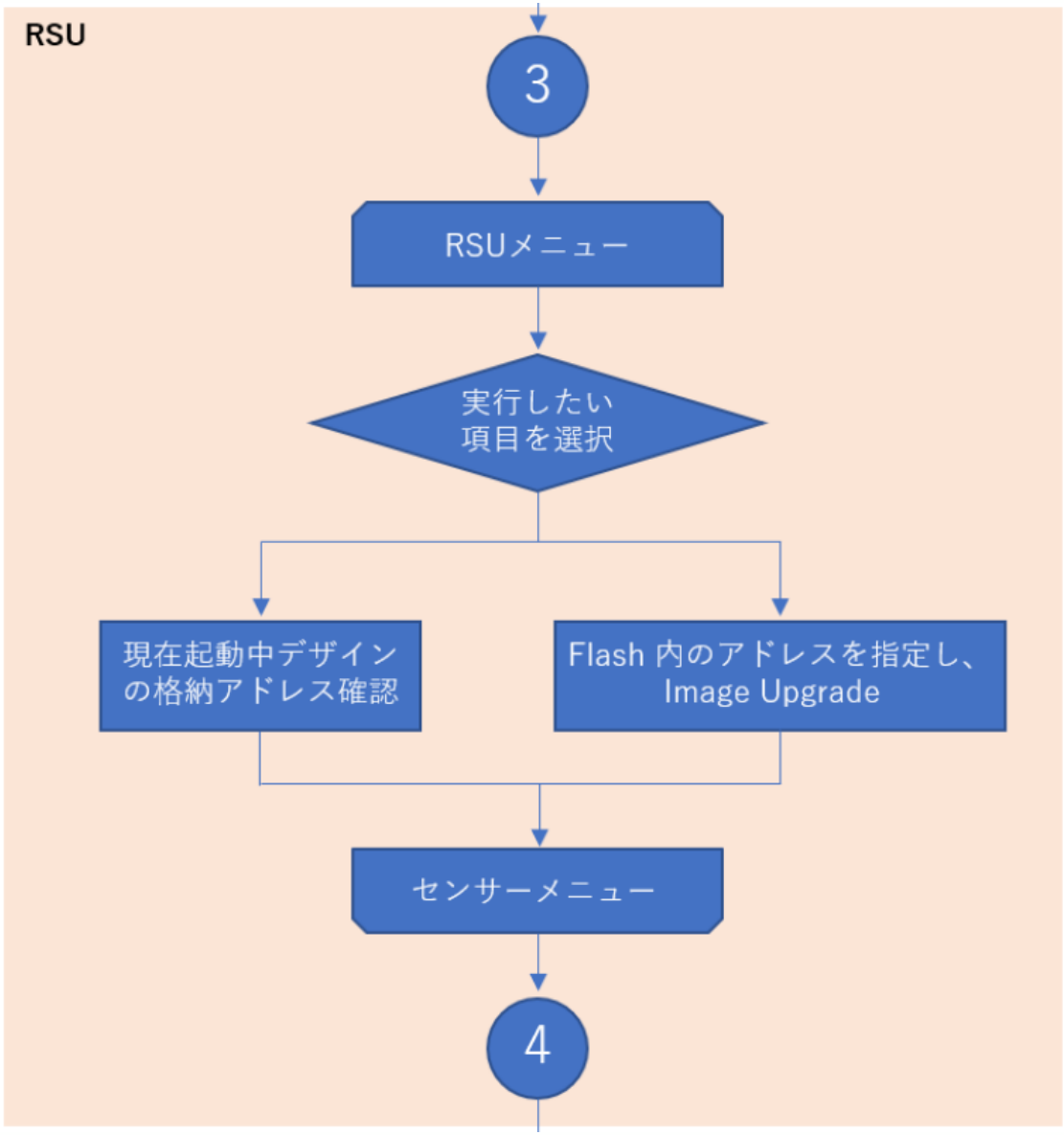
フローチャート

- **センサー値取得**
  - デバイス内部には、
    温度センサーと電圧センサーが内蔵
  - センサーメニュー
    - 温度取得
    - 電圧取得



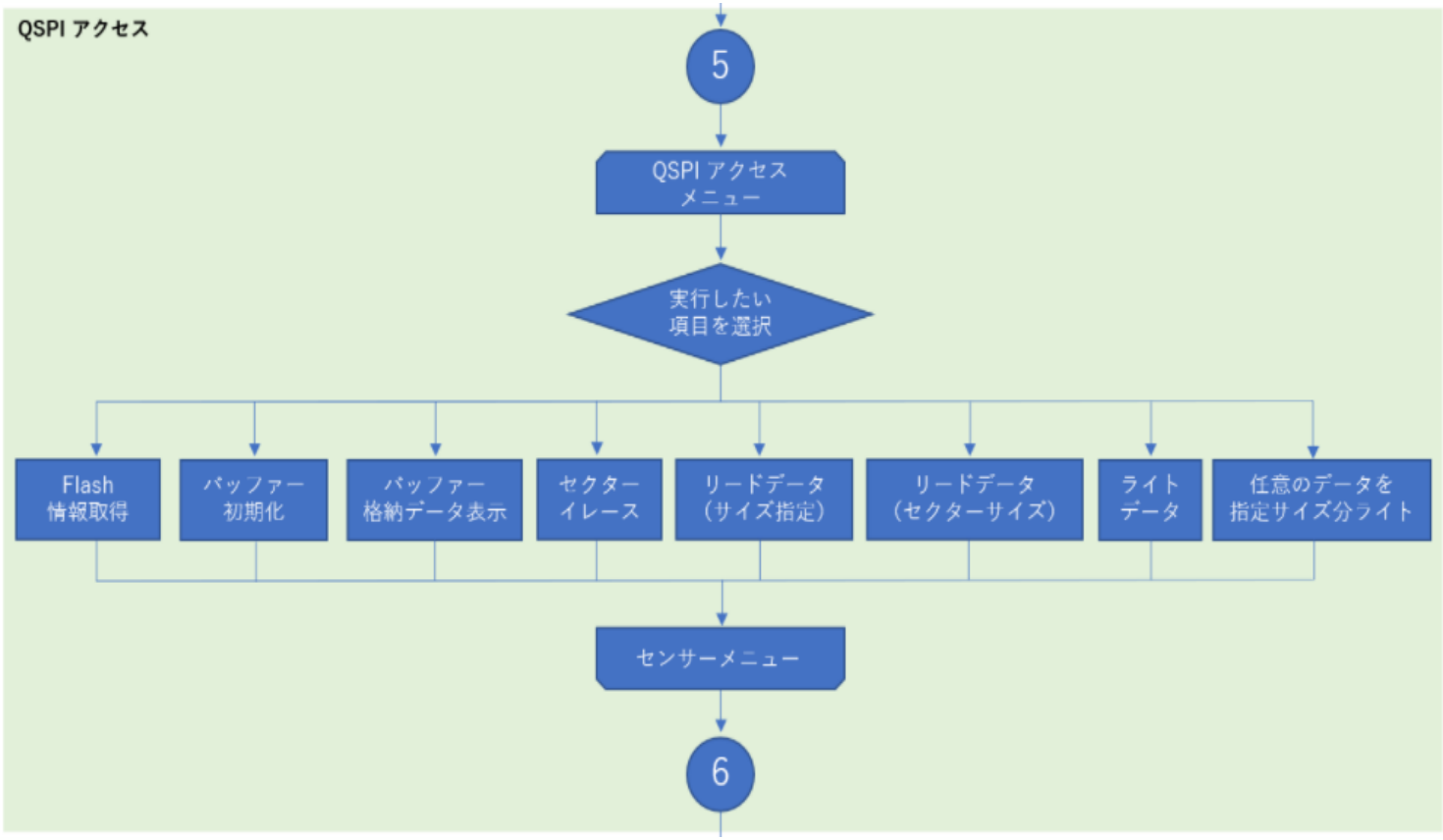
フローチャート

- **RSU**
  - RSU メニュー
    - 現在起動中デザインの格納アドレス確認
    - Flash 内のアドレスを指定し、Image Upgrade



フローチャート

- **QSPI アクセス**
  - QSPI アクセスメニュー
    - Flash 情報取得
    - バッファー初期化
    - バッファー格納データ表示
    - セクターイレース
    - リードデータ（サイズ指定）
    - リードデータ（セクターサイズ）
    - ライトデータ
    - 任意のデータを指定サイズ分ライト

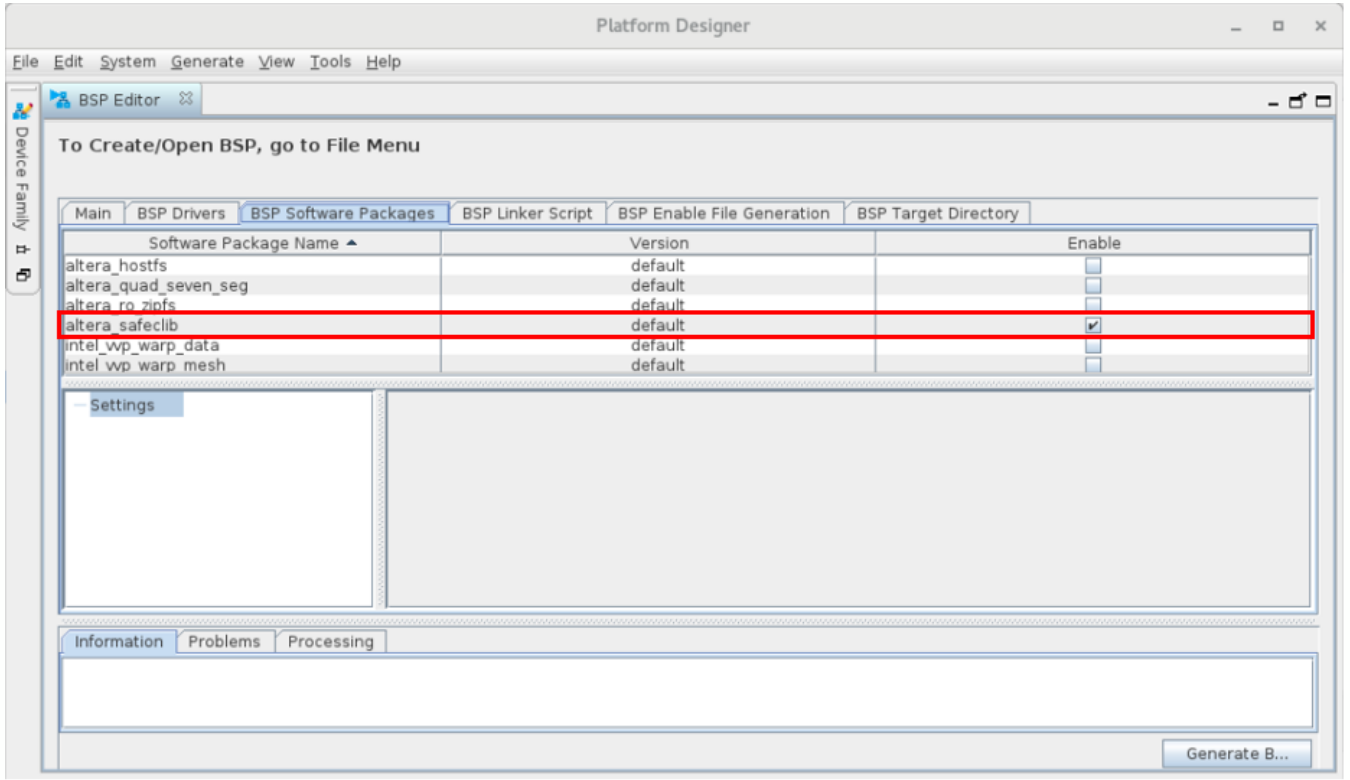


サンプル使用時注意点

MACNICA

サンプル使用時注意点 1

- **HAL API を使用するための設定**
  - サンプルでは HAL API が使用されています
  - HAL API を使用する際には事前に BSP-Editor で altera_safeclib を有効化
    - 設定箇所は BSP Editor の BSP software Packages タブの altera_safeclib



サンプル使用時注意点 2

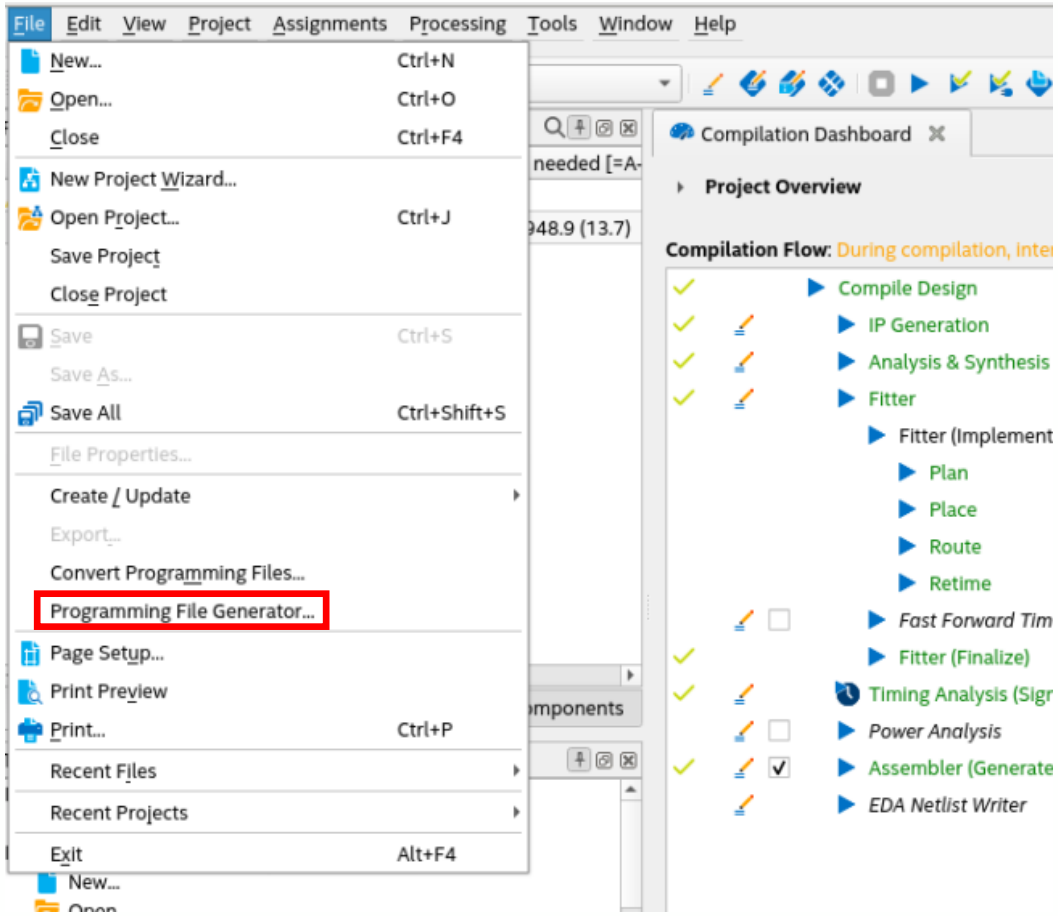
※(RSU を使用する場合)

- **.jic ファイル作成**
  - 本サンプルで RSU 機能を使用する場合は、
    予め Flash にイメージを書き込んでおく必要があります

  - .jic ファイル作成手順は次のページ以降で紹介
    - 手順通りに作成した .jic ファイルはプロジェクト直下に、
      RSU_image.jic というファイル名で予め格納されています
      ※インテル® Stratix® 10 SX SoC Development Kit 用

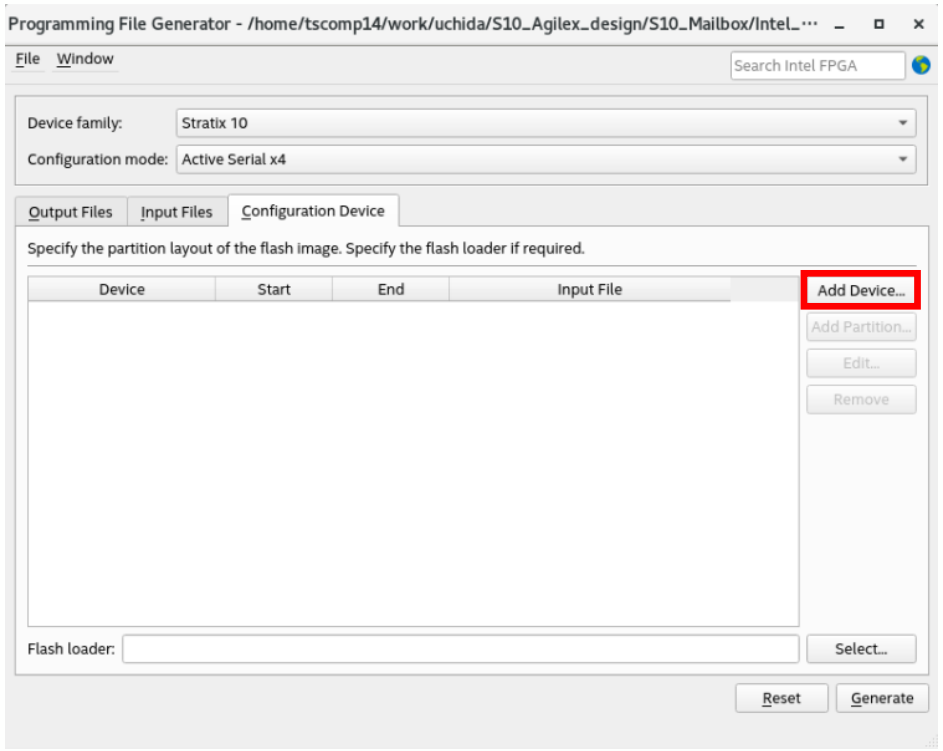
.jic ファイル作成手順

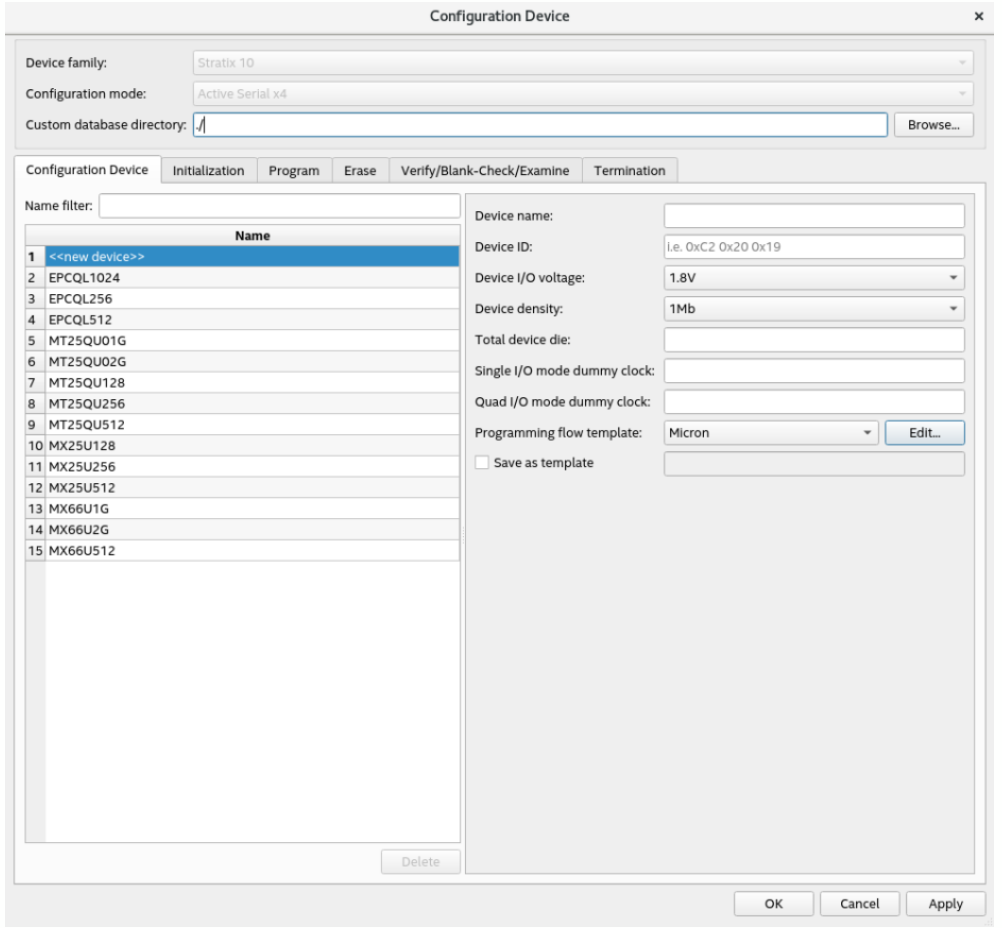
- 作成手順
  - 1. Quartus メニューバーから
    “programming File Generator” を選択



.jic ファイル作成手順

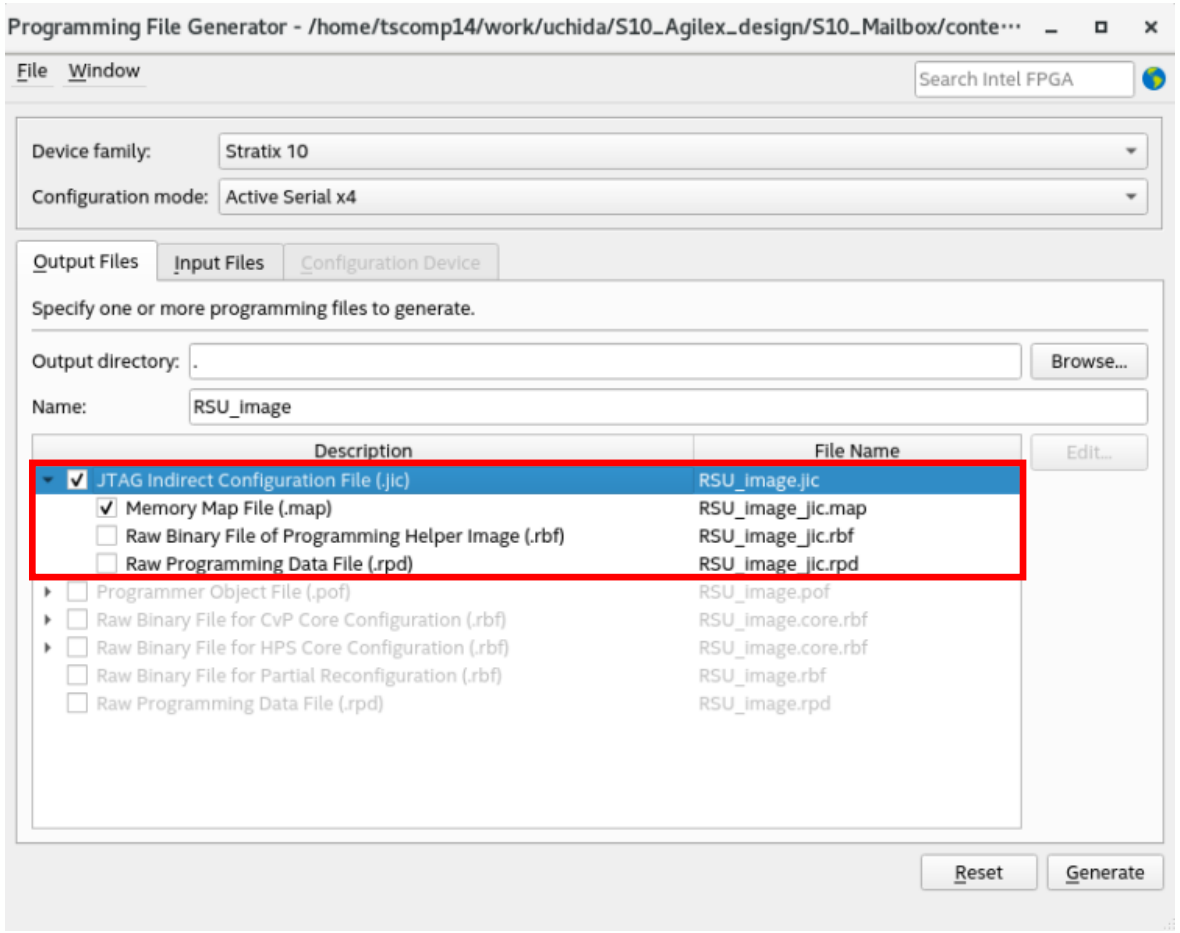
- 2. “Configuration Device” タブで、
  Add Device を選択し使用している QSPI を選択





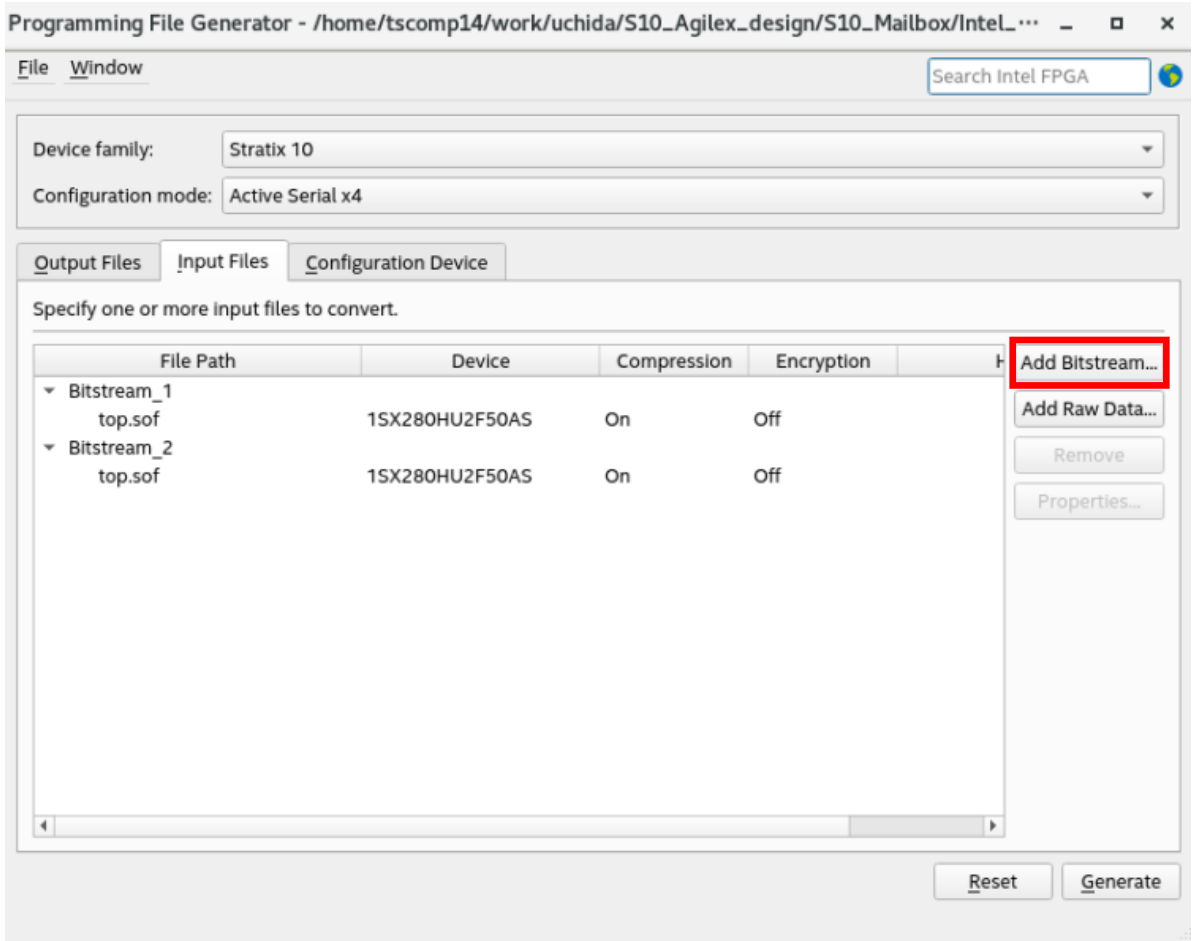
.jic ファイル作成手順

- 3. Output Files タブで下記 2 つのファイルを選択
  - “JTAG Indirect Configuration File（.jic）”
  - “Memory Map File（.map）”を選択



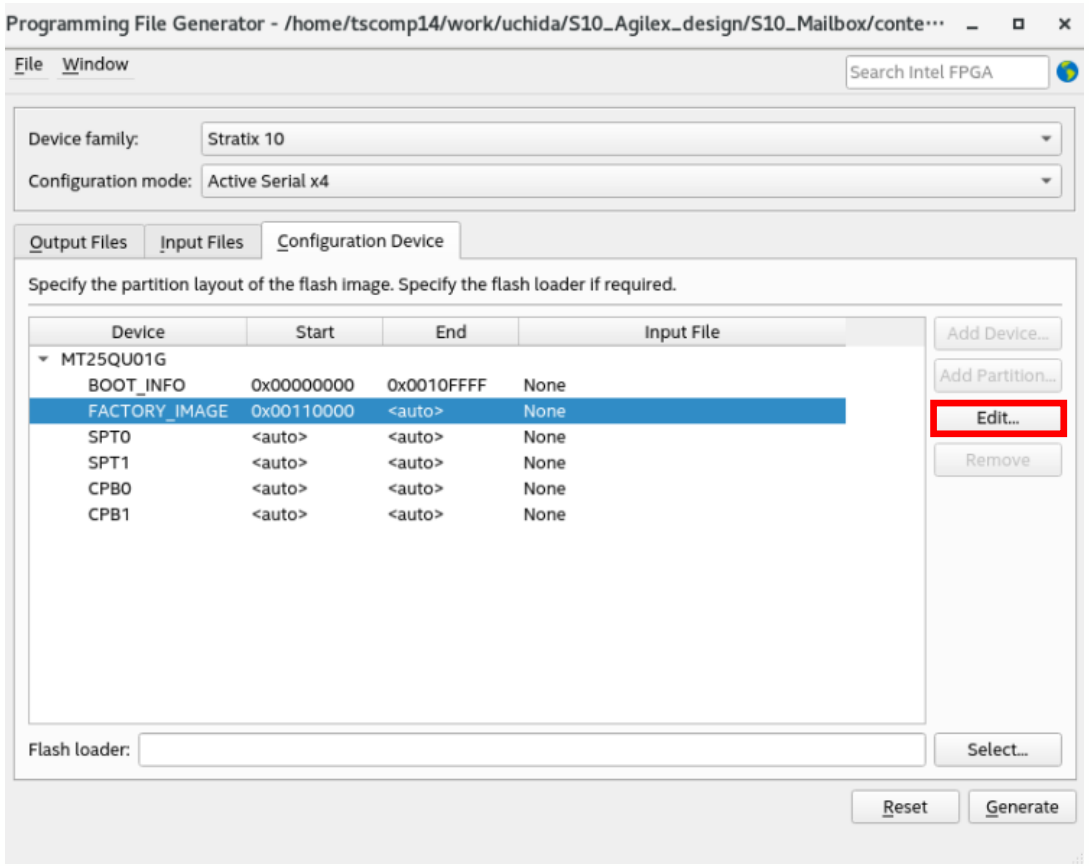
.jic ファイル作成手順

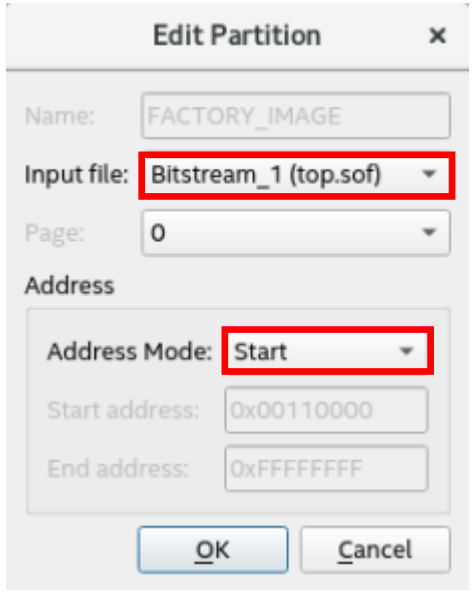
- 4. Input Files タブで .sof ファイルを 2 つ登録
  - Add Bitstream ボタンからファイルを選択
  - 今回は同一のファイルを 2 つ登録



.jic ファイル作成手順

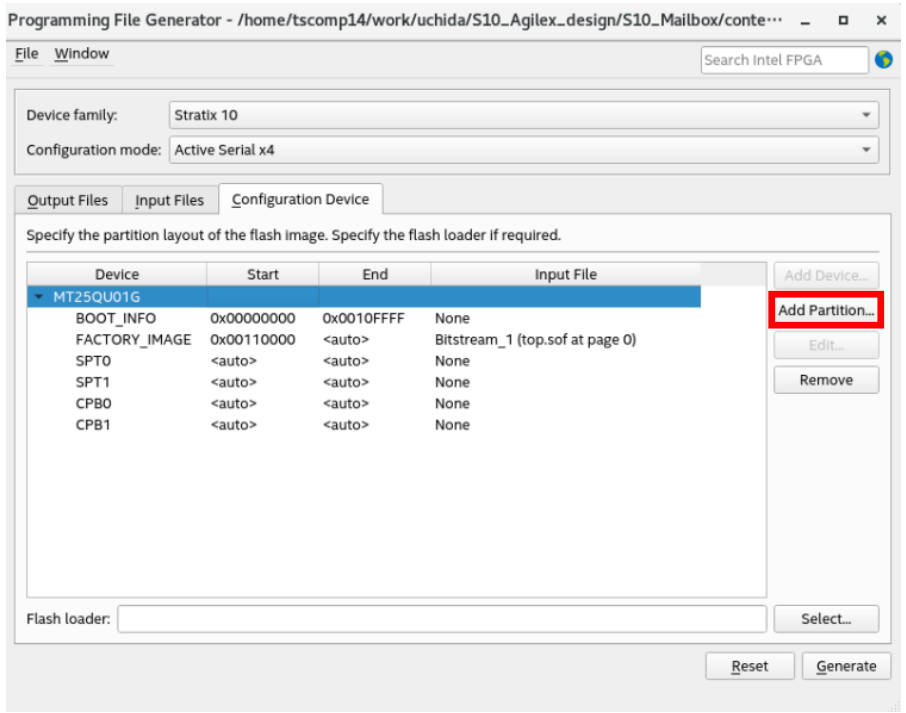
- 5. FACTORY_IMAGE を設定
  - FACTORY_IMAGE をハイライトし Edit ボタンを押す
  - Input file: を Bitstream_1 に設定
  - Address Mode: を Start に設定

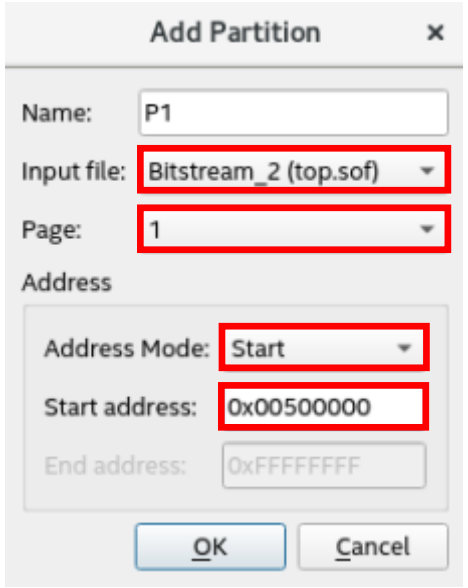




.jic ファイル作成手順

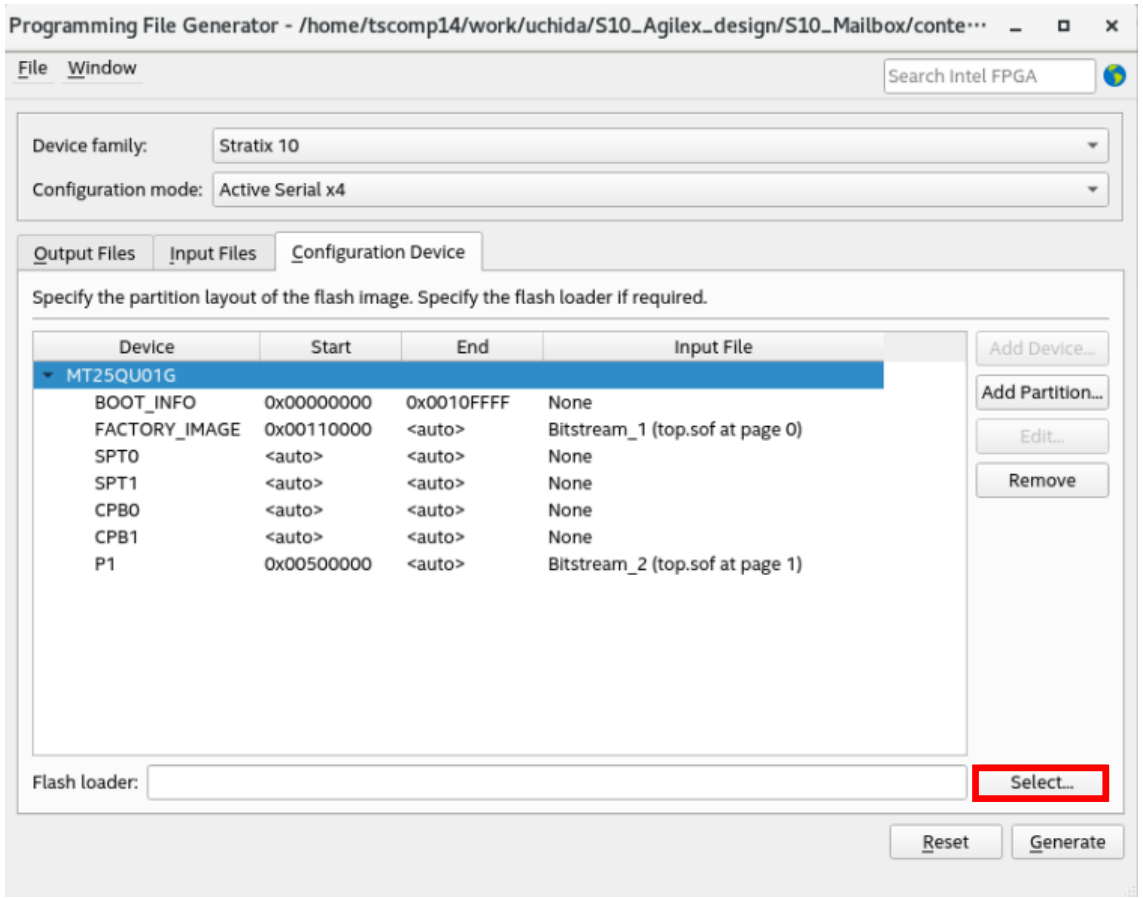
- 6. アプリケーションイメージの追加
  - Flash デバイスをハイライトし “add partition” ボタンをクリック
  - Input file: を Bitstream_2(top.sof) に設定
  - Page: を 1 に設定
  - Address Mode: を Start に設定
  - Start address: を 0x00500000 に設定

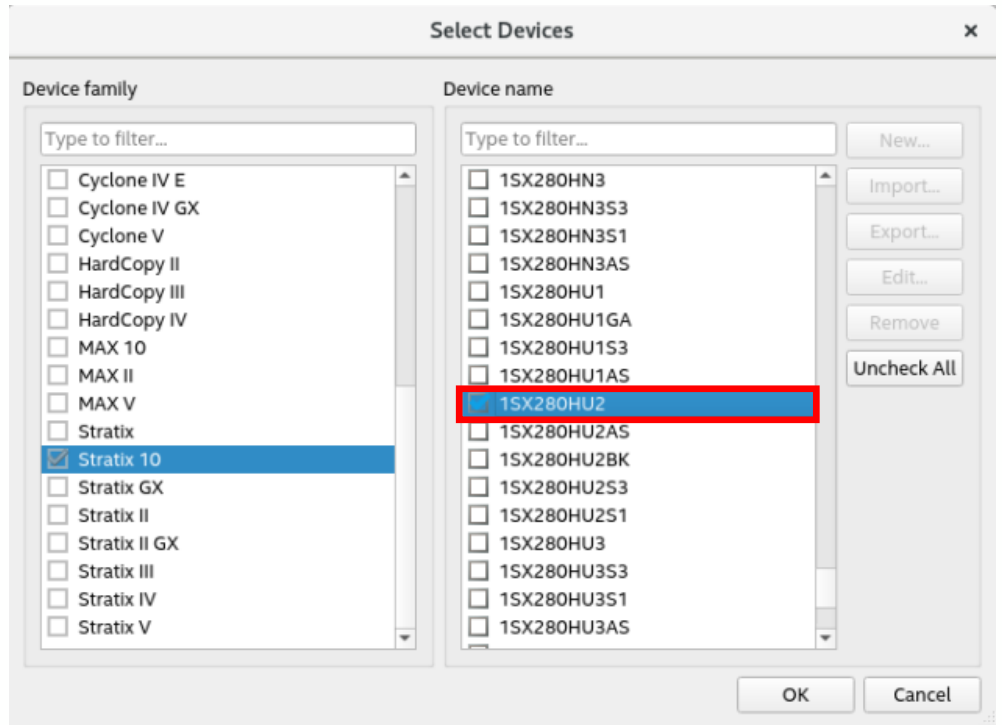




.jic ファイル作成手順

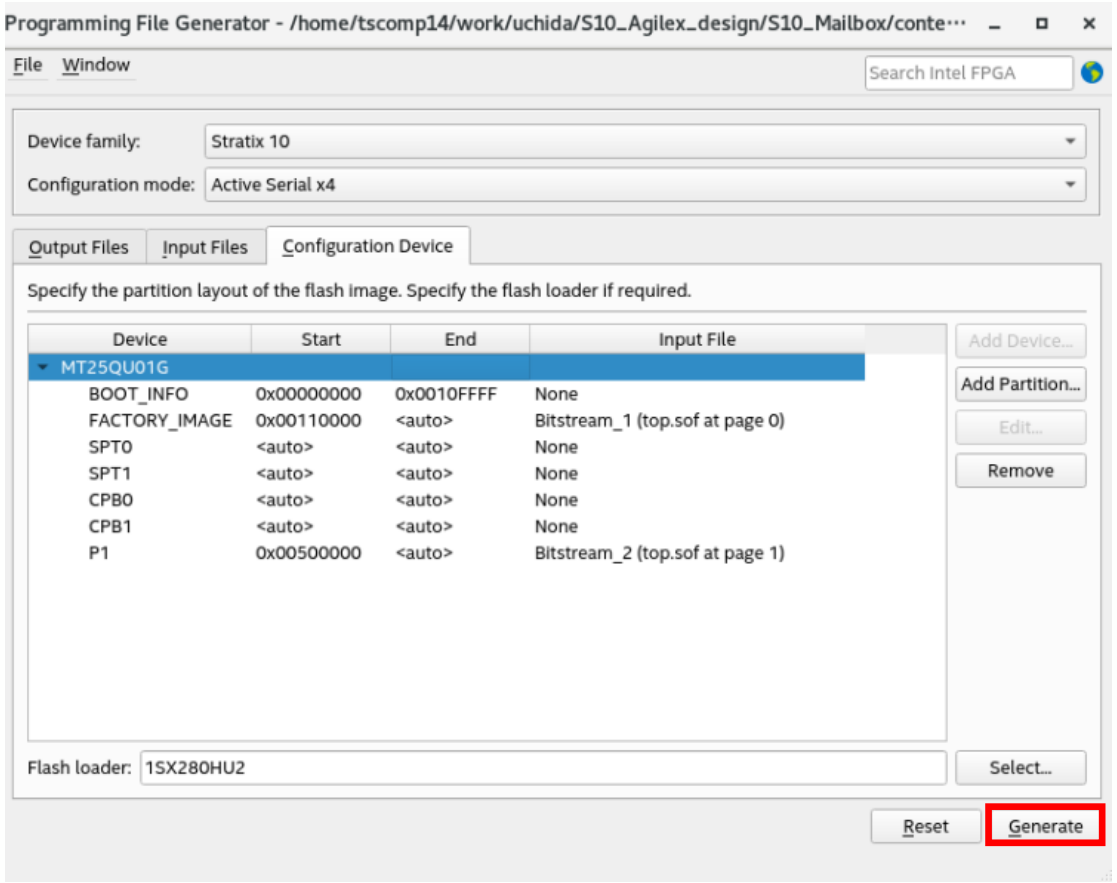
- 7. Flash loader を設定
  - “select” ボタンをクリックし、FPGA を選択
  - インテル® Stratix® 10 SX SoC Development Kit の場合は、1SX280HU2 を選択





.jic ファイル作成手順

- 7. .jic ファイル作成
  - “Generate” ボタンをクリックし、.jic ファイルを生成



サンプル動作説明

MACNICA

サンプル動作説明

- **サンプル起動時**
  - 全体メニューが表示されるので、
    0~3 の中から希望する処理を選択

```
Mailbox Client IP Sample Start!

result=0, Mailbox Client IP Open Success

=====Menu=====
0: Temperature/Voltage Sensor
1: Remote System Upgrade(RSU)
2: QSPI access
3: Sample Finish
==============
```

  - 処理毎に説明しているので、下記記載に沿ってページを移動
    - 0 を選択 -> P25 を参照
    - 1 を選択 -> P29 を参照
    - 2 を選択 -> P34 を参照

サンプル動作説明 ~センサー編~

MACNICA

サンプル動作説明 ~センサー編~

- **0: Temperature/Voltage Sensor を選択した場合**
  - 下記メニューが起動

```
=====Menu=====
0: Voltage Sensor
1: Temperature Sensor
2: Finish
==============
```

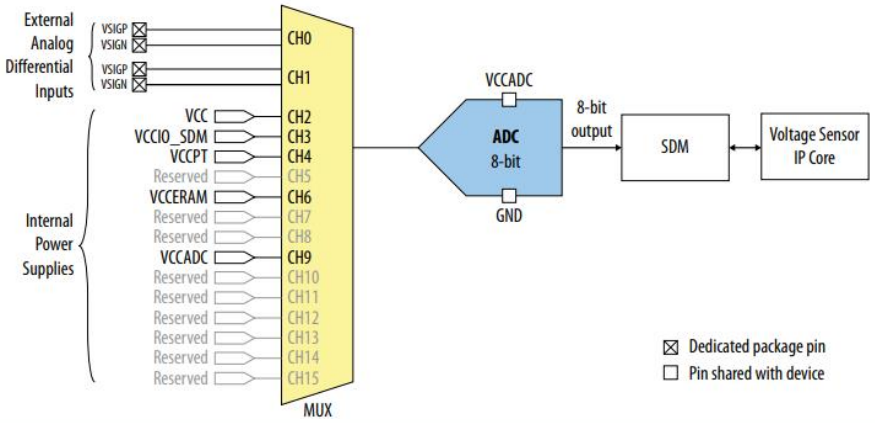
  - 値を取得したいセンサーを選択
    - 0 -> Voltage Sensor
    - 1 -> Temperature Sensor

サンプル動作説明 ~センサー編~

- **Voltage Sensor 動作**
  - Channel を選択し値を取得
    - 下記は Channel 0 を選択した場合

```
---Get Voltage value---
Channel:
0
result=0, Get Voltage Success
==========
Channel 0
Voltage Value is 0x00000013
==========
```

Figure 1. Intel Stratix 10 Voltage Sensor



※ Intel® Stratix® 10 Analog to Digital Converter User Guide から抜粋

  - Voltage Sensor については下記資料を参照
    - Intel® Stratix® 10 Analog to Digital Converter User Guide
    - Intel® Agilex™ Power Management User Guide

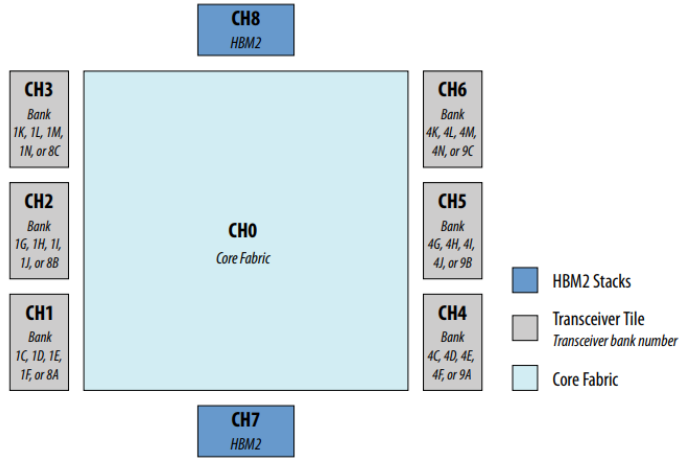
サンプル動作説明 ~センサー編~

- **Temperature Sensor 動作**
  - Channel を選択し値を取得
    - 下記は Channel 0 を選択した場合

```
---Get Temperature value---
dump_select Channel:
0
result=0, Get Temperature value Success
==========
Channel 0
Temperature value is 0x00001ae0
==========
```

Figure 5. Locations and Channel Numbers of Intel Stratix 10 TSDs

This diagram shows the temperature sensor channel locations from a package bottom view. Each transceiver tile in the diagram is labeled using the bank number of one of its transceiver banks.



※ Intel® Stratix® 10 Analog to Digital Converter User Guide から抜粋

  - Temperature Sensor については下記資料を参照
    - Intel® Stratix® 10 Analog to Digital Converter User Guide
    - Intel® Agilex™ Power Management User Guide

サンプル動作説明 ~RSU 編~

MACNICA

サンプル動作説明 ~RSU 編~

- **1: Remote System Upgrade(RSU) を選択した場合**
  - 下記メニューが起動

```
=====Menu=====
0: check current Image Address
1: Remote System Upgrade(RSU)
2: Finish
==============
```

  - 処理を選択
    - 0 -> check current Image Address
    - 1 -> Remote System Upgrade(RSU)

サンプル動作説明 ~RSU 編~

- **check current Image Address**
  - 実行すると現在起動しているイメージが格納されているアドレス表示

```
=====Current status=====
Current Image Address = 0x00110000
========================
```

→今回は QSPI の 0x00110000 から起動していることが分かります

サンプル動作説明 ~RSU 編~

- **Remote System Upgrade(RSU)**
  - 注意
    - RSU を実行すると Jtag が切断される
      実行後動作を継続する場合には再度デバッガを接続する必要があります
  - 起動したいイメージを選択

```
Note. Jtag Connection is disconnected by RSU.
Please reconnect the debugger again.

=====choose IMG=====
0: Factory IMG(0x00110000)
1: APP0(0x00500000)
2: Other Image
====================
```

※Other Image を選択すると、任意のアドレスを指定可能

サンプル動作説明 ~RSU 編~

- “1: APP0(0x00500000)” を選択した場合

```
Note. Jtag Connection is disconnected by RSU.
Please reconnect the debugger again.

=====choose IMG=====
0: Factory IMG(0x00110000)
1: APP0(0x00500000)
2: Other Image
====================

1
Upgrading the image...
If error doesn't appear, probably RSU is success.
```

RSU が実行され、エラーが表示されなければ成功しています

JTAG の接続が切断されているので、再度デバッガを接続する必要があります

- 現在のイメージのアドレスを表示し、切り替わっていることを確認
  - 再度デバッガを接続し、”Check current image address” を実施

```
=====Current status=====
Current Image Address = 0x00500000
========================
```

サンプル動作説明 ~QSPI アクセス編~

MACNICA

サンプル動作説明 ~QSPI アクセス編~

- **“2: QSPI access” を選択した場合**
  - 下記メニューが起動

```
result=0, QSPI Flash Open Success
result=0, getinfo Success


========================================
QSPI Access MENU
0: Get QSPI Flash Info
1: Buffer Initialize(Sector size(0x10000 Byte))
2: Buffer Dump(4/16/64/SectorSize(0x10000) Byte)
3: Erase Sector
4: Read Data
5: Read Sector
6: Write Data
7: Flash Fill Data
8: QSPI Flash Close
========================================
```

メニューから実施したい項目を選択

尚、メニュー表示時点で QSPI オープン処理が実施されています

その為、終了する際は “QSPI Flash Close” を実施する必要があります

サンプル動作説明 ~QSPI アクセス編~

- **Get QSPI Flash Info**
  - 下記情報を表示

```
~~~ 0: Get QSPI Flash Info ~~~
offset = 0
region_size = 0x8000000
number_of_blocks(number of sector) = 2048
block_size(sector size) = 0x10000 (decimal: 65536)
number_of_regions = 1
```

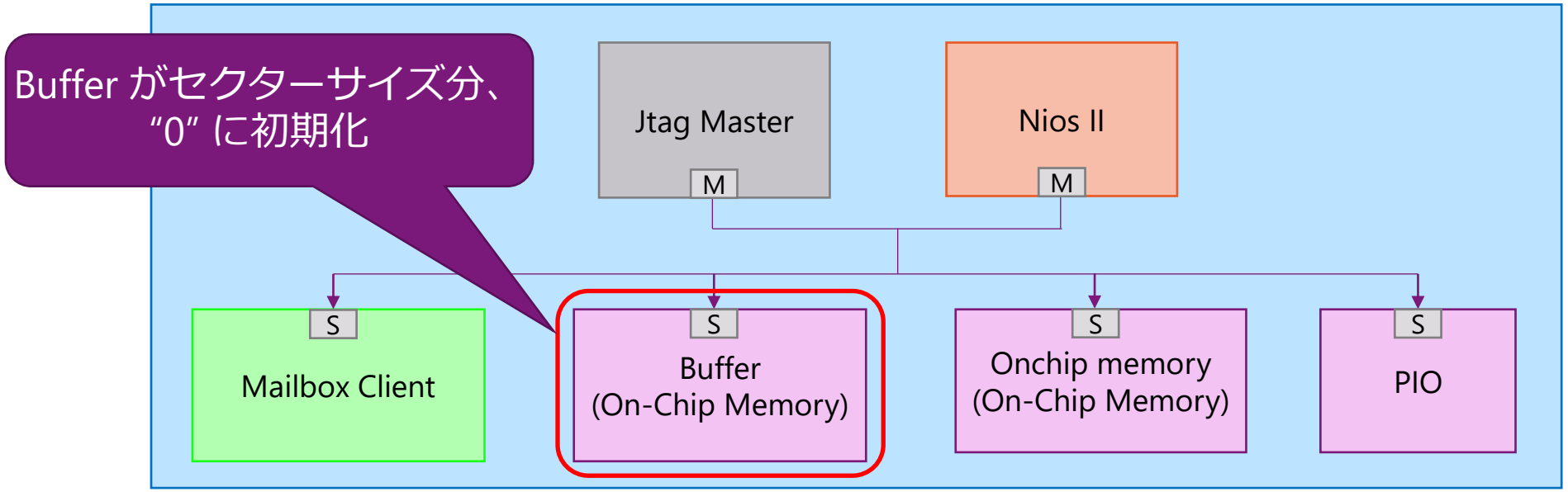
  - パラメータ
    - Offset: オフセット
    - Region_size: QSPI のサイズ
    - Number_of_blocks: セクター数
    - Block_size(sector size): セクターサイズ
    - Number_of_regions: QSPI の数

サンプル動作説明 ~QSPI アクセス編~

- **Buffer Initialize(Sector size)**
  - セクターサイズ分、バッファーが “0” に初期化されています

```
~~~ 1: Buffer Initialize(Sector size(0x10000)) ~~~
Buffer Initialization finish
```

※バッファーはリード/ライト時に一時的にデータを格納する役割です。



サンプル動作説明 ~QSPI アクセス編~

- **Buffer Dump(x/16/64/SectorSize Byte)**
  - サイズを選択し、バッファーをダンプ

```
~~~ 2: Buffer dump ~~~
=====DumpSize=====
0: 4Byte(1word)
1: 16Byte(4word)
2: 64Byte(16word)
3: 1KByte
4: sector Size(0x10000Byte)

Select dump size:
=================

2
Dump Size is 64 Byte
Addr 0x00000000:   0x00000000  0x00000000  0x00000000  0x00000000
Addr 0x00000010:   0x00000000  0x00000000  0x00000000  0x00000000
Addr 0x00000020:   0x00000000  0x00000000  0x00000000  0x00000000
Addr 0x00000030:   0x00000000  0x00000000  0x00000000  0x00000000
```

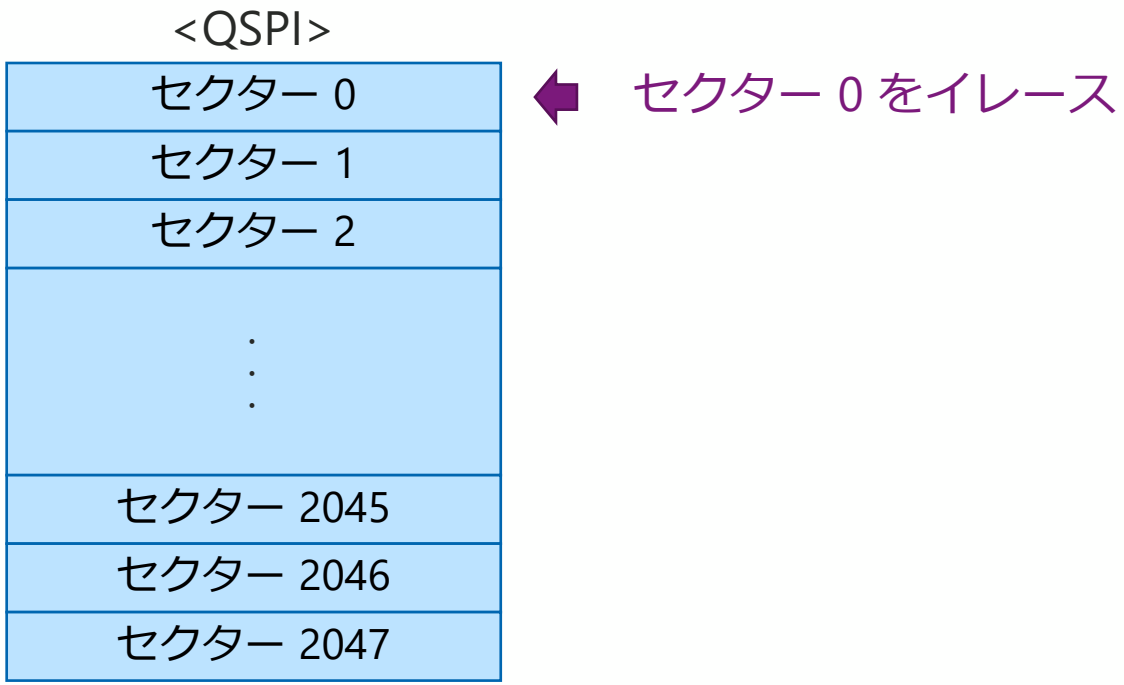
2: 64Byte(16word) を選択した場合

サンプル動作説明 ~QSPI アクセス編~

- **Erase Sector**
  - 選択したセクターをイレース

```
~~~ 3: Erase Sector ~~~
Please Input Sector number(0 ~ 2048):
0
result=0, Erase Success
```

セクター 0 を選択しイレースを実行



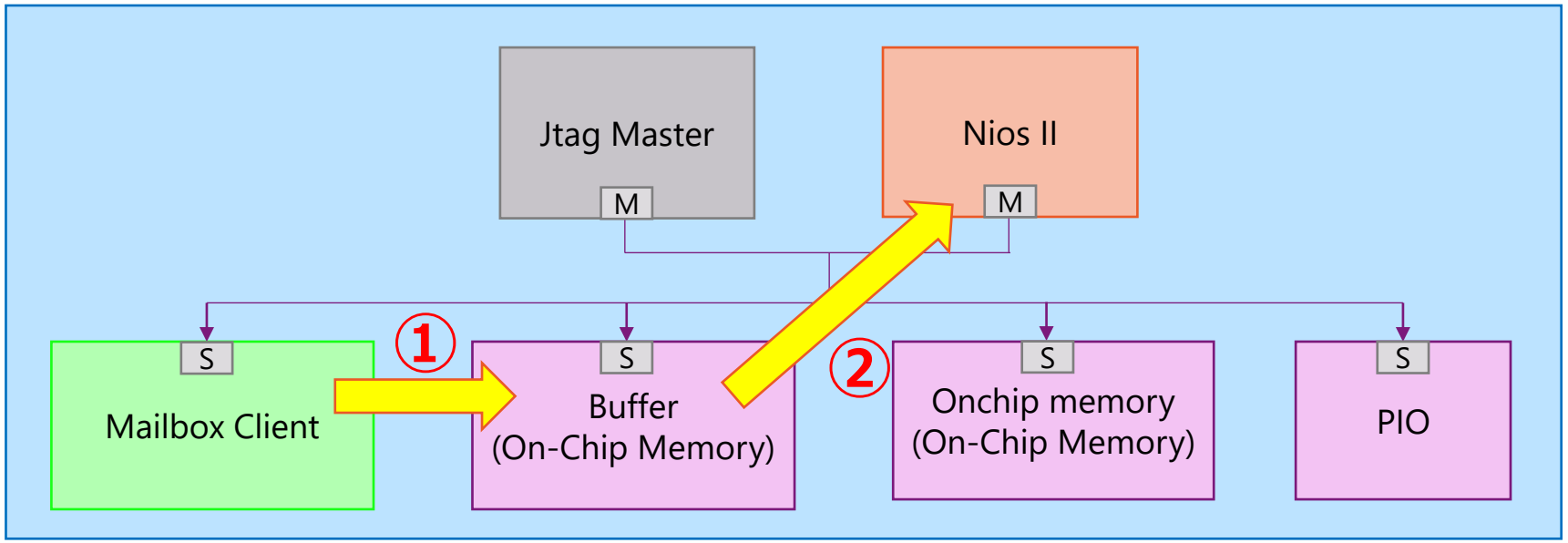
サンプル動作説明 ~QSPI アクセス編~

- **Read Data**
  - QSPI のアドレスを指定し 1 word 分をリード

```
~~~ 4: Read Data(1word(32bit)) ~~~
Read addr(hex):
0x0
result=0, Read Success
Read Data is 0xffffffff
```

0x0 をリード
（今回はセクター 0 をイレース後のため、
値が 0xffffffff となっている）

  - リード動作



①: リードデータを
バッファーに格納

②: Nios II から
バッファーを参照し、
リードデータを取得

サンプル動作説明 ~QSPI アクセス編~

- **Read Sector**
  - 指定したセクター全体をリード

```
~~~ 5: Read Sector ~~~
Please Input Sector number(0 ~ 2048):
0
result=0, Read Sector Success
Read Sector data is stored to buffer
```

セクター 0 をリードし、バッファーへ格納

バッファーをダンプし、
リードしたデータを確認

```
~~~ 2: Buffer dump ~~~
=====DumpSize=====
0: 4Byte(1word)
1: 16Byte(4word)
2: 64Byte(16word)
3: 1KByte
4: sector Size(0x10000Byte)

Select dump size:
=================

2
Dump Size is 64 Byte
Addr 0x00000000:   0xffffffff  0xffffffff  0xffffffff  0xffffffff
Addr 0x00000010:   0xffffffff  0xffffffff  0xffffffff  0xffffffff
Addr 0x00000020:   0xffffffff  0xffffffff  0xffffffff  0xffffffff
Addr 0x00000030:   0xffffffff  0xffffffff  0xffffffff  0xffffffff
```

Pxx で説明した Buffer dump を実行

64Byte(16 word) 分を表示

サンプル動作説明 ~QSPI アクセス編~

- **Write data**
  - QSPI のアドレスとライトするデータを入力し実行

```
~~~ 6: Write Data(1word(32bit)) ~~~
write addr(hex):
0x0
write data(hex):
0x01234567
result=0, Write Success
```

0x0 に 0x01234567 をライト

ライトしたデータを確認するために、
リードを実行

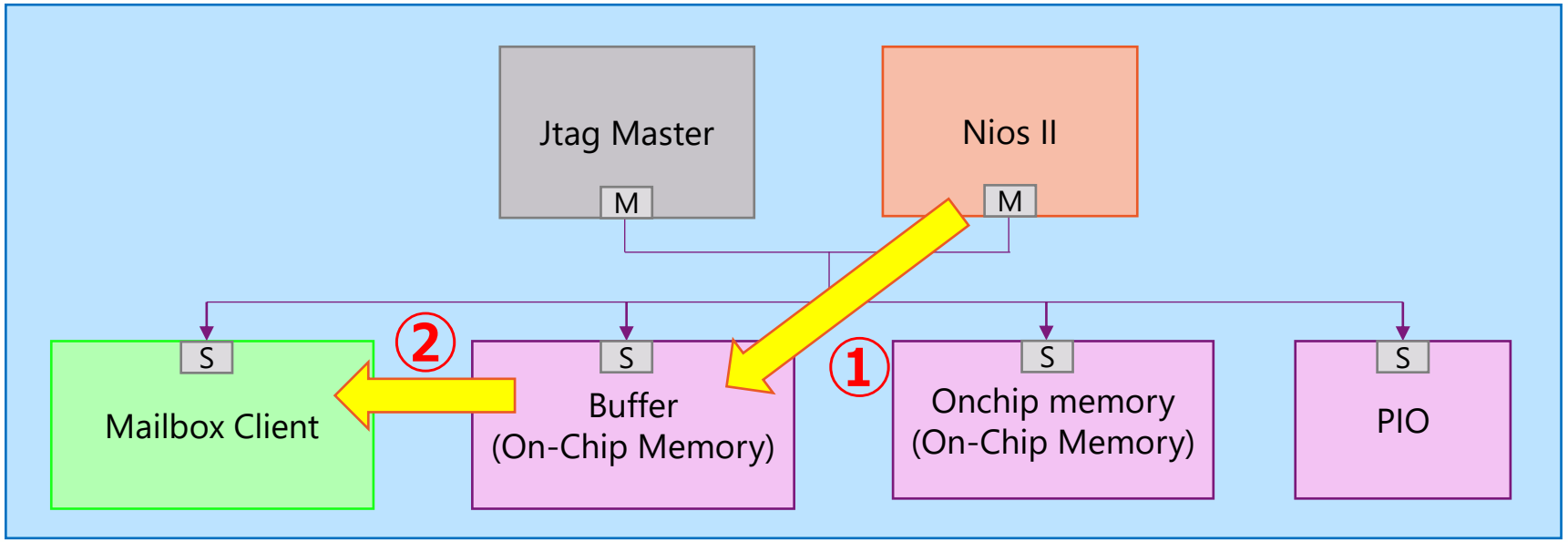
```
~~~ 4: Read Data(1word(32bit)) ~~~
Read addr(hex):
0x0
result=0, Read Success
Read Data is 0x01234567
```

Pxx で紹介した Read Data を使用し、
リードしたデータを確認

0x0 をリードすると 0x01234567 となっているので、
正常にライトが実行されていることが分かります

サンプル動作説明 ~QSPI アクセス編~

- ライト動作



①: バッファーに
データーを格納

②: Mailbox Client IP が
Buffer からライトデータ
を取得

サンプル動作説明 ~QSPI アクセス編~

- **Flash Fill Data**
  - 入力データを指定アドレスから指定サイズ分ライト

```
~~~ 7: Flash Fill data ~~~
Please Input Write Address(hex):
0x0
Please Input Write Byte data(hex):
0x5a
Please Input length (0~1024)(decimal):
10
result=0, Flash Fill data Success
```

0x0 から 0x5a を 10 byte 分ライト

```
~~~ 5: Read Sector ~~~
Please Input Sector number(0 ~ 2048):
0
result=0, Read Sector Success
Read Sector data is stored to buffer
```

Pxx で紹介した、セクターサイズ分のリードを実施

```
Dump Size is 64 Byte
Addr 0x00000000:   0x5a5a5a5a  0x5a5a5a5a  0xffff5a5a  0xffffffff
Addr 0x00000010:   0xffffffff  0xffffffff  0xffffffff  0xffffffff
Addr 0x00000020:   0xffffffff  0xffffffff  0xffffffff  0xffffffff
Addr 0x00000030:   0xffffffff  0xffffffff  0xffffffff  0xffffffff
```

Pxx で紹介した、バッファーのダンプで、正常にライトされていることを確認

サンプル動作説明 ~QSPI アクセス編~

- **QSPI Flash Close**
  - QSPI のクローズ処理を実施

```
~~~ 8: QSPI Flash Close ~~~
result=0, QSPI Flash Close Success
```

Co.Tomorrowing