TOPPERS 活用アイデア・アプリケーション開発コンテスト

部門　：　アプリケーション開発部門

作品のタイトル　：　Raspberry Pi Pico への TOPPERS/ASP の移植

作成者　：　石岡之也（個人）

共同作業者　：

対象者　：　TOPPERS/ASP 利用者、組込みソフト開発者

使用する開発成果物　：　TOPPERS/ASP 1.9.3
ARM Cortex-M0 アーキテクチャ・GCC 依存部パッケージ
コンフィギュレータ Release 1.9.6

目的・狙い

Raspberry Pi 財団が開発、販売している Raspberry Pi Pico が安価でありながら 264KB の RAM、2MB の Flash メモリ、26 本の GPIO と電子工作などで多くの利用が見込める。
また、ＣＱ出版の Interface 誌編集者から TOPPERS の移植記事を執筆しないかとの打診があったことから Raspberry Pi Pico と同じコアの Cortex-M0+へ対応済みの TOPPERS/ASP を Raspberry Pi Pico へ移植することとした。

アイデア/アプリケーションの概要

TOPPERS/ASP の ARM Cortex-M0 アーキテクチャ・GCC 依存部パッケージがサポートする STM32L073RZ-Nucleo のマイコン STM32L073RZ が Cortex-M0+であることから STM32L073RZ 用のコードを改造し、Raspberry Pi Pico 用の TOPPERS/ASP を開発した。
また、Raspberry Pi 財団から提供されている SDK の起動処理では CPU の動作クロックが 125MHz であるが、CPU の能力として 133MHz まで利用可能なことから 133MHz への改造も行った。

１．開発の背景

Raspberry Pi財団が開発､販売しているRaspberry Pi Picoが安価でありながら264KBのRAM､2MBのFlashメモリ、26本のGPIOと拡張性が高いこと、搭載デバイス用のSDKも提供されSPIやUSBを使ったプログラムを比較的容易に開発できることから電子工作などで多くの利用が見込める。

また、CQ出版社のInterface誌編集者からRaspberry Pi Picoに関する執筆依頼があり、編集者との検討で過去のTOPPERSコンテストの入賞経験を生かしてRaspberry Pi Pico用の TOPPERSを移植してはとの提案があった。

TOPPERSの利用経験はTOPPERS/ASPであることから対応ボードとCPUを確認し､ARM Cortex-M0アーキテクチャ･GCC依存部パッケージのSTM32L073RZ-Nucleoボード用のコードが Raspberry Pi PicoのCPU、RP2040の同一コアのCortex-M0+用であることが分かり、このコードを流用してRaspberry Pi Pico用のTOPPERS/ASPを開発することとした。

２．開発環境

Cortex-M系の TPPERS/ASPのビルド環境は、Qiitaへ投稿した以下の記事で構築できる。

ラズパイを使ったTOPPERS用Cortex-M開発環境の構築 < https://qiita.com/Yukiya_Ishioka/items/e68a6aabf72f0721948d>

Raspberry Piを利用した環境構築の概要を以下に記す。

１）Raspberry Pi 3/4へ ubuntu 20.04系 serverをインストールする。
２）TOPPES/ASPに必要な以下のパッケージを追加インストールする。
- libboost-all-dev
- libxerces-c-dev
- unzip
- net-tools
- g++
- make

３）クロスツールチェーンのインストールする。
GNU Arm Embedded Toolchain Downloads のページからクロスツールチェーンをインストールする。
<
https://developer.arm.com/-/media/Files/downloads/gnu-rm/10-2020q2/gcc-arm-none-eabi-10-2020-q2-preview-aarch64-linux.tar.bz2?revision=a7134e5c-fad8-490c-a62b-9200acca15ef&la=en&hash=12954B763712885C1DDE3318D10DF96B1606463A>

４）TOPPERS コンフィギュレータを準備する。
以下のURLからコンフィギュレータをダウンロードし、Qittaの記事に従い展開、修正、configure、ビルドを行い cfgコマンドを生成する。
<https://www.toppers.jp/download.cgi/cfg-1.9.6.tar.gz>

５）TOPPERS/ASPのソースコードを準備する。
以下のURLからTOPPERS/ASPをダウンロードし、展開する。
<https://www.toppers.jp/download.cgi/asp-1.9.3.tar.gz>
< https://www.toppers.jp/download.cgi/asp_arch_arm_m0_gcc-1.9.4.tar.gz>

ただし､開発したRaspberry Pi Pico用のソース利用して実行モジュールを作成する場合、ARM Cortex-M0アーキテクチャ・GCC依存部パッケージのダウンロード、展開は不要。

３．開発内容

TOPPES/ASP の ARM Cortex-M0 アーキテクチャ・GCC 依存部パッケージを使い、STM32L073RZ-Nucleo 用のコードから Raspberry Pi Pico 対応の内容を以下に記す。

なお、ASP カーネルのソースコード内の定義と Pi Pico SDK 内の定義でコンフリクトなどの発生が予想されるため、SDK を呼び出す処理は新たに作成した以下のファイルに集約し、ASP カーネル内のコードから SDK を直接には呼び出さず、作成したファイル内の関数を呼び出すよう改造を行った。

```
arch/arm_m_gcc/pico/pico_dev.c
```

（1）Raspberry Pi Pico SDK 入手

Raspberry Pi Pico の初期化や搭載デバイスへのアクセス用コードが含まれる Raspberry Pi Pico SDK を以下のページからダウンロードする。

<https://github.com/raspberrypi/pico-sdk.git>

TOPPERS/ASP や SDK を以下のような構成になるよう展開する。

```
|--asp
|  |--arch
|  |  |--arm_m_gcc
|  |
|  |--cfg
|  |  |--cfg          ・・・コンフィギュレータ配置用ディレクトリ
|  |
|  |--configure       ・・・configure スクリプト
|  |--include
|  |--kernel
|  |--sample
|  |--syssvc
|  |--target
|
|--build
|  |--obj             ・・・TOPPERS/ASP 用ディレクトリ
|
|--cfg                ・・・コンフィギュレータソース用ディレクトリ
|  |--cfg             ・・・コンフィギュレータ生成ディレクトリ
|  |--configure       ・・・コンフィギュレータ用 configure スクリプト
|
|--elf2uf2            ・・・elf2uf2 コマンド生成用ディレクトリ
|--pico-sdk           ・・・Pico SDK 展開ディレクトリ
|  |--src
|  |--tools
|  |  |--elf2uf2      ・・・elf2uf2 コマンドソースファイル
```

（2）コアの停止

Pi Pico の CPU である RP2040 はデュアルコアであるが、今回流用した TOPPERS/ASP はシングルコア用のコードであるため、CPU 起動直後に 1 コアを停止する改造を start.S に対して行った。

```
/*
 *  スタートアップルーチン
 *
 *  Thread モードで呼び出されることを前提としてる.
 */
        ATEXT
        ABALIGN(4)
        ATHUMB(_start)
        AGLOBAL(_start)
ALABEL(_start)
        cpsid f                 /* 割込みロック状態へ */

        /* for PR2040 */                                    ┐
        ldr r0, =(SIO_BASE + SIO_CPUID_OFFSET)              │
        ldr r0, [r0]                                        ├ 追加コード
        cmp r0, #0                                          │
        bne hold_non_core0_in_bootrom                       ┘

#ifdef INIT_MSP
```

（3）システムクロックの対応

Pi Pico の動作クロックを設定するため、SDK 内の clocks_init()を呼びだすよう改造する。

クロック設定は ASP の target_inithook.c 内の以下の関数を改造し、Pi Pico 対応させる。

sysclock_init_value()

sysclock_change()

（4）オンチップハードウェア初期化

ASP の target_inithook.c 内の以下の関数を改造し、Pi Pico のオンチップハードウェア初期化が行るようにする。

hardware_init_hook()

（5）tick タイマ

Cortex-M0+のベースである ARMv6-M アーキテクチャでは、systick タイマはオプション扱いでプロセッサごとの対応が必要になる可能性があったが、流用元の STM32L073RZ、対応先の Pi Pico とも systick タイマをサポートしていることから STM32L073RZ 用のコードそのまま利用した。

（６）シリアルコンソール用 UART

syslog などの入出力で使用する UART の設定を追加し、chip_serial.c などで実行する UART に対する処理を SDK 内の処理へ置き換えるための改造を行った。

コンソールとして使用する Pi Pico のピンは以下を利用するよう設定した。

```
#define UART_TX_PIN 0
#define UART_RX_PIN 1
```

また、UART の割り込みは、以下の通り IRQ の２０番、割り込み番号「３６」になるよう設定を行った。

```
pico-sdk/src/rp2040/hardware_regs/include/hardware/regs/intctrl.h
#define UART0_IRQ 20
```

```
#define INHNO_SIO1      (UART0_IRQ+16)
#define INTNO_SIO1      (UART0_IRQ+16)
#define INHNO_SIO2      (UART1_IRQ+16)
#define INTNO_SIO2      (UART1_IRQ+16)
```

（７）CPU クロック 133MHz 化

Pi Pico の CPU である RP2040 は下図のように最大 CPU クロック 133MHz まで利用可能である記述があるが、SDK のクロック設定処理では 125MHz となっている。

このため、今回、133MHz で動作する改造も行った。

Chapter 1. About the RP2040

RP2040 is a low-cost, high-performance microcontroller device with flexible digital interfaces. Key features:

- Dual Cortex M0+ processors, up to 133 MHz
- 264 kB of embedded SRAM in 6 banks
- 30 multifunction GPIO
- 6 dedicated IO for SPI Flash (supporting XIP)
- Dedicated hardware for commonly used peripherals
- Programmable IO for extended peripheral support
- 4 channel ADC with internal temperature sensor, 0.5 MSa/s, 12-bit conversion
- USB 1.1 Host/Device

SDK の CPU クロック設定、および、CPU クロックに依存する設定処理は、以下の clocks_init()関数内で行っている。この中の設定値を 133MHz 用に変更することで CPU を 133MHz で動作させることができるようになる。

pico-sdk/src/rp2_common/hardware_clocks/clocks.c

```
145     /// ¥tag::pll_settings[]
146     // Configure PLLs
147     //                   REF     FBDIV VCO            POSTDIV
148     // PLL SYS: 12 / 1 = 12MHz * 125 = 1500MHZ / 6 / 2 = 125MHz
149     // PLL USB: 12 / 1 = 12MHz * 40  = 480 MHz / 5 / 2 =  48MHz
150     /// ¥end::pll_settings[]
151
152     reset_block(RESETS_RESET_PLL_SYS_BITS | RESETS_RESET_PLL_USB_BITS);
153     unreset_block_wait(RESETS_RESET_PLL_SYS_BITS | RESETS_RESET_PLL_USB_BITS);
154
155     /// ¥tag::pll_init[]
156     pll_init(pll_sys, 1, 1500 * MHZ, 6, 2);  ・・・(1)
157     pll_init(pll_usb, 1, 480 * MHZ, 5, 2);  ・・・(2)
158     /// ¥end::pll_init[]
         :
169     // CLK SYS = PLL SYS (133MHz) / 1 = 133MHz
170     clock_configure(clk_sys,
171                     CLOCKS_CLK_SYS_CTRL_SRC_VALUE_CLKSRC_CLK_SYS_AUX,
172                     CLOCKS_CLK_SYS_CTRL_AUXSRC_VALUE_CLKSRC_PLL_SYS,
173                     125 * MHZ,
174                     125 * MHZ);
           :
198     // CLK PERI = clk_sys. Used as reference clock for Peripherals. No div・・
199     // Normally choose clk_sys or clk_usb
200     clock_configure(clk_peri,
201                     0,
202                     CLOCKS_CLK_PERI_CTRL_AUXSRC_VALUE_CLK_SYS,
203                     125 * MHZ,
204                     125 * MHZ);
```

また、ASP カーネルが使用する systic タイマは CPU クロックを用いてカウントが行われるため、ASP カーネルの target_inithook.c 内の以下の設定を変更し、ASP カーネルも 133MHz で動作するよう変更する。

```
uint32_t
sysclock_init_value(void)
{
        return 125000000;       /* 125 MHz */
}


int
sysclock_change(uint8_t range)
{
        pico_clock_init();
        SystemFrequency = 125000000;
        return  0;
}
```

※ダウンロードして入手可能な Raspberry Pi Pico 用の TOPPERS/ASP 用のコードでは以下のパスになる。
ダウロード先などの情報は「最後に」を参照。

```
target/pico_gcc/target_inithook.c
```

４．最後に

TOPPERS/ASP の Raspberry Pi Pico の移植は CQ 出版 Inetrface 誌 2021 年 8 月号に私が執筆した記事が掲載され、すでに公開されています。

< https://interface.cqpub.co.jp/magazine/202108/>

< https://interface.cqpub.co.jp/wp-content/uploads/if2108_109.pdf>

「３．開発内容」のうち「（７）CPU クロック 133MHz 化」以外のソースコードは Interface 誌のダウンロードサイトから入手可能で、以下の URL からダウンロード可能です。

https://www.cqpub.co.jp/interface/download/2021/08/IF2108TOPP.zip

「（７）CPU クロック 133MHz 化」分の修正ファイルをコンテスト応募のソースファイルとして添付します。

以上