# mruby/cの PIC24 へのポーティング手順について

しまねソフト研究開発センター

### 1.はじめに

mruby/c を Microchip 社製1chip マイコン PIC24 シリーズ用の Explorer 16/32 Development Board 上で動作させるための手順を記述する。

### 2.使用機材及び開発環境

2.1 使用機材

Explorer 16/32 Development Board (P/N DM240001) PIC24FJ256GB210搭載 Explorer 16/32 用ドータボード (P/N MA240021) windows 8.1

### 2.2 開発環境

MPLAB X Version 5.0.5 (NetBeans ベースの統合開発環境) MPLAB XC16 Compiler Version 1.35 (Free License を使用) MPLAB Code Configurator Version 3.65.1

### 3.開発環境の構築方法

3.1 MPLAB ツール群のインストール 1.ダウンロードサイト MPLAB X 下記 URL の画面下方のダウンロードからダウンロードする。 https://www.microchip.com/mplab/mplab-x-ide

直接リンクは、下記であるが、バージョン UP 等により変更になる可能性が高いので上記 URL を推奨する。 https://www.microchip.com/mplabx-ide-windows-installer

#### MPLAB XC16

MPLAB X インストールの最後にインストールを促すメッセージが出るが、web サイトを立ち上げるだけなので、 あらかじめダウンロードしておくと良い。

下記 URL の画面下方のダウンロードからダウンロードする。

https://www.microchip.com/mplab/compilers

直接リンクは、下記であるが、バージョン UP 等により変更になる可能性が高いので上記 URL を推奨する。 https://www.microchip.com/mplabxc16windows 2.インストール

MPLAB X

1にてダウンロードした「MPLABX-v5.05-windows-installer.exe」を実行する事により、統合開発環境がインストールされる。

MPLAB X インストールの最後に XC のインストールを促すメッセージが出るが、ダウンロードして有る場合には、 そのまま終了するとよい。

XC16

1にてダウンロードした「xc16-v1.35-full-install-windows-installer.exe」を実行する事により、統合開発環境がインストールされる。

MPLAB Code Configurator

MPLAB Code Configurator (以下 MCC と略す)は、MPLAB X の Plug-in なので、インストールは、MPLAB X を立ち上げ行う。

・1.「Tools」->「Plugins」を選択する。



図 3-2.1.1 Plugins 設定の立ち上げ

・2.「Available Plugins」タブから「MPLAB Code Configurator」を探し、「Install」チェックボックスにチェック を入れ、「Install」ボタンを押す。

Checl	k <u>f</u> or Newest				Search
İnstall	Name	Category	Source		
	Power Monitor	MPLAB Plugin	66		MPLAD& Code Configurator
Н	BTOS Viewer (FreeBTOS)	MPLAB Plugin	66		All Community Constributed Bluein
П	FGAN Bit Bate Galculator	MPLAB Plugin	66		W Community Contributed Flugin
П	POL int	MPLAB Plugin	66		Version: 366
Π	DMGI	MPLAB Plugin	66		Author: Microchip Technology Inc
П	Halt Notifier (Trial)	MPLAB Plugin	66		Date: 18/10/17
Π	Remote LISB Debugging (Trial V	MPLAB Plugin	66		Source: Microchip Plugins
П	Plugin Update Services	MPLAB Plugin			Tromepage. Interview increasing compression ince
Π	USB Tool Connection Diagnostics	MPLAB Plugin	66		
П	Doxygen Integrator	MPLAB Plugin	66		Plugin Description
<b>v</b>	MPLAB® Code Configurator	MPLAB Plugin	<u></u>		The MPLAB® Code Configurator (MCC) generates seamless easy to understand C
	MPLABX KeeLog Plugin	MPLAB Plugin	<b>66</b>		code that's inserted into your project. It enables, configures and utilizes a rich set
Π	App Launcher	MPLAB Plugin	<u>66</u>		of peripherals across a select list of devices. It's integrated into MPLAB X (IDE) to
П	MemoryStarterkit	MPLAB Plugin	ÅÅ		provide a very poweriul and extremely easy to use development platform.
Π	Code Profiling (Trial Version)	MPLAB Plugin	ŝõ		
	dsPICWorks	MPLAB Plugin	<u>.</u>		System requirements
	Save As v4xx Project	MPLAB Plugin	ŵŵ		■ MPLAB X: v5 10
	Digital Compensator Design Tool	MPLAB Plugin	<u>-</u>		
	MPLAB® Harmony Configurator	MPLAB Plugin	<u>-</u>		
	Simple Serial Port Terminal	MPLAB Plugin	<b>66</b>		Visit the MCC website for user's guides and release notes containing the lists of
	SEGGER JLink Probe	MPLAB Plugin	<b>66</b>		supported devices.
	Come la Ducolue a înt Diveria	MOLAD DU.	<u>88</u>	*	

図 3-2.2.1 Plugins 選択及びインストール方法

・3.注意事項等が表示されるので、確認しインストールを行う。

・4. MPLAB X をいったん終了し、再度起動するとインストール完了する。

4.mruby/cインポート
今回は、最新版のmruby/cバージョン 1.2を PIC24FJ256GB210 ヘインポートする。
ダウンロードは、下記より行う。
https://github.com/mrubyc/mrubyc/
また、mruby コンパイラは、バージョン 1.4 以降のが必須なため、合わせて下記よりダウンロードする。
http://www.s-itoc.jp/activity/research/mrubyc/mrubyc\_download/

なお、mruby/c の動作を確認する方法としては、itoc サイトにて公開されているチュートリアル1~3 が動く事を確 認することとする。

http://www.s-itoc.jp/activity/research/mrubyc/mrubyc\_tutorial/

4.1 PIC24F 環境への mruby/c インポート

4.1 itoc チュートリアル「Chapter01「LED 点滅」」を PIC24F の環境で動かす

itoc チュートリアル「Chapter01「LED 点滅」」

http://www.s-itoc.jp/activity/research/mrubyc/mrubyc\_tutorial/735

1.MPLAB X で新規プロジェクトを作成する。

「File」->「New Project」を選択し、新規プロジェクトを作成する。



図 4.1.1-1 新規プロジェクト作成

2.「New Project」設定画面が出るので、 「Microchip Embedded」の 「Standalone Project」を選択し、 「Next」ボタンを押す。



# 図 4.1.2-1 プロジェクト作成

3.「New Project」設定画面が出るので、 Family 欄に「16-bit MCUs (PIC24)」、 Device 欄に「PIC24FJ256GB210」 を表示させ「Next」ボタンを押す。

\$		New Project	
Steps	Select Devic	e	
1. Choose Project 2. Select Device 3. Select Header 4. Select Tool 5. Select Project Name and 6. Select Compiler 7. Select Project Name and Folder	Family: Device:	All Families PIC24FJ256GB210	v
MPLAB X IDE			
		< <u>B</u> ack Next > E	iinish Cancel <u>H</u> elp

# 図 4.1.3-1 プロジェクト作成

4. 書き込みツールの設定画面になるので、 「Microchip Starter Kits」 を選択し「Next」ボタンを押す。

8	New Project ×
Steps	Select Tool
1. Choose Project 2. Select Device 3. –	Hardware Tools AtmeHDE Solo 1 Solo 1 Solo 4 Solo 2 Solo 4 Solo 2 Solo 4 Solo 2 Solo
	Karl Karl Karl Karl Karl Karl Karl Karl

図 4.1.4-1 書き込みツール選択

5. コンパイラの設定画面になるので、 「XC16(V1.35)」 を選択し「Next」ボタンを押す。



図 4.1.5-1 コンパイラ選択

6.プロジェクト名の設定画面になるので、 任意のプロジェクト名(ここでは、「mrubyc\_led1」) を入力し「Finish」ボタンを押す。

Steps	Select Project Nam	e and Folder	
1. Choose Project 2. Select Device 3. Select Header 4. Select Tool 5. Select Plugin Board 6. Select Compiler 7. Select Project Name and Folder	Project Name: Project Location: Project Folder:	mrubyc_led I  C/#W0 RK#PIC#mrubyc C/#W0 RK#PIC#mrubyc/#mrubyc_led I X	Browse
MPLAR	Overwrite existin Also delete soun ✔ Set as main proj Use project locan	ne project. ces. cet tion as the project folder	
XIDE	Encodine: UT	F-8 v	
		( Back Nevt ) Finish	Gancel Help

図 4.1.6-1 プロジェクト名入力

7.MCC を使いピンアサインを決定する

MCC でソースコードの自動生成を行うので、マージ作業を避けるため、ソースコードの無い状態で MCC を使いピンアサインなどを決定しておくと効率的である。

ここでは、EXPLORER 16/32の

LEDsと Buttons のアサインを決定する。

まず、「MCC」ボタンを押し、MCC を 起動する



図 4.1.7-1 プロジェクト画面

MCC のコンフィグレーションファイル名の 設定画面になるので、任意の名前を入力し 「保存」ボタンを押す。

問題なければ、初期値のままで良い。

	Si	ave MCC Configuration File			×
⊕ ∋ ≤ ↑ → PC + Wi	indows (C:) → WORK → PIC →	mrubyc + mrubyc_Led2.X +	v ¢	mrubyc_Led2.Xの検索	,p
整理 ▼ 新しいフォルダー				# •	- 0
★ お気に入う     ▲ ダウンロード     ■ デスフトップ     ■ 単近原 沢した場所     ■ デスフトップ     ■ 単近原 沢した場所     ■ グランロード     ■ デスフトップ     ドキュント     ビジオ     ■ ビジオ     ■ ビジオ     ■ ビジオ     ■ ジェージック     ■ Windows (C)       ボリエーム (C)     ■ ビデオ     ■ ビデオ	名前 Gebug Gebroject	更新日前 2018/10/15 13:04 2018/10/15 13:04	種類 ファイル フォルダー ファイル フォルター	942	
ファイル名(N): MyConfig.me ファイルの種類(I): MCC Configu	c3 uration Files (*.mc3)				v
● フォルダーの非表示				保存( <u>S</u> ) キャ	ンセル

図 4.1.7-2 ファイル名入力

ubyc\_Led1\_210 : default MPLAB X IDE v5.05 – 🗗 🗙 8 Eile Edit Viev Refactor 1 🔁 🖴 🔀 🐂 💼 🦻 🦿 defaul 1 🕨 - 👱 - 🔁 - 🍋 🚯 - PC:0x0 dc n ov z c How do 19 286:87461 .t0c 😢 main.c 🛛 🖭 halh 🗶 🖭 clockh 🗶 Pin Modul Interrupt Module × System Module × • • • □ Pin Projects Services Classes Re × : Package View × System Module 0 Project Ger rate In 🛞 Easy Setup 🗐 🗄 Clock + Hz FRC Oscillator ▼ (8.0 N • FRC Po 4 MHz 1:2 \* Postscaler PLL Enable 803 4 804 9 805 10 807 13 804 10 809 1 909 1 909 1 ources ۷ 4 MHz Fosc MICROCHIP PIC2 CR210 Product Page 2 MH; Fosc/2 USB Clock PIC24FJ256GB210 N2-5 7229 F305 F304 N244 F305 F305 F305 F305 Pin Config ) uration OSCO/CLKO/RC15 functions as CLKO (FC (31 - 33) kH dary Oscillator Interrupt ubyc\_Led1\_210 - Da ard Na tor Versions [MCC] × Versions 
 Versions

 MPLA8\* Code Configurator (Plugin) v3.65.1

 • Uibraries

 • Microcontrochiles and Peripherals

 • Microcontrolles and Peripherals

 • B-bent AVR MCUs (v1.00)

 • AVR-107 WCS ensor Node

 • MCP15000C (v1.1)

 • PICE10 / PIC12 / PIC16 / PIC18 MCUs (v1.75)

 • PIC24 / AprCl3 / PIC32MM MCUs (v1.75)

 • PIC32MX MCUs (v1.35)
 Eail ▼ ICD 
 Output
 Notifications [MCC]
 Pin Manage

 Package:
 TQFP100
 ▼
 Pin No:
 7 8 9 63 73 74 64 72 76 77 78 81 82 83 5 24 23 22 21 20 26 27 32 17 38 58 Port A V Port B 🔻 0 1 2 3 4 5 6 7 9 10 14 15 0 1 2 3 4 5 6 7 8 10 14 15 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 1 2 3 4 12 13 14 15 1 2 3 4 12 13 14 15 1 2 3 4 12 13 14 15 1 2 3 4 12 13 14 15 1 2 3 4 12 13 14 15 1 2 3 4 12 13 14 13 2 3 4 5 6 Di + â CLKO output î OSCI OSCO Software
 8-bit Bootloader Library (v2.2.31)
 47XXX I2C EERAM î sosc input output sosco CoAP Library
 Dac Library PCCV î. î î Foundation Services Library (v0.1.27)
 I2C EEPROM nput **b** b b **b b b b** . . . . . . . â b ิษ โต î. în în ĵ∎ ĵ∎ Pin Module 🔻 LIN Library (v2.3) GPIO LoRaWAN Library 2

保存すると、MCC のメイン画面が開く。

Device Resources の一覧枠、設定項目選択の枠と、設定のメイン枠、ピンアサインの枠が表示される。

図 4.1.7-3 MCC メイン画面

右上のチップの絵については、ピンの状況の表示を行っている。

8.MCC のコンフィグレーションを開始する。 設定項目選択の枠で、「System Module」を選択する。 「System Module」は、クロック等の設定を行う画面となる。

Project Resources	Generate	Import	Export
▼ System			
Interrupt Module			
Pin Module			
System Module			

図 4.1.8-1 設定選択画面

System Module については、ICD(In Curcuit Debugger)の設定を変更する以外は初期値のままで問題ない。 EXPLORER 16/32 と MA240021 の組み合わせでは、ICD は、「PGEC2/PGED2」に接続されるので、そのように設定する。

System Module	
🛱 Easy Setup 📃 Reg	isters
▼ ✓ FRC Postscaler	
4 MH:	z 1:2 - Postscaler
▶ PLL Enable	
4 MHz	Fosc
2 MHz	Fosc/2
	USB Clock
Clock Output Pin Config	guration OSCO/CLKO/RC15 functions as CLKO (FOSC/2)
Use Secondary Osci	llator (31 - 33) kHz
Enable Clock Switch	<b>ling</b> Monitor
▼ ICD	
Emulator Pin Placen ent	Emulator functions are shared with PGEC2/PGED2
<ul> <li>Watchdog</li> </ul>	

図 4.1.8-2 System Module 設定選択画面

 設定項目選択の枠で、「Pin Module」を選択する。
 「Pin Module」の設定は、GPIO 等のピンアサイン設定を 行う画面となる。

Project Resources	Generate	Import	Export
▼ System			
Interrupt Module			
Pin Module			
System Module			

図 4.1.9-1 設定選択画面

ピンアサイン設定画面で機能と使用するピンを設定し、「Pin Module」設定画面で詳細を設定する。 プログラム中で分かり易いように「Custom Name」を設定しておくと良い。



#### 図 4.1.9-2 ピンアサイン設定画面

Pin Module			I							0
Selected Packag	e 🗄 Register ge : TQFP100	s								
Pin Name 🔺	Module	Function	Custom Name	Start High	Analog	Output	WPU	WPD	OD	IOC
RA0	Pin Module	GPIO	O_LED3			$\checkmark$				none 💌
RA1	Pin Module	GPIO	O_LED4			$\checkmark$				none 💌
RA2	Pin Module	GPIO	O_LED5			$\checkmark$				none 🔻
RA3	Pin Module	GPIO	O_LED6			$\checkmark$				none 💌
RA4	Pin Module	GPIO	O_LED7			$\checkmark$				none 👻
RA5	Pin Module	GPIO	O_LED8			$\checkmark$				none 👻
RA6	Pin Module	GPIO	O_LED9			$\checkmark$				none 👻
RA7	Pin Module	GPIO	IO_S5_LED10							none 👻
RB6	ICD	PGEC2								none 👻
RB7	ICD	PGED2								none 👻
RD6	Pin Module	GPIO	I_S3							none 👻
RD7	Pin Module	GPIO	I_S6							none 👻
RD13	Pin Module	GPIO	I_54							none 👻

図 4.1.9-3Pin Module 設定画面

10.MCC の設定をソースに展開する。 「Generate」ボタンを押し、MCC の設定をソースに展開する。 MCC で変更を行った際には、この操作を行わないと、ソースに 反映されないので注意が必要である。

Project Resources	Generate	mport	Export
▼ System			
Interrupt Module			
Pin Module			
System Module			

#### 図 4.1.10-1 設定選択画面

11.mruby/c VM ソースコードを、プロジェクトへ追加

・mruby/c ソースコードを展開し、src フォルダー内の全ファイルを src フォルダーごとプロジェクトフォルダー (項目 6 で保存したフォルダ内)にコピーする。

・src フォルダー内の hal\_psoc5lp フォルダーを hal にリネームする。

・src フォルダー内の hal\_posix、hal\_esp32 フォルダーを消去する。

・「mrubyc\_Led1」の統合環境プロジェクト内にソースを登録する。

下記手順で、ソースを登録する。

プロジェクトツリーを表示し、「Header Files」の項目で、右クリックし、コンテキストメニューで、「Add Existing Items From Folders」を選ぶ。

Projects × Files	Services	Classes	
en mrubyc_Led1	_		
⊕-^ (	New New Add	Logical Folder Existing Item	•
🕀 🚡 Libraries	Add	- Existing Items	from Folders
⊕- 🕢 Loadables	Find Cut Copy Past	•• • • •	Ctrl+V
	Rena	ime	
	Prop	erties	

図 4.1.11-1 mruby/cヘッダファイル登録

下記の「Add Files」のダイアログが出るので、「Add Folder」ボタンを押す。

\$	Add Files	×
Specify the folders that contain the sour	e files for the project	
Source File Folder	Files of Type	Add Folder.
Ignored Folders Pattern:		
(See also IDE level file filter in Options-	>Miscellaneous=>Files)	Detagin

図 4.1.11-2 mruby/cヘッダファイル登録

下記の「Select Source File Folder」のダイアログが出るので、ファイルのタイプを「Header Files」にした後に mruby/c の src フォルダーを選択し、「Select」ボタンを押す。



図 4.1.11-3 mruby/cヘッダファイル登録

下記の「Add Files」のダイアログに戻るので、「Add」ボタンを押しヘッダファイルを登録する。

3	Add Files	×
Specify the folders that contain the s	ource files for the project:	
Source File Folder	Files of Type	Add Folder
src	Header Files (H SUNWCCh h hpp hox too ino 1	NC) <u>R</u> emove
Innored Folders Pattern:		
		Default
(See also IDE level file filter in Optio	ns->Miscellaneous->Files)	
Source files from specified folders w	ill be added to the project.	

図 4.1.11-4 mruby/cヘッダファイル登録

次に、ソースファイルを登録する。ヘッダファイルの時と同様にプロジェクトツリーを表示し、「Source Files」の項目で、右クリックし、コンテキストメニューで、「Add Existing Items From Folders」を選ぶ。

「Add Files」のダイアログが出るので、「Add Folder」ボタンを押す。

下記の「Select Source File Folder」のダイアログが出るので、ファイルのタイプを「C Source Files」にした後に mruby/c の src フォルダーを選択し、「Select」ボタンを押す。

8	Select Source File Folder		×
<ul> <li>最近使った項</li> <li>デスクトップ</li> <li>ドキュメント</li> </ul>	ファイルの場所の:   mrubyc led 1X  mrubyc led 2X  build  debue  build  debue  build  debue  build  for a cynerrated files  mrubyc led 2X  build  debue  build  for a cynerrated files  mrubyc led 1  for a cynerrated files  mrubyc led 2	2	P
PC			
<b>ریا</b> رد ۲-۲۱رد	レ ファイル名(N): CXWORKWPDWmrubycVmrubyc_Led2XWsrc ファイルのタイプ(T): C Source Files (c i m) ・	Sele 取i	ct lá

図 4.1.11-5 mruby/cソースファイル登録

下記の「Add Files」のダイアログに戻るので、「Add」ボタンを押しヘッダファイルを登録する。

3	Add Files	
Specify the folders that contain the so	purce files for the project:	
Source File Folder	Files of Type	Add Folder
src	C Source Files (ɛ i m)	Bemove
gnored Folders Pattern: See also IDE level file filter in Option	ns->Miscellaneous->Files)	Defa <u>u</u> lt
Source files from specified folders wi	II be added to the project.	
oource mes nom specified folders wi	n be dade to ale project.	Add Cancel

図 4.1.11-6 mruby/cソースファイル登録

プロジェクトプロパティの設定を設定する。設定内容一覧は下記である。

設定場所1	設定場所2	設定項目	内容
XC16	Most Useful Options	C include dirs	src,src¥hal
XC16	Most Useful Options	Define common macros	MRBC_NO_TIMER
xc16-ld	General	Heap size	1024
xc16-ld	General	Min stack size	1024

表 4.1.12-1 プロジェクトプロパティの設定

プロジェクトツリーのルート、プロジェクト名の所で右クリックし、プロジェクトのプロパティを選択する。



図 4.1.12-1 プロジェクトプロパティの設定

「Project Propeties」ダイアログが開くので、左ペインで「設定場所1」を選択(ここでは「XC16」)し、右ペインの「Option categories」で「設定場所2」を選択(ここでは、「Most Useful Options」)を選び、「設定項目」の右側の欄をクリックすると数字入力や、パラメータ入力が出来る。

	Proje	Project Properties - mrubyc_Led1					
Catecories:	Options for xc16-g Option categories:	Options for xc16-ecc (v1.35) Option categories: Most Useful Options v					
	Optimization level	Optimization level					
- • Loading	Common include di	Common include dirs					
O Libraries     Duilding	C include dirs		src;src¥hal				
	Code model		[default]				
- · XC16 (Global Optic s)	Data model		[default]				
- • xc16-as	Isolate each functi	on in a section					
• xc16-ld	Remove unused se	ctions					
└── ◎ xc16−ar	Heap size		1024				
	Define common ma	cros	MRBC_NO_TIMER				
	Define C macros			l.			
	Option Descriptio	n Generated Command Line					
	If you select an op	otion its description will appear here	8.				
Manage Configurations							
			OK Cancel Apply	Unlock Help			

図 4.1.12-2 プロジェクトプロパティの設定

複数の項目を入力する場所では、下記のダイアログが開くので、それぞれ入力していく。

(下記は、「C include dirs」をクリックした際のダイアログなので、それぞれ「src」ディレクトリと「hal」ディレクト リを登録する)

Destroy	Down	Up	Browse	
(Enter or 'Brov	vse'string h	nere)		

図 4.1.12-3 プロジェクトプロパティの設定

「Define common macros」の所では、下記のダイアログが開くので、それぞれ入力していく。

Define common macros
Destroy Down Up
MRBC_NO_TIMER
OK Cancel

図 4.1.12-4 プロジェクトプロパティの設定

13.mruby/c プログラムの作成及びコンパイル mruby/c プログラムの作成及びコンパイルを行う。 以下のプログラムを「sample1.rb」として保存する。

```
led = 0
while true
    if led == 1
        led = 0
    else
        led = 1
    end
    if sw1_read() == 0
        led = 1
    end
        led1_write( led )
        sleep 1
end
```

このプログラムを下記の通りコンパイルする。出来た「sample1.c」を、「mrubyc\_Led1」のプロジェクトフォルダーに配置する。

mrbc.exe -E -Bsample1 sample1.rb

次に、ソースファイルを登録する。ヘッダファイルの時と同様にプロジェクトツリーを表示し、「Source Files」の項目で、右クリックし、コンテキストメニューで、「Add Existing Item」を選ぶ。

「Select Item」のダイアログが出るので、「sample1.c」を選び、「Select」ボタンを押し登録する。



図 4.1.13-1 ソースファイルの追加

### 14.「main.c」プログラムの変更

MCC によって「main.c」が自動生成されているので、ボード初期化ルーチンと mruby/c 起動ルーチンを組み込む。

```
·main.c
```

```
#include "mcc_generated_files/system.h"
#include "mcc_generated_files/clock.h"
\texttt{\#include ``mcc\_generated\_files/pin\_manager.h''}
#define FCY _XTAL_FREQ/2
#include "mrubyc.h"
// extern char sample1[];
extern const uint8_t sample1[];
#define MEMORY_SIZE (1024*20)
static uint8_t memory_pool[MEMORY_SIZE];
_____
/*! オンボード SW 現在状態の読み込み
*/
static void c_sw1_read(mrb_vm *vm, mrb_value *v, int argc) {
   int sw1 = I_S4_GetValue();
   SET_INT_RETURN(sw1);
}
/*! オンボード LED ON/OFF
*/
static void c_led1_write(mrb_vm *vm, mrb_value *v, int argc) {
   int set_value = GET_INT_ARG(1);
   if (set_value == 0)
      0_LED3_SetLow();
   else
      0_LED3_SetHigh();
}
//=======
        _____
/*! HAL
*/
int hal_write(int fd, const void *buf, int nbytes) {
   return 0;
}
int hal_flush(int fd) {
   return 0;
}
```

```
/* Main application
*/
int main(void)
{
    // initialize the device
    SYSTEM_Initialize();
    mrbc_init(memory_pool, MEMORY_SIZE);
    mrbc_define_method(0, mrbc_class_object, "sw1_read", c_sw1_read);
    mrbc_define_method(0, mrbc_class_object, "led1_write", c_led1_write);
    mrbc_create_task(sample1, 0);
    mrbc_run();
    return 1;
}
```

#### 15.「hal.h」の修正

ハードウエアレイヤーのルーチンである「hal.h」を修正する。

```
#ifdef __cplusplus
extern "C" {
#endif
#include "../mcc_generated_files/clock.h"
#ifndef FCY
#define FCY
         _XTAL_FREQ/2
#endif
#include <xc.h>
#include <libpic30.h>
#ifndef MRBC_NO_TIMER
# define hal_init()
               ((void)0)
# define hal_enable_irq() ((void)0)
# define hal_disable_irq() ((void)0)
# define hal_idle_cpu()
             ((void)0)
#else // MRBC_NO_TIMER
# define hal_init()
               ((void)0)
# define hal enable irg() ((void)0)
#define hal_disable_irq() ((void)0)
#define hal_idle_cpu()
              ((__delay_ms(1)), mrbc_tick())
```

```
#endif
```

#### 16.「vm\_comfig.h」の変更

PIC は、little endian の16bit alignment の CPU なので、対応出来る様に「vm\_comfig.h」を変更する。

```
/* Hardware dependent flags */
/* Endian
    Define either MRBC_BIG_ENDIAN or MRBC_LITTLE_ENDIAN.
*/
#if !defined(MRBC_BIG_ENDIAN) && !defined(MRBC_LITTLE_ENDIAN)
# define MRBC_LITTLE_ENDIAN
#endif
/* 32bit alignment
    If 32-bit alignment is required, enable the following line.
    */
#define MRBC_REQUIRE_32BIT_ALIGNMENT
```

#### 17. 「alloc.c」の変更

「alloc.c」の「mrbc\_raw\_alloc()」関数で、メモリ確保の下限値丸め込みルーチンを有効化する。

これは、mruby/c で利用しているメモリアロケータの最小メモリアロケーションサイズが16byteであり、32bit 以 上のマシンでは、下限値を下回る事が無い為、無駄なコードとなるのでコメントアウトされている。

// check minimum alloc size. if need.
#if 1 // <- 元は「0」無効であった
if(alloc\_size < (1 << MRBC\_ALLOC\_IGNORE\_LSBS) ) {
 alloc\_size = (1 << MRBC\_ALLOC\_IGNORE\_LSBS);
 }
#else
 assert(alloc\_size >= (1 << MRBC\_ALLOC\_IGNORE\_LSBS) );
#endif</pre>

18.クリーンビルド

「Production」->「Crean and Build Project」を行う。

エラーが出ずにコンパイル出来る事を確認する。



図 4.1.19-1 ビルドメニュー

19.本体への書き込み及び実行

EXPLORER16/32をPCと付属のUSBケーブル(USB-A MicroB)で接続する。



写真 4.1.20-1 EXPLORER16/32 と PC との接続

「Debug」->「Debug Project」を選択する。

City Falts Manufactor Courses Defectors Development	Dala	Tanan Tasla Mindaw Itala	
	E ebi	Debug Project (mrubyc Ledi 210)	
		Discrete Debugger Operation Einish Debugger Session	, Shift+F5
Projects × traise services Classes ■ multiple (Lad) 20 ⊕ mult		Pause Continue Step Oyer Step Into Step Qut	Ctrl+Alt+8 F5 F8 F7 Ctrl+F7
		Step Instruction Run to Cursor Beset Set PC at Cursor Ecous Cursor at PC Stack	F4
	8	Ioggle Line Breakpoint New Breakpoint New Watch New Runtime Watch	Ctrl+F8 Ctrl+Shift+F8 Ctrl+Shift+F9 Ctrl+Shift+F10
		Disconnect from Debug Tool Run Debugger/Programmer Self Test Hardware Tool Emergency Boot Firmware Recovery	

図 4.1.20-1 EXPLORER16/32 書き込み

書き込みが終わるとデバッガ画面となるので、「Continue」ボタンを押すと、ボード上でプログラムが実行され、 LED D3 が緑色に 1 秒おきに点滅する。



図 4.1.20-2 デバッガ画面

LED D3 が緑色に1秒おきに点滅する。



なお、左方の LED D1は、電源ランプであり緑色に点灯していると電源が入っている事を示す。

また、スイッチ S4 を押すと押している間 LED D3 が点灯したままになる事を確認する。

4.2 Chapter02「LED 点滅の速さを変える」を PIC24F の環境で動かす

4.1 で作成したプロジェクトをベースにする。

1.mruby スクリプトの自動コンパイル処理の追加 ・コンパイル用バッチファイル mrbc.bat を mruby ソースコードをおいたディレクトリへコピーする。 バッチファイルは、ITOC 様のページの mruby コンパイラをダウンロードする。 https://www.s-itoc.jp/activity/research/mrubyc/mrubyc\_download/ ・mrbc.bat 中の set MRBC 行を mrbc.exe の絶対パスに書き替える。

・プロジェクトツリーのルート、プロジェクト名の所で右クリックし、プロジェクトのプロパティを選択する。





・「Building」を選び、「Execute This line before build」にチェックをいれ、「mrbc.bat」を登録する。

Ø	Project Properties - mrubyc_Led2_210
Cateories: Concertis: Concertification/Exclusion Conc. (default) Concertification (Exclusion) Concertification (Exclusion) Concertification (Calebal Options) Concertification Concertification (Calebal Options) Concertification (Calebal O	Project Properties - mrubyc_Led2_210     X       Configuration type     applicati       Pre and post step operations: Note commands are run from the project directory (ProjectDir macro below)       Image at this line before build       Image at the project Dir "C:\NDBKY.PIC\artubyc\mrubyc_Led2_210.X"       Configuration type       Execute this line after build       Image bath       "dist\default\\$ [1MGR_TYPE]\mrubyc_Led2_210.X.* (1MGR_TYT_V)       Execute this line after build       Image bath       Device       FIC24F073560B210       ProjectDIr "C:\NDBKY.PIC\artubyc\mrubyc_Led2_210.X.* (1MGR_TYT_V)       Configuration       Macro       Value       Image bath       "dist\default\\$ (1MGR_TYPE)\mrubyc_Led2_210.X.* (1MGR_TYT_V)       Configuration       Options affecting hes file:
Manage Configurations	Insert unprotected checksum in user ID memory
	OK Cancel Apply Unlock Help

図 4.2.1-2 自動コンパイルスクリプト追加

2.浮動小数点の利用

1秒おきのループを、以下の通り0.2秒おきに点滅するように変更する

```
led = 0
while true
    if led == 1
        led = 0
    else
        led = 1
    end
    if sw1_read() == 0
        led = 1
    end
        led1_write( led )
        sleep 0.2
end
```

4.1の18以下の手順通りにコンパイルし実行する。

すると、LED D3 が 0.2 秒間隔で点滅する。(写真 4.1.20-2 LED 点滅 参照)

また、スイッチ S4 を押すと押している間 LED D3 が点灯したままになる事を確認する。

3.タイマーの利用

次にハードウェアタイマーを利用するように設定を変更する。

・MCC にてハードウェアタイマーの TMR1 を設定する。

MCC を立ち上げ、Device Resources 欄から「Timer」->「TMR1」をダブルクリックし、「Project Resources」 に登録する。

Device Resources	۷	-
► <u>_~</u> OC		^
► III PMP		
RTCC		
► 롭 SPI		
🔻 🐧 Timer		
Š TMR1		
S TMR2		
🕚 TMR3		
S TMR4		
₼ TMR5		~

図 4.2.3-1 Device Resources の Timer 項目

Project Resources	Generate	Import	Export	
▼ System				
Interrupt Module				
Pin Module				
System Module				
<ul> <li>Peripherals</li> </ul>				
NTMR1				×

図 4.2.3-2 Project Resouces に TMR1 を登録

TMR1の設定を行う。

TMR1 を1ms毎に実行するので、「Tmer Period」欄に「1ms」を入力し、「Enable Timer interrupt」を有効化し、「Generate」する。

TMR1				۲
💮 Easy Setup 📃 Reg	isters			
Hardware Settings				
Enable TMR			Enable Gate	
Timer Clock			Timer Period	
Clock Source	FOSC/2	-	Period Count 0x0 ≤ 0x7D0 ≤ 0xFFFF	
Input Frequency	2 MHz		Timer Period 20 ns ≤ 1 ms 32.7675 ms	
Prescaler	1:1	-	Calculated Period 1 ms	
Enable Timer Inter	rupt			
Softwar settings				
Callback Function Rate	e 0x1	×	×Timer Period = 1 ms	

図 4.2.3-2 TMR1 の設定

4. 「hal.h」の修正

ハードウエアレイヤーのルーチンである「hal.h」を修正する。

割り込み許可、禁止、cpu休止命令を追加する。

#ifndef MRBC\_NO\_TIMER
#define hal\_init() ((void)0)
#define hal\_enable\_irq() \_\_builtin\_disi(0x0000)
#define hal\_disable\_irq() \_\_builtin\_disi(0x3FFF)
#define hal\_idle\_cpu() Idle()

5.「main.c」に、タイマー割り込みハンドラとスタートを設定する。

```
/*
                             Main application
*/
int main(void)
{
    // initialize the device
    SYSTEM_Initialize();
    mrbc_init(memory_pool, MEMORY_SIZE);
    mrbc_define_method(0, mrbc_class_object, "sw1_read", c_sw1_read);
mrbc_define_method(0, mrbc_class_object, "led1_write", c_led1_write);
    mrbc_create_task(sample1, 0);
    TMR1_Start(); // <- added</pre>
    mrbc_run();
    return 1;
}
 * TIMER CallBack
*/
                                   // <- added
void TMR1_CallBack(void) {
    mrbc_tick();
                                     // <- added
                                     // <- added
}
```

6.プロジェクトプロパティの設定

プロジェクトプロパティの設定を設定する。設定内容一覧は下記である。

4.1 との違いは、ハードウェアタイマーを使う為、「MRBC\_NO\_TIMER」宣言をしないことである。

設定場所1	設定場所2	設定項目	内容
XC16	Most Useful Options	C include dirs	src,src¥hal
xc16-ld	General	Heap size	1024
xc16-ld	General	Min stack size	1024

表 4.2.6-1 プロジェクトプロパティの設定

プロジェクトツリーのルート、プロジェクト名の所で右クリックし、プロジェクトのプロパティを選択する。

以前のプロジェクトを引き継いでいる場合は、設定が記述済みである。



図 4.2.6-1 プロジェクトプロパティの設定

「Project Propeties」ダイアログが開くので、左ペインで「設定場所1」を選択(ここでは「XC16」)し、右ペインの「Option categories」で「設定場所2」を選択(ここでは、「Most Useful Options」)を選び、「設定項目」の右側の欄をクリックすると数字入力や、パラメータ入力が出来る。

3	Project Properties - mruby	_Led1
Categories:	Options for xc16-gcc (v1.35)	
File Inclusion/Exclusion	Option categories: Most Useful Options	✓ Reset
Starter Kit (PKOB)	Optimization level	0
- O Loading	Common include dirs	
- • Libraries	C include dirs	src;src¥hal
	Code model	[default]
<ul> <li>XC16 (Global Optic s)</li> </ul>	Data model	[default]
- • xc16-as	Isolate each function in a section	
• xc16-ld	Remove unused sections	
└── © xc16−ar	Heap size	1024
	Define common macros	MRBC_NO_TIMER
	Define C macros	
_	Option Description Generated Command Line	
	If you select an option its description will appe	ar here.
Manage Configurations		
		OK Cancel Annhy Unlock Hele

図 4.2.6-2 プロジェクトプロパティの設定

複数の項目を入力する場所では、下記のダイアログが開くので、それぞれ入力していく。

(下記は、「C include dirs」をクリックした際のダイアログなので、それぞれ「src」ディレクトリと「hal」ディレクト リを登録する)

Common include dirs	×
Destroy Down Up Browse	
(Enter or 'Browse' string here)	
Relative paths are from OK Cance	1

図 4.2.6-3 プロジェクトプロパティの設定

「Define common macros」の所では、以前のプロジェクトを引き継いでいる場合には、下記のように「MRBC\_NO\_TIMER」が入力済みであるので、削除する。

S Define common macros	×
Destroy Down Up	
MRBC_NO_TIMER	
OK Cancel	

図 4.2.6-4 プロジェクトプロパティの設定

7.クリーンビルトと実行

4.1 の 18 以下の手順通りにコンパイルし実行する。

すると、LED D3 が 0.2 秒間隔で点滅する。(写真 4.1.20-2 LED 点滅 参照)

また、スイッチ S4 を押すと押している間 LED D3 が点灯したままになる事を確認する

4.3 Chapter03「複数の mruby プログラムを同時に動かす」を PIC24F の環境で動かす

複数の mruby プログラムを同時に動かします。

1.mruby プログラムの用意とコンパイル 下記の mruby プログラム「sample1.rb」「sample2.rb」を用意する。 コンパイルは、4.2 で準備した自動コンパイル処理にて行われる。

「sample1.rb」

```
led = 0
while true
    if $sw1 == 1
        led = 0
    else
        led = 1
    end
    if sw1_read() == 0
        led = 1
    end
        led1_write( led )
        sleep 0.2
end
```

「sample2.rb」

```
while true
   $sw1 == sw1_read()
end
```

2.ビルド及び実行を行う。

4.1の18以下の手順通りにコンパイルし実行する。

すると、LED D3 が 0.2 秒間隔で点滅する。(写真 4.1.20-2 LED 点滅 参照)

また、スイッチ S4 を押すと押している間 LED D3 が点灯したままになる事を確認する。

3.コンソールを有効にする

UART を使ったコンソールを有効化する。

・MCC にてハードウェア UART の UART1 を設定する。

MCC を立ち上げ、Device Resources 欄から「UART」->「UART1[Foundation Services Library by Microchip Technology,Inc]」をダブルクリックし、「Project Resources」に登録する。



図 4.3.3-1 Device Resources の UART 項目

Project Resources	Generate	Import	Export		
▼ System					
Interrupt Module					
Pin Module					
System Module					
<ul> <li>Peripherals</li> </ul>					
🕚 TMR1					×
📄 UART1 [Foundati	on Services	Library by M	licrochip	Technology,	I 🗙
<u> ۲</u>					

図 4.3.3-2 Project Resouces に UART1 を登録

### UART1の設定を行う。

UART1 を Hardware Settings は初期値で、ピンアサインは、「U1RX:49」「U1TX:50」に設定し、「Generate」 する。

UART1							
🔯 Easy Setup 📃 Registers							
Hardware Setting	5						
Enable UART							
Baud Rate	9600		*	Error Rate = 0.160			
Parity	None		•				
Data Bits	8		*	]			
Stop Bits	1		*	]			
Flow Control	None		-				
Enable UART	Enable UART Interrupts						
Software Settings							
Redirect Print	tf to UART						
Software Transm	it Buffer Size	0×8		Bytes			
Software Receive Buffer Size 0x8			Bytes				

図 4.3.3-3 UART1 設定

		····						,		
Package:	TQFP100	-		Pin No:		49	50	53	40	
						ort F	▼			
Module	Fund	ction		Direction		4	5	8	12	
	U1CTS	U1CTS		input		î.	î.	î.	Ъ	
	U1RTS	U1RTS		output		2	3	î.		
UARTI V	U1RX	U1RX		input		â	î	Ъ	î.	
	U1TX	U1TX		output		ì	ô	14		
2										1

図 4.3.3-4 UART1 ピンアサイン設定

4. 「main.c]プログラムの変更

下記の通り、hal\_write()を実装する。

```
/*! HAL (別途説明します)
*/
int hal_write(int fd, const void *buf, int nbytes) {
    int i;
    while (U1STAbits.TRMT == 0);
    for (i = nbytes; i; --i) {
        while (U1STAbits.TRMT == 0);
        U1TXREG = *(char*) buf++;
    }
    return (nbytes);
}
```

\_\_\_\_\_

5.mruby/c プログラムの修正

「sample1.rb」を下記の通り修正する

6. ビルド及び実行を行う。

4.1 の 19 の手順通りにコンパイルし実行する。

すると、LED D3 が 0.2 秒間隔で点滅する。(写真 4.1.20-2 LED 点滅 参照)

さらにシリアルコンソールに LED D3 点灯時1消灯時 0 が出力される。

なお、文字区切りコードは、LF なので、注意が必要である。

また、確認の際には、EXPLORER16/32 に搭載されているシリアル・USB 変換の ICMicrochip 社製の MCP2221A を利用する。

ドライバーは、下記からダウンロードできる。

https://www.microchip.com/wwwproducts/en/MCP2221A

から、下方の Tab の「Doccuments」内「Software」より、適宜ダウンロードする。

windows ドライバー直接ダウンロード(URL が変わる可能性が有ります)

http://ww1.microchip.com/downloads/en/DeviceDoc/MCP2221%20Windows%20Driver%202014-10-09.zip



図 4.3.6-1 ターミナル画面



写真 4.3.6-1 EXPLORER16/32とPCとの接続

また、スイッチ S4 を押すと押している間 LED D3 が点灯したままになる事を確認する。 その際には、シリアルコンソールには、LED D3 点灯中なので1が出力される。

# 5.、メモリ消費量について

mruby/c は、メモリ必要量が少ない事が特徴の 1 つであるが、CPU のビット数によって変化するかを検証した。 本レポートで使用している PIC 16bit 環境、比較対象として、intel 64bit 環境、OPENRISC 32bit 環境を比較 する事とし、メモリ消費量としては、mrbc\_init()で確保した memory\_pool の RAM 使用量で比較する。 ruby のコードは、mruby/c の「sample1.rb」を利用した。

CPU ビット数	Intel x64 64bit OS FreeBSD	OPENRISC 32bit OS なし	PIC24F 16bit 0S なし	
RAM 消費量	12, 952	8, 416	5, 304	
16bitの消費量を1とした比	2.4	1.6	1	

表 5-1 CPU ビット数と RAM 消費量

計測の結果、CPU の bit 数が小さくなるほど消費するメモリ量が削減される事が分かった。

理由としては、ポインタのサイズが、それぞれ CPU のビット数で異なる事が原因と思われる。

また、この事は、同じ RAM の量でも16bitCPUの方が 32bitCPU より 1.6 倍 RAM が使える事を示しており、より大きなプログラムを動かせる可能性を示すといえる。

また、PIC24Fシリーズ(dsPIC33 含む)には、427 品種あるが、RAM が 8kByte を超えるのは324品種あるので、PIC24F の半数以上の品種で mruby/c が動作可能といえそうである。

ただし、8Kbyre の品種では、利用可能な RAM は 2kByte を切るため、どの程度のコードが動作可能かは十分検証する必要が有る。

実用的には16kByte以上のメモリが必要となると思われるので、199品種に減る事となる。

また、PIC24Fのデータメモリエリアは 32kByte 毎のバンク切り替えとなっている為、32kByte 以上のエリアを アクセスする配列を作るには、「\_\_eds\_\_」型修飾子を付ける必要が有るので注意する必要が有る。

なお、現状の mruby/c コードでは、vm 立ち上げ時に一時的にではあるが、スタックに多量のメモリを確保するのでその点を改良するとさらに使用できるメモリを増やす事が出来ると思われる。

# 6.まとめ

Microchip 社製の PIC24F の16bitの環境にて mruby/c のチュートリアルの動作が確認出来た。 CPUの bit 数が少ないほどメモリ消費量が少なくて済む傾向がわかった。

mruby/cのPIC24へのポーティング調査報告書作成

調査報告書作成 株式会社バイタルリード 片岡 晃 調査報告書監修 しまねソフト研究開発センター 東 裕人