AP-RA8P-0A
AI サンプルプログラム解説

1.0 版　2026 年 03 月 09 日

1. 概要

1.1 概要

本アプリケーションノートでは、AP-RA8P-0A（RA8P1 CPU）を用いて、Flexible Software Package を使用した AI サンプルプログラムについて解説します。
ソフトウェアは、静電容量式 LCD キット（LCD-KIT-B01、LCD-KIT-B02、LCD-KIT-D02、LCD-KIT-D03）および抵抗膜式 LCD キット（LCD-KIT-C01、LCD-KIT-C02）に対応しています
本サンプルプログラムで使用する主な機能を以下に記します。

デバイス	機能	動作内容
AP-RA8P-0A	・AI 処理	・顔検出
	・グラフィック LCD コントローラ（GLCDC）	・グラフィック表示
	・ビデオ入力モジュール	・カメラ映像入力処理
	・MIPI CSI	・MIPI カメラ（Rasberry Pi Camera Module V2）のデータ受信
	・UART 通信	・ログ出力
LCD-KIT ・LCD-KIT-B01 ・LCD-KIT-B02 ・LCD-KIT-C01 ・LCD-KIT-C02 ・LCD-KIT-D02 ・LCD-KIT-D03	・LCD パネル	・各画面の表示
	・バックライト	・バックライトの点灯
	・スイッチ	・各種機能の切り替え

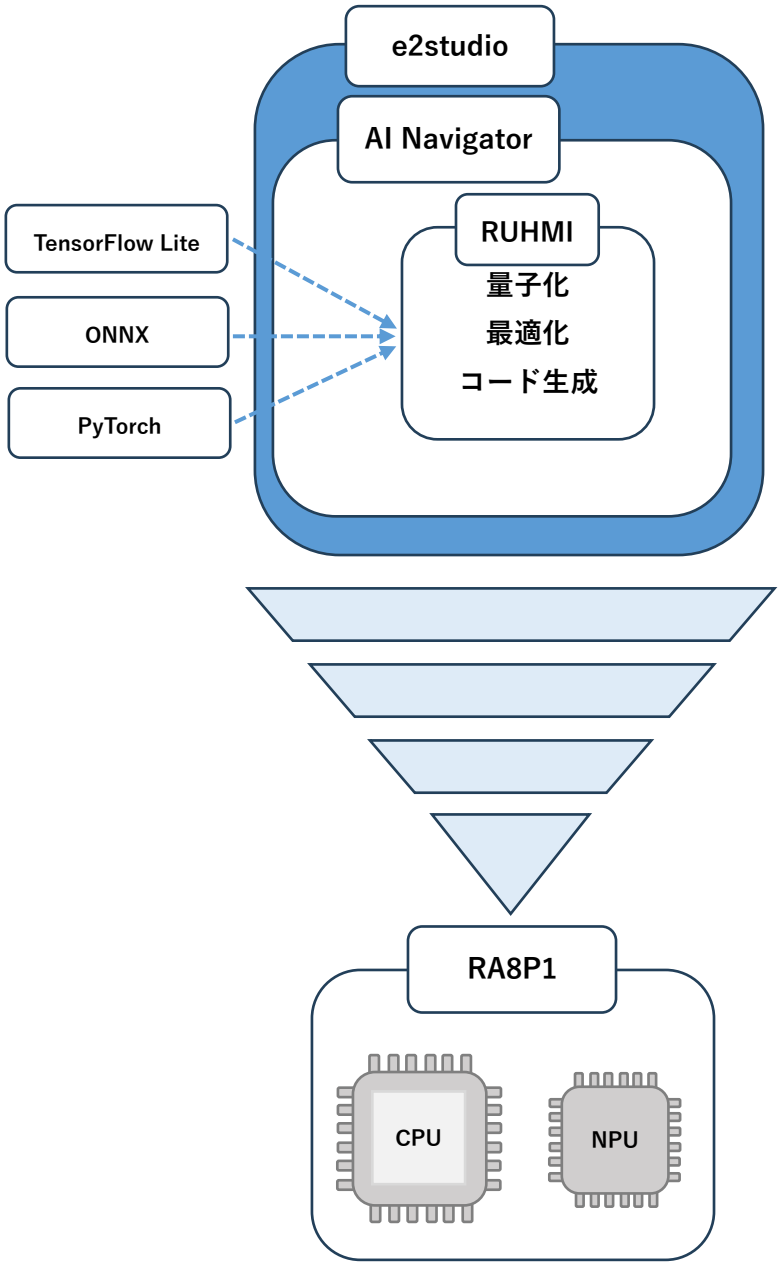
1.2　Robust Unified Heterogeneous Model Integration について

Robust Unified Heterogeneous Model Integration（以下、RUHMI と略す）はルネサス エレクトロニクス株式会社が無償で提供する、AI アプリケーションの開発を加速させるツールです。RUHMI によって AI モデルの効率的かつスケーラブルな展開を可能にします。RUHMI の一環として GUI ツール「AI Navigator」が e2studio のプラグイン形式で提供されています。

ルネサス エレクトロニクス株式会社

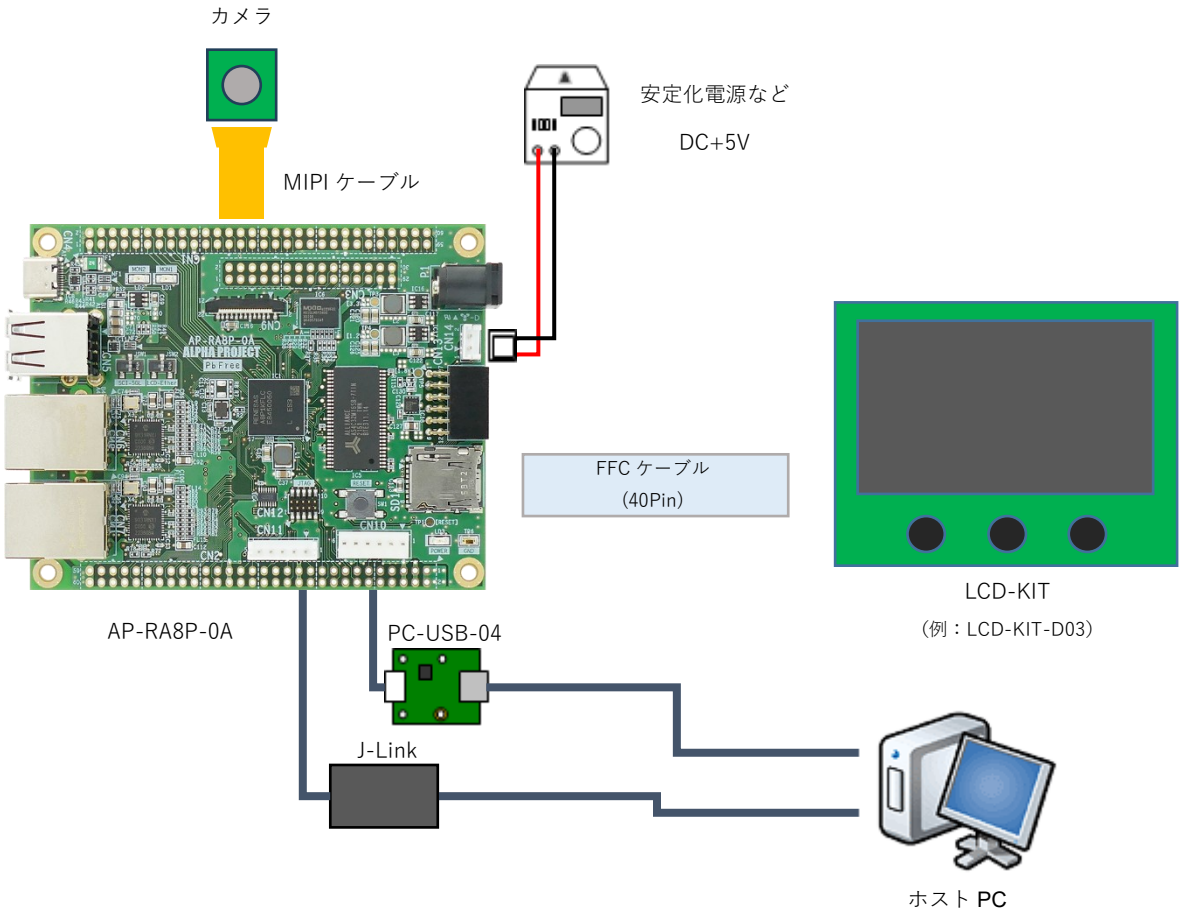
RUHMI 公開ページ RUHMI

RUHMI は TensorFlow Lite、ONNX、PyTorch/ExecuTorch などのモデル形式に対応しており、RUHMI 搭載の AI コンパイラを用いて量子化し、最適化された C/C++ソースコードを自動生成します。また CPU のみ及び CPU＋NPU といったヘテロジニアスなモデル展開が可能です。



1.3　接続概要

本サンプルプログラムの動作を確認する上で必要な CPU ボードの接続例を以下に示します。
詳細な接続に関しては後述の「3 動作説明」を参照してください。



1.4　本サンプルプログラムについて

本サンプルプログラムおよび本書含むアプリケーションノートは、弊社 Web サイトのボード紹介ページで公開されています。

株式会社アルファプロジェクト
　AP-RA8P-0A 製品ページ　https://www.apnet.co.jp/product/ra/ap-ra8p-0a.html

1.5　開発環境について

本サンプルプログラムは統合開発環境「e2 studio」と「Flexible Software Package（以下、FSP）」を用いて開発されています。
本サンプルプログラムに対応する開発環境、FSP、コンパイラ、デバッガのバージョンは次の通りです。

ソフトウェア	バージョン	備考
e2 studio	V2025-12	–
FSP	6.3.0	–
LLVM Embedded Toolchain for Arm	21.1.1	–
AI Navigator	2.1.0.v20251002-0524	–

デバッガ	ハードウェアバージョン	備考
J-Link	V11	Segger Microcontroller Systems 社

※AP-RA8P-0A と J-Link を直接接続することはできません。
　AP-RA8P-0A 側（ハーフピッチコネクタ）と J-Link 側（フルピッチコネクタ）を接続するための変換アダプタが必要となります。
　変換アダプタについては、J-Link 取扱店へご確認ください。

1.6　ワークスペースについて

本サンプルプログラムのプロジェクトファイルは次のフォルダに格納されています。
以降のフォルダ表記は LCD-KIT-B01/B02/C01/C02 用の「ap_ra8p_0a_face_detection_wvga_sample」とします。
LCD-KIT-D02/D03 をご使用の際はフォルダ名を「ap_ra8p_0a_face_detection_wqvga_sample」としてご使用ください。

サンプルプログラム	フォルダ
LCD-KIT-B01/B02/C01/C02 用 AI サンプルプログラムプロジェクトフォルダ	¥sample¥ap_ra8p_0a_face_detection_wvga_sample
LCD-KIT-D02/D03 用 AI サンプルプログラムプロジェクトフォルダ	¥sample¥ap_ra8p_0a_face_detection_wqvga_sample

2.　サンプルプログラムの構成

2.1　フォルダ構成

サンプルプログラムは下記のようなフォルダ構成になっています。

2.1.1　LCD-KIT-B01/B02/C01/C02 用 AI サンプルプログラムのフォルダ構成

sample	
└ ap_ra8p_0a_face_detection_wvga_sample	AI サンプルプログラムフォルダ
├ Debug	Debug ビルド用ワークフォルダ
└ src	ソースフォルダ
├ ai_application	AI アプリケーション処理格納フォルダ
│ ├ common	AI アプリケーション共通処理格納フォルダ
│ └ face_detection	顔検出用処理格納フォルダ
│ └ mera	AI モデル格納フォルダ
├ camera_layer	カメラ制御処理格納フォルダ
│ └ camera_imx219	imx219 ドライバ格納ファイル
├ console_output	コンソール出力処理格納フォルダ
│ └ SEGGER_RTT	デバッグ出力処理格納フォルダ
├ display_layer	画面制御処理格納フォルダ
│ └ lcdkit	LCD-KIT ドライバ格納フォルダ
├ external_memory	外部メモリ制御処理格納フォルダ
├ fsp_custom	カスタムドライバ格納フォルダ
│ └ r_mipi_phy	mipi_phy 制御処理格納フォルダ
└ time_counter	処理時間計測用タイマドライバ格納フォルダ

2.1.2　LCD-KIT-D02/D03 用 AI サンプルプログラムのフォルダ構成

sample	
└ ap_ra8p_0a_face_detection_wqvga_sample	AI サンプルプログラムフォルダ
├ Debug	Debug ビルド用ワークフォルダ
└ src	ソースフォルダ
├ ai_application	AI アプリケーション処理格納フォルダ
│ ├ common	AI アプリケーション共通処理格納フォルダ
│ └ face_detection	顔検出用処理格納フォルダ
│ └ mera	AI モデル格納フォルダ
├ camera_layer	カメラ制御処理格納フォルダ
│ └ camera_imx219	imx219 ドライバ格納ファイル
├ console_output	コンソール出力処理格納フォルダ
│ └ SEGGER_RTT	デバッグ出力処理格納フォルダ
├ display_layer	画面制御処理格納フォルダ
│ └ lcdkit	LCD-KIT ドライバ格納フォルダ
├ external_memory	外部メモリ制御処理格納フォルダ
├ fsp_custom	カスタムドライバ格納フォルダ
│ └ r_mipi_phy	mipi_phy 制御処理格納フォルダ
└ time_counter	処理時間計測用タイマドライバ格納フォルダ

2.2　ファイルの構成

サンプルプログラムは以下のファイルで構成されています。
本節では、サンプルプログラムの作成にあたって追加したファイルについてのみ記述し、自動生成ファイルなどに関しては説明を省略します。

＜¥sample¥CustomBSP フォルダ内＞

AlphaProject.ap_ra8p_0a.6.3.0.pack	・・・	AP-RA8P-0A パックファイル

2.2.1　LCD-KIT-B01/B02/C01/C02 用 AI サンプルプログラムのファイル構成

＜¥sample¥ap_ra8p_0a_face_detection_wvga_sample フォルダ内＞

.cproject	・・・	CPROJECT ファイル
.project	・・・	PROJECT ファイル
ap_ra8p_0a_face_detection_wvga_sample Debug.launch	・・・	AP-RA8P-0A AI サンプルプログラム デバッグおよびランタイム設定ファイル（Debug ビルド用）
configuration.xml	・・・	FSP コンフィギュレータファイル

＜¥sample¥ap_ra8p_0a_face_detection_wvga_sample¥src フォルダ内＞

time_counter	・・・	処理時間計測用タイマドライバ格納フォルダ
external_memory	・・・	外部メモリ制御処理格納フォルダ
console_output	・・・	コンソール出力処理格納フォルダ
ai_application	・・・	AI アプリケーション処理格納フォルダ
common_app.h	・・・	共通ヘッダファイル
common_utils.h	・・・	共通ユーティリティ処理ヘッダファイル
common_util.c	・・・	AI 処理用共通ユーティリティ処理ソースファイル
common_util.h	・・・	AI 処理用共通ユーティリティ処理ヘッダファイル
hal_entry.c	・・・	hal_entry 関数ソースファイル
camera_display_thread_entry.c	・・・	camera_display_thread_entry 関数ソースファイル
application_config.h	・・・	アプリケーションコンフィグ用ヘッダファイル
ai_inference_thread_entry.c	・・・	ai_inference_thread_entry 関数ソースファイル

＜¥sample¥ap_ra8p_0a_face_detection_wvga_sample¥src¥fsp_custom フォルダ内＞

r_mipi_phy	・・・	mipi_phy 制御処理格納フォルダ
mipicsi_vin_hal_driver_config.h	・・・	mipicsi_vin_hal_driver コンフィグ用ヘッダファイル
mipicsi_vin_hal_driver.h	・・・	mipicsi_vin_hal_driver 関数ヘッダファイル
mipicsi_vin_hal_driver.c	・・・	mipicsi_vin_hal_driver 関数ソースファイル

＜¥sample¥ap_ra8p_0a_face_detection_wvga_sample¥src¥display_layer フォルダ内＞

face_detection_screen_mipi.c	・・・	face_detection_screen_mipi 関数ソースファイル
display_layer_config.h	・・・	display_layer コンフィグ用ヘッダファイル
display_layer.h	・・・	display_layer 関数ヘッダファイル
display_layer.c	・・・	display_layer 関数ソースファイル

＜¥sample¥ap_ra8p_0a_face_detection_wvga_sample¥src¥display_layer¥lcdkit フォルダ内＞

lcdkit.h	・・・	LCD-KIT ヘッダファイル
lcdkit_app.c	・・・	LCD-KIT 制御処理ソースファイル
lcdkit_app.h	・・・	LCD-KIT 制御処理ヘッダファイル
lcdkit_b01.c	・・・	LCD-KIT-B01/B02 ドライバソースファイル
lcdkit_b01.h	・・・	LCD-KIT-B01/B02 ドライバヘッダファイル
lcdkit_c01.c	・・・	LCD-KIT-C01/C02 ドライバソースファイル
lcdkit_c01.h	・・・	LCD-KIT-C01/C02 ドライバヘッダファイル
lcdkit_d02.c	・・・	LCD-KIT-D02 ドライバソースファイル
lcdkit_d02.h	・・・	LCD-KIT-D02 ドライバヘッダファイル
lcdkit_d03.c	・・・	LCD-KIT-D03 ドライバソースファイル
lcdkit_d03.h	・・・	LCD-KIT-D03 ドライバヘッダファイル

＜¥sample¥ap_ra8p_0a_face_detection_wvga_sample¥src¥camera_layer フォルダ内＞

camera_layer_config.h	・・・	camera_layer コンフィグ用ヘッダファイル
camera_layer.h	・・・	camera_layer 関数ヘッダファイル
camera_layer.c	・・・	camera_layer 関数ソースファイル

＜¥sample¥ap_ra8p_0a_face_detection_wvga_sample¥src¥camera_layer¥camera_imx219 フォルダ内＞

camera_imx219_regdata.h	・・・	camera_imx219 レジスタ設定用ヘッダファイル
camera_imx219_regdata.c	・・・	camera_imx219 レジスタ設定用ソースファイル
camera_imx219.h	・・・	camera_imx219 関数ヘッダファイル
camera_imx219.c	・・・	camera_imx219 関数ソースファイル

2.2.2　LCD-KIT-D02/D03 用 AI サンプルプログラムのファイル構成

＜¥sample¥ap_ra8p_0a_face_detection_wqvga_sample フォルダ内＞

.cproject	・・・	CPROJECT ファイル
.project	・・・	PROJECT ファイル
ap_ra8p_0a_face_detection_wqvga_sample Debug.launch	・・・	AP-RA8P-0A AI サンプルプログラム デバッグおよびランタイム設定ファイル（Debug ビルド用）
configuration.xml	・・・	FSP コンフィギュレータファイル

＜¥sample¥ap_ra8p_0a_face_detection_wqvga_sample¥src フォルダ内＞

time_counter	・・・	処理時間計測用タイマドライバ格納フォルダ
external_memory	・・・	外部メモリ制御処理格納フォルダ
console_output	・・・	コンソール出力処理格納フォルダ
ai_application	・・・	AI アプリケーション処理格納フォルダ
common_app.h	・・・	共通ヘッダファイル
common_utils.h	・・・	共通ユーティリティ処理ヘッダファイル
common_util.c	・・・	AI 処理用共通ユーティリティ処理ソースファイル
common_util.h	・・・	AI 処理用共通ユーティリティ処理ヘッダファイル
hal_entry.c	・・・	hal_entry 関数ソースファイル
camera_display_thread_entry.c	・・・	camera_display_thread_entry 関数ソースファイル
application_config.h	・・・	アプリケーションコンフィグ用ヘッダファイル
ai_inference_thread_entry.c	・・・	ai_inference_thread_entry 関数ソースファイル

＜¥sample¥ap_ra8p_0a_face_detection_wqvga_sample¥src¥fsp_custom フォルダ内＞

r_mipi_phy	・・・	mipi_phy 制御処理格納フォルダ
mipicsi_vin_hal_driver_config.h	・・・	mipicsi_vin_hal_driver コンフィグ用ヘッダファイル
mipicsi_vin_hal_driver.h	・・・	mipicsi_vin_hal_driver 関数ヘッダファイル
mipicsi_vin_hal_driver.c	・・・	mipicsi_vin_hal_driver 関数ソースファイル

＜¥sample¥ap_ra8p_0a_face_detection_wqvga_sample¥src¥display_layer フォルダ内＞

face_detection_screen_mipi.c	・・・	face_detection_screen_mipi 関数ソースファイル
display_layer_config.h	・・・	display_layer コンフィグ用ヘッダファイル
display_layer.h	・・・	display_layer 関数ヘッダファイル
display_layer.c	・・・	display_layer 関数ソースファイル

＜¥sample¥ap_ra8p_0a_face_detection_wqvga_sample¥src¥display_layer¥lcdkit フォルダ内＞

lcdkit.h	・・・	LCD-KIT ヘッダファイル
lcdkit_app.c	・・・	LCD-KIT 制御処理ソースファイル
lcdkit_app.h	・・・	LCD-KIT 制御処理ヘッダファイル
lcdkit_b01.c	・・・	LCD-KIT-B01/B02 ドライバソースファイル
lcdkit_b01.h	・・・	LCD-KIT-B01/B02 ドライバヘッダファイル
lcdkit_c01.c	・・・	LCD-KIT-C01/C02 ドライバソースファイル
lcdkit_c01.h	・・・	LCD-KIT-C01/C02 ドライバヘッダファイル
lcdkit_d02.c	・・・	LCD-KIT-D02 ドライバソースファイル
lcdkit_d02.h	・・・	LCD-KIT-D02 ドライバヘッダファイル
lcdkit_d03.c	・・・	LCD-KIT-D03 ドライバソースファイル
lcdkit_d03.h	・・・	LCD-KIT-D03 ドライバヘッダファイル

＜¥sample¥ap_ra8p_0a_face_detection_wqvga_sample¥src¥camera_layer フォルダ内＞

camera_layer_config.h	・・・	camera_layer コンフィグ用ヘッダファイル
camera_layer.h	・・・	camera_layer 関数ヘッダファイル
camera_layer.c	・・・	camera_layer 関数ソースファイル

＜¥sample¥ap_ra8p_0a_face_detection_wqvga_sample¥src¥camera_layer¥camera_imx219 フォルダ内＞

camera_imx219_regdata.h	・・・	camera_imx219 レジスタ設定用ヘッダファイル
camera_imx219_regdata.c	・・・	camera_imx219 レジスタ設定用ソースファイル
camera_imx219.h	・・・	camera_imx219 関数ヘッダファイル
camera_imx219.c	・・・	camera_imx219 関数ソースファイル

3.　動作説明

3.1　サンプルプログラムの動作

本サンプルプログラムでは LCD-KIT「LCD-KIT-C02」、MIPI カメラ「Rasberry Pi Camera Module V2」及び PC との通信には「PC-USB-04」の使用を仮定しています。また使用する AI モデルに関しては Yolo-Fastest をもとに、エッジデバイス向けに最適化された「YOLO_fastest_192」を RHUMI にて変換し使用しています。

3.1.1　動作の流れ

サンプルプログラムの動作の流れを解説します。

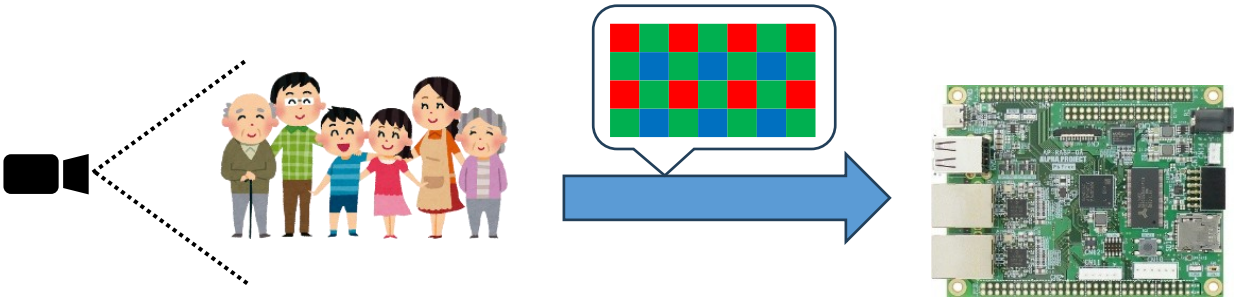
①　LCD-KIT を CPU ボードの CN8、MIPI カメラを CN9、PC-USB-04 は CN10 を経由してホスト PC に接続します。

②　ホスト PC でターミナルソフト(※本サンプルプログラムでは Teraterm)を起動し以下の設定にします。
COM ポートに関しては PC の環境によって異なるため、ご自身の COM ポートを確認し設定してください
COM ポートの設定は以下となっています。

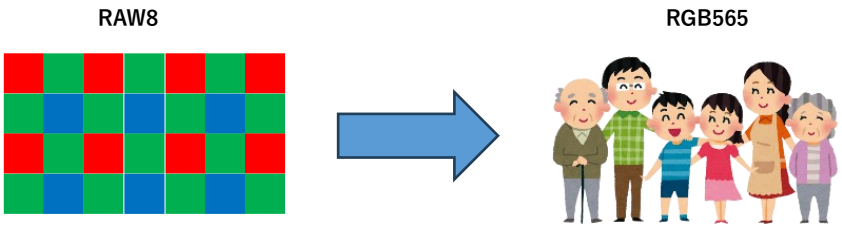
ボーレート	119200[bps]
ビット長	8 [bit]
パリティビット	なし
ストップビット	1[bit]
フロー制御	なし

③　CPU ボードに電源を投入し、サンプルプログラムを動作させます。

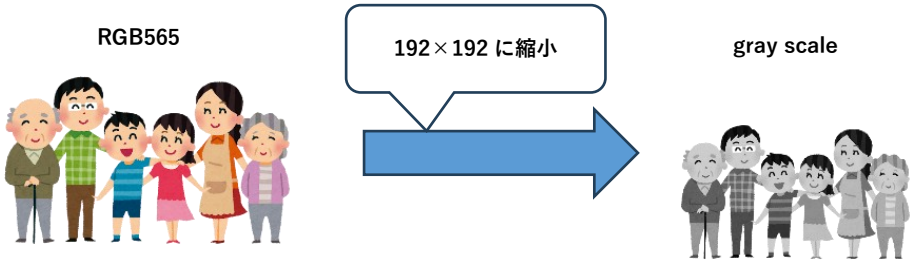
④　CPU ボードの起動後、カメラ映像の取得を開始します。
カメラ映像の取得後、MIPI 経由で RAW8 形式のデータを入力します。



⑤　RAW8 形式データ入力後、そのデータを RGB565 へ変換します。

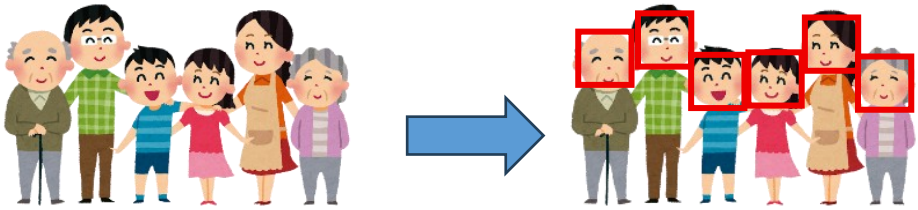


⑥　変換後の画像をもとに、AI モデルに入力するために gray scale 画像を作成し、画像のサイズを 192×192 に縮小します。

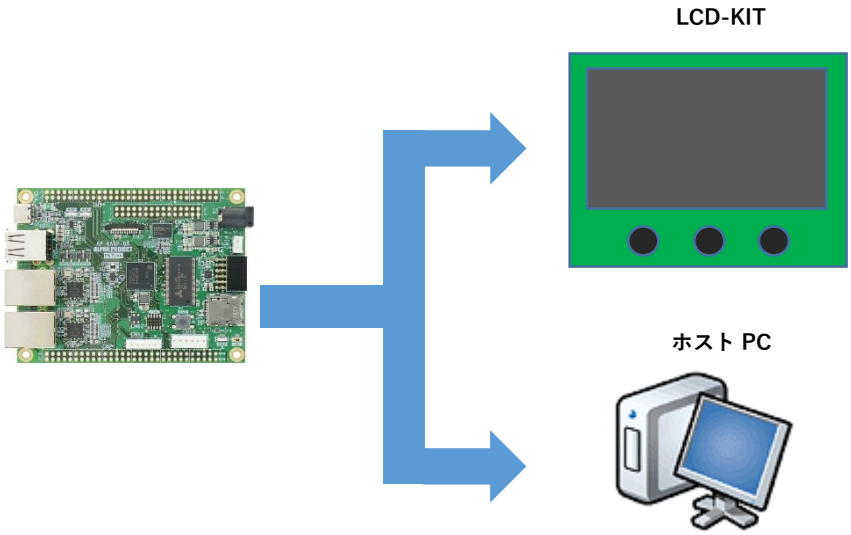


⑦　gray scale 画像を「YOLO_fastest_192」へ入力し、NPU を用いて顔を検出します。
その後、検出した顔の四隅の座標を出力します。

⑧　座標をもとに RGB565 の画像の顔部分を赤枠で囲うように、画像の編集を行います。



⑨　画像の編集が完了した後、画像を LCD で描画します。
また合わせて各処理に要した時間や検出された顔の座標をホスト PC にてターミナルソフトに出力します。

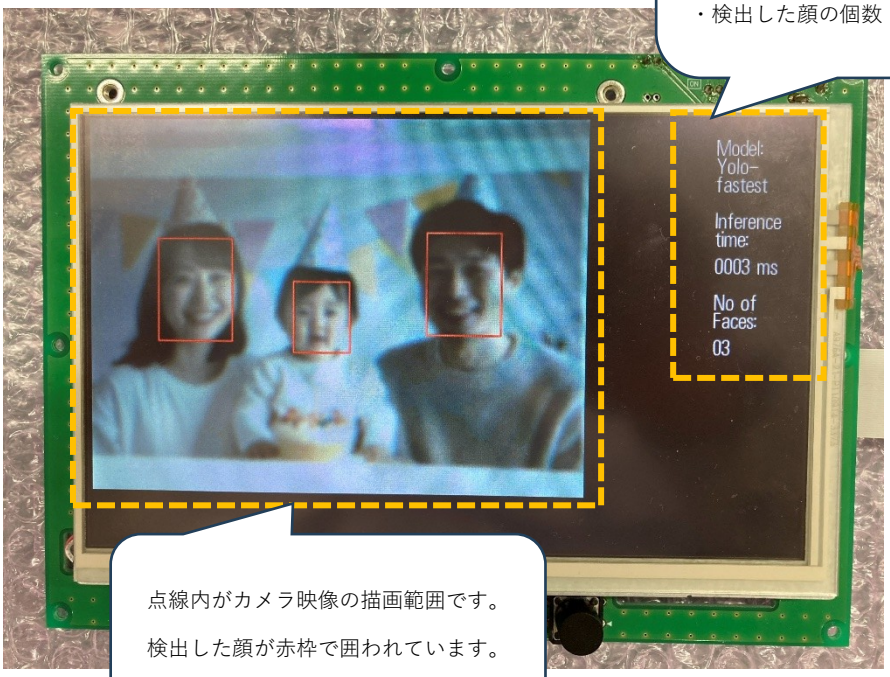


⑩　LCD-KIT の SW1 を押下することで LCD の更新を一時的に止めることができます。
画面が停止した後、画面の更新を再開したい場合もう一度 SW1 を押下してください。
また AI の推論処理ごとに LD1 が点滅します。

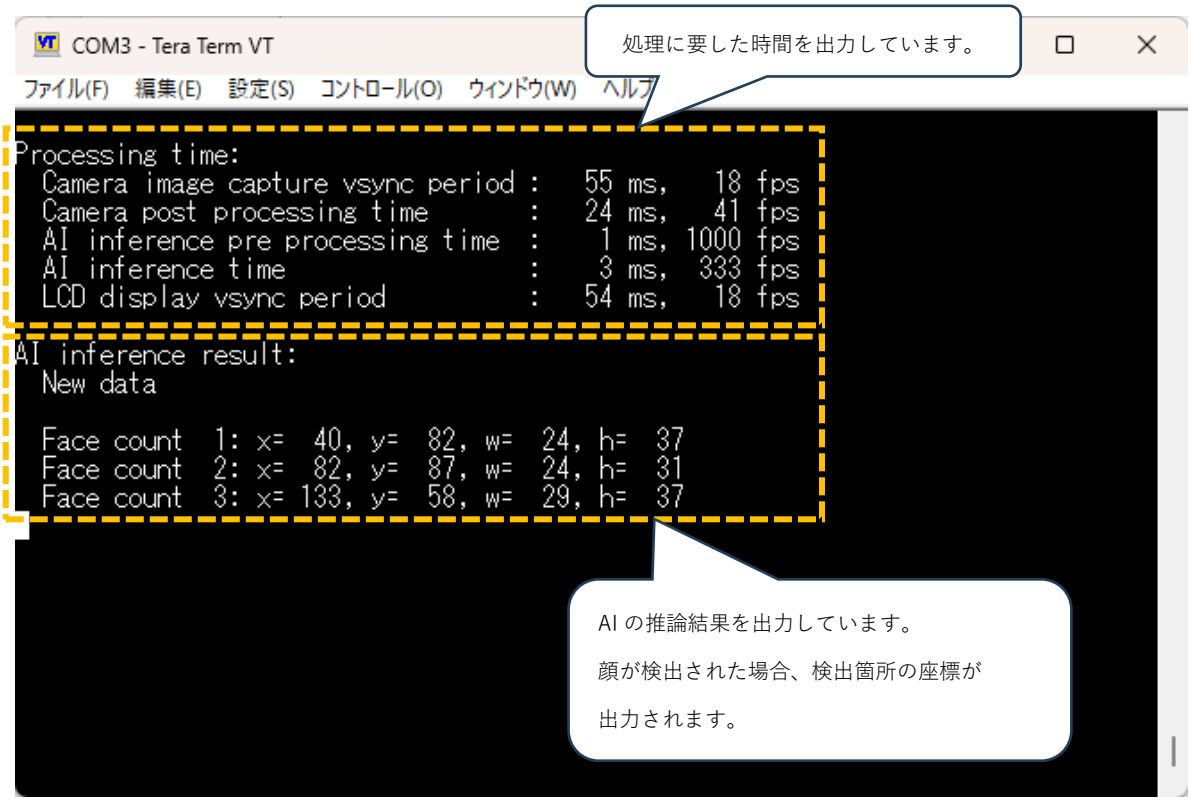
3.1.2　実行結果の例

実際にサンプルプログラムを動作させた際の、画面及びターミナルソフトの出力結果を解説します。
以下の写真をカメラに映した際の出力結果です。

①　LCD の表示結果



②　ターミナルソフトの出力結果



「Processing time」にて出力されている内容は以下の通りです。

名称	内容
Camera image capture vsync period	カメラ画像の取得周期
Camera post processing time	カメラ画像の処理に要した時間
AI inference pre processing time	AI 推論用の前処理に要した時間
AI inference time	AI の推論に要した時間
LCD display vsync period	LCD の画面更新周期

3.1.3　LCD-KIT の選択・設定

本サンプルプログラムは、使用する LCD-KIT によって、下記ソースファイルの変更が必要な場合があります。
変更後は、e2 studio 上で [プロジェクトをクリーンにする]を行ってから [プロジェクトのビルド]を選択して、
必ずプロジェクトを再ビルドしてください。

● LCD-KIT-B01/B02/C01/C02 を使用する場合
プロジェクト「¥sample¥ap_ra8p_0a_face_detection_wvga_sample」を使用します。
デフォルトでは LCD-KIT-C02 を使用する設定になっています。D02/D03 の設定は使用しないでください。

¥sample¥ap_ra8p_0a_face_detection_wvga_sample¥src¥display_layer¥lcdkit¥lcdkit.h（29～38 行目）

```
29：  #define LCDKIT_NONE    (0)
30：  #define LCDKIT_B01     (LCDKIT_NONE + 1)      ←LCD-KIT-B01 と接続時の設定値
31：  #define LCDKIT_B02     (LCDKIT_B01 + 1)       ←LCD-KIT-B02 と接続時の設定値
32：  #define LCDKIT_C01     (LCDKIT_B02 + 1)       ←LCD-KIT-C01/C02 と接続時の設定値
33：  #define LCDKIT_D02     (LCDKIT_C01 + 1)       ←LCD-KIT-D02 と接続時の設定値
34：  #define LCDKIT_D03     (LCDKIT_D02 + 1)       ←LCD-KIT-D03 と接続時の設定値
                                  中略
39：  #define LCDKIT  (LCDKIT_C01)    /* <- Please specify the LCD-KIT to use. */   ←赤字の値を適切に設定してください
```

● LCD-KIT-D02/D03 を使用する場合
プロジェクト「¥sample¥ap_ra8p_0a_face_detection_wqvga_sample」を使用します。
デフォルトでは LCD-KIT-D03 を使用する設定になっています。D02/D03 以外の設定は使用しないでください。

¥sample¥ap_ra8p_0a_face_detection_wqvga_sample¥src¥display_layer¥lcdkit¥lcdkit.h（29～38 行目）

```
29：  #define LCDKIT_NONE    (0)
30：  #define LCDKIT_B01     (LCDKIT_NONE + 1)      ←LCD-KIT-B01 と接続時の設定値
31：  #define LCDKIT_B02     (LCDKIT_B01 + 1)       ←LCD-KIT-B02 と接続時の設定値
32：  #define LCDKIT_C01     (LCDKIT_B02 + 1)       ←LCD-KIT-C01/C02 と接続時の設定値
33：  #define LCDKIT_D02     (LCDKIT_C01 + 1)       ←LCD-KIT-D02 と接続時の設定値
34：  #define LCDKIT_D03     (LCDKIT_D02 + 1)       ←LCD-KIT-D03 と接続時の設定値
                                  中略
39：  #define LCDKIT  (LCDKIT_D03)    /* <- Please specify the LCD-KIT to use. */   ←赤字の値を適切に設定してください
```

3.1.4　ホワイトバランスの選択・設定

本サンプルプログラムは、RAW8 形式から RGB565 形式へデータを変換する際にホワイトバランスの調整を行うことができます。ホワイトバランスの有効・無効及び調整する係数の変更が可能です。
変更後は、e2 studio 上で [プロジェクトをクリーンにする]を行ってから [プロジェクトのビルド]を選択して、
必ずプロジェクトを再ビルドしてください。

なお以下の説明では「¥sample¥ap_ra8p_0a_face_detection_wvga_sample」を例に説明します。
wvga 及び wqvga で変更箇所は同じですので、読み替えて変更ください。

デフォルトでは、両サンプルともにホワイトバランスの調整を行う設定となっています。

● ホワイトバランスの調整を使用する場合

¥sample¥ap_ra8p_0a_face_detection_wvga_sample¥src¥camera_layer¥camera_imx219.c（23～26 行目）

23：	#define WHITE_BALANCE_ADJUSTMENT_ENABLE (1)	ホワイトバランスの調整の 有効・無効設定 1:有効 0:無効
24：	#define WHITE_BALANCE_GAIN_R　(1.0f)	←赤成分のホワイトバランス係数
25：	#define WHITE_BALANCE_GAIN_G　(0.8f)	←緑成分のホワイトバランス係数
26：	#define WHITE_BALANCE_GAIN_B　(1.0f)	←青成分のホワイトバランス係数

● ホワイトバランスの調整を使用しない場合

¥sample¥ap_ra8p_0a_face_detection_wvga_sample¥src¥camera_layer¥camera_imx219.c（23～26 行目）

23：	#define WHITE_BALANCE_ADJUSTMENT_ENABLE (0)	ホワイトバランスの調整の 有効・無効設定 1:有効 0:無効
24：	#define WHITE_BALANCE_GAIN_R　(1.0f)	←赤成分のホワイトバランス係数
25：	#define WHITE_BALANCE_GAIN_G　(0.8f)	←緑成分のホワイトバランス係数
26：	#define WHITE_BALANCE_GAIN_B　(1.0f)	←青成分のホワイトバランス係数

3.2　サンプルプログラムのダウンロード

サンプルプログラムを CPU ボード上で実行するためには、ビルドしたサンプルプログラムの実行ファイルを CPU ボードにダウンロードする必要があります。
サンプルプログラムのビルド方法、CPU ボードにサンプルプログラムをダウンロードして、実行する方法については、以下のアプリケーションノートに詳細な手順が記されています。

・AN2002 RA ファミリ 開発チュートリアル

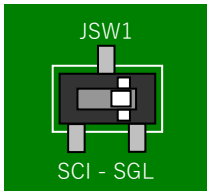
4.　開発環境使用時の各設定値

開発環境を使用する際の、AP-RA8P-0A 固有の設定を以下に示します。
なお、各ファイル名・フォルダ名につきましては、WVGA サンプルプログラムの内容
（ap_ra8p_0a_face_detection_wvga_sample）で記載されておりますので、使用するサンプルプログラムに合わせて、
赤文字の箇所を読み替えてください。

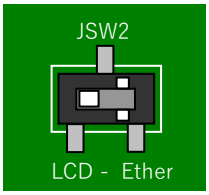
ビルド・動作確認方法	
項目名	設定値
サンプルプログラムフォルダ	sample¥ap_ra8p_0a_face_detection_wvga_sample
プロジェクト	ap_ra8p_0a_face_detection_wvga_sample
デバッグ時のボード設定	「4.1 スイッチ設定」参照
Debug 用出力フォルダ	/ ap_ra8p_0a_face_detection_wvga_sample¥Debug
Debug 用実行ファイル	ap_ra8p_0a_face_detection_wvga_sample.elf
Debug hardware	J-Link ARM
Target Device	R7FA8P1KF

4.1　スイッチ設定

CPU ボードを動作させる際は、動作モードに応じてボード上のディップスイッチを設定する必要があります。



BootMode
SCI　：SCI ブートモード・USB ブートモード
SGL　：シングルチップモード
　　（プログラム動作時やデバッガ使用時）



JSW2
Ether ：Ethernet を使用する
LCD　：LCD を使用する

ご注意

・本文書の著作権は株式会社アルファプロジェクトが保有します。
・本文書の内容を無断で転載することは一切禁止します。
・本文書に記載されているサンプルプログラムの著作権は株式会社アルファプロジェクトが保有します。
・本サンプルプログラムで使用されているミドルウェアおよびドライバの著作権はルネサス エレクトロニクス株式会社が保有します。
・本文書に記載されている内容およびサンプルプログラムについてのサポートは一切受け付けておりません。
・本文書の内容およびサンプルプログラムに基づき、アプリケーションを運用した結果、万一損害が発生しても、弊社では一切責任を負いませんのでご了承ください。
・本文書の内容については、万全を期して作成いたしましたが、万一ご不審な点、誤りなどお気付きの点がありましたら弊社までご連絡ください。
・本文書の内容は、将来予告なしに変更されることがあります。

商標について

・RA8P1 は、ルネサス エレクトロニクス株式会社の登録商標、商標または商品名称です。
・e2 studio は、ルネサス エレクトロニクス株式会社の登録商標、商標または商品名称です。
・Flexible Software Package は、ルネサス エレクトロニクス株式会社の登録商標、商標または商品名称です。
・Robust Unified Heterogeneous Model Integration は、ルネサス エレクトロニクス株式会社の登録商標、商標または商品名称です。
・AI Navigator は、ルネサス エレクトロニクス株式会社の登録商標、商標または商品名称です。

・その他の会社名、製品名は、各社の登録商標または商標です。

ALPHA PROJECT
株式会社アルファプロジェクト
〒431-3114
静岡県浜松市中央区積志町 834
https://www.apnet.co.jp
E-Mail: query@apnet.co.jp