

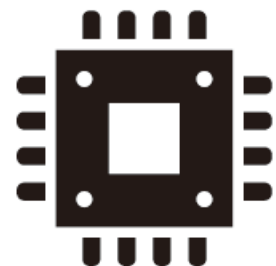
# C 言語ではじめる Raspberry Pi 徹底入門



## Chapter 14

### シリアルポートを使う

ラズパイや Arduino 等には UART と呼ばれるシリアル通信のためのハードウェア回路を内蔵しています。UART の送信と受信の端子をシリアルポートと呼びます。本章では、UART のデータフォーマットと RS-232 規格について解説し、シリアル通信のプログラミングについて学びます。UART をインタフェースとするデバイスや RS-232 装備した計測器や産業用機器は数多くあります。シリアルポートを習得することで、ラズパイの活用の可能性が広がることでしょう。



# 目次

<b>Chapter14</b>	<b>シリアルポートを使う</b>	<b>1</b>
はじめに		4
14.1	本章で準備するもの	8
14.2	シリアルポートとは	9
	非同期通信のデータフォーマット/通信速度の単位 bps	
14.3	RS-232 とは	13
	信号線と機能/コネクタのピン配置と結線の種類/電気的特性	
14.4	ラズパイのシリアルポートの設定と構成	17
	シリアルポートの設定/mini UART と PLO11 の特徴	
14.5	RS-232 インタフェース回路	20
	回路図/配線図	
14.6	PC のシリアルポートの通信設定	23
	COM 番号の確認方法/ターミナルソフト Tera Term の設定	
14.7	Linux のコンソールからログインする	28
	RS-232 ケーブルで接続する/PC からログインする/コンソールでプログラムを作成し実行する	
14.8	ラズパイと PC でテレタイプする	32
	Serial Console を無効にする/minicom のインストール/テレタイプの実験	
14.9	WiringPi Serial Library 関連	35
	wiringSerial.h ヘッダファイル/シリアルポートをオープンする serialOpen 関数/シリアルポートをクローズする serialClose 関数/1 バイトを送信する serialPutchar 関数/バッファをクリアする serialFlush 関数/1 バイトを受信する serialGetchar 関数/受信をチェックする serialDataAvail 関数/null 付き文字列を送信する serialPuts 関数/文字列を送信する serialPrintf 関数/通信パラメータを設定する termios 構造体	
14.10	Geany のビルドの設定	39
14.11	mini UART でデータ伝送する	40
	データを送信する/送信のフローチャート/送信のソースコード/送信の動作確認/データを受信する/受信のフローチャート/受信のソースコード/受信の動作確認	
14.12	パリティビットを付加してデータ伝送する	48
	mini UART と PLO11 の接続先を交換する/データを送信する/送信のソースコード/送	

信の動作確認/データを受信する/受信のソースコード/受信の動作確認	
14.13 章末問題	55
参考文献	61

#### 必ずお読みください

本書では、次のようなシステム構成を前提に解説しています。

- 本体 : Raspberry Pi 3/3B+および4B
- OS : Raspbian (Buster) 2019-09-30 版 (NOOBS Ver.3.2.1)
- ディスプレイ : HDMI仕様
- キーボード・マウス : USB仕様
- SDカード : microSD SDHC 16GB スピードクラス10
- ネットワーク環境 : Raspberry Piに内蔵した無線LAN (Wi-Fi) でインターネットに接続

本書は、Windows PCやインターネットの一般的な操作を一通りできる方を対象にしているため、基本的な操作方法などは解説していません。C言語や電子回路の基本的な知識についても、習得されていることを前提にしています。電子部品や工具などを使用する際は、ケガなどをされないようにご注意ください。また、電子部品は静電気に弱いので取り扱いにもご注意ください。

#### 本書のサポートページについて

本書で利用するソースコードなどが記述されたサンプルファイルと追加情報や、正誤表、本書発行後にバージョンアップなどにより設定手順などに変更が生じた場合の補足情報は、本書サポートページに掲載しています。

- 本書サポートページ  
<https://gihyo.jp/book/2020/978-4-297-11299-8>

本書に記載された内容は、情報の提供のみを目的としています。したがって、本書を用いたご利用は、必ずお客様自身の責任と判断によって行ってください。これらの情報によるご利用の結果について、技術評論社および著者はいかなる責任も負いません。

本書記載の情報は、2020年1月現在のものを掲載していますので、ご利用時には変更されている場合もあります。また、ソフトウェアに関する記述は、とくに断わりのないかぎり、2020年1月時点での最新バージョンをもとにしています。それぞれバージョンアップされる場合があり、本書での説明とは内容などが異なってしまうこともあり得ます。本書ご購入の前に、必ずバージョン番号などをご確認ください。

本書の配線図は「Fritzing」(<https://fritzing.org/>)を使って描いたものを使用しています。

以上の注意事項をご承諾いただいたうえで、本書をご利用願います。これらの注意事項をお読みいただかずに、お問い合わせいただいても、技術評論社および著者は対処しかねます。あらかじめ、ご承知おください。

本書で記載されている会社名、団体名、製品名、サービス名などは、一般に関係各社・団体の商標または登録商標です。なお、本書では™、®などは明記していません。また、本書では、通称やそのほかの名称で記述する場合がありますが、あらかじめご了承ください。

## はじめに

「Chapter14 シリアルポートを使う」は、「C 言語ではじめる Raspberry Pi 徹底入門」（以下、「徹底入門」）の追加の章として作成しました。本章では、Chapter13 までに学習した内容をベースに、シリアル通信の仕組みや応用事例を紹介することを目的にしています。そのため、本章では必要に応じて Chapter13 までに解説した電子回路およびソースコード等の技術的な内容を引用しています。理由は、前章までに学習した内容やコードを引用することは、本章で学ぶ新しい内容との比較により、読者にとって理解し易いと考えたからです。また、紙面の都合上、前章までに解説した学習内容については割愛しています。読み飛ばされている読者は、必要に応じて該当章を参照してください。

本章では、本書で前提にしているラズパイのシステム構成において例題等で必要とする範囲で情報を提供しています。さらに詳しく勉強されたい方は、man ページや本章の参考文献等を参照してください。

- 本章のソースコードは、著者の GitHub の Web サイトからダウンロードできます。

- <https://tamamori2020.github.io/raspi-gh1/>

なお、「徹底入門」で作成したライブラリ・ファイル (libMyPi.a, MyPi.h) については、出版社の Web サイトにアップされている MyApp.zip に含まれています。

- <https://gihyo.jp/book/2020/978-4-297-11299-8/support>

- 「徹底入門」出版後に、追加した内容については著者のサポートページから資料をダウンロードできます。是非、ご参照ください。

- 章末問題 Chapter4 から Chapter9
- Chapter13 7 セグメント LED 表示器で数字で表示する
- <http://raspi-gh1.blogspot.com/>



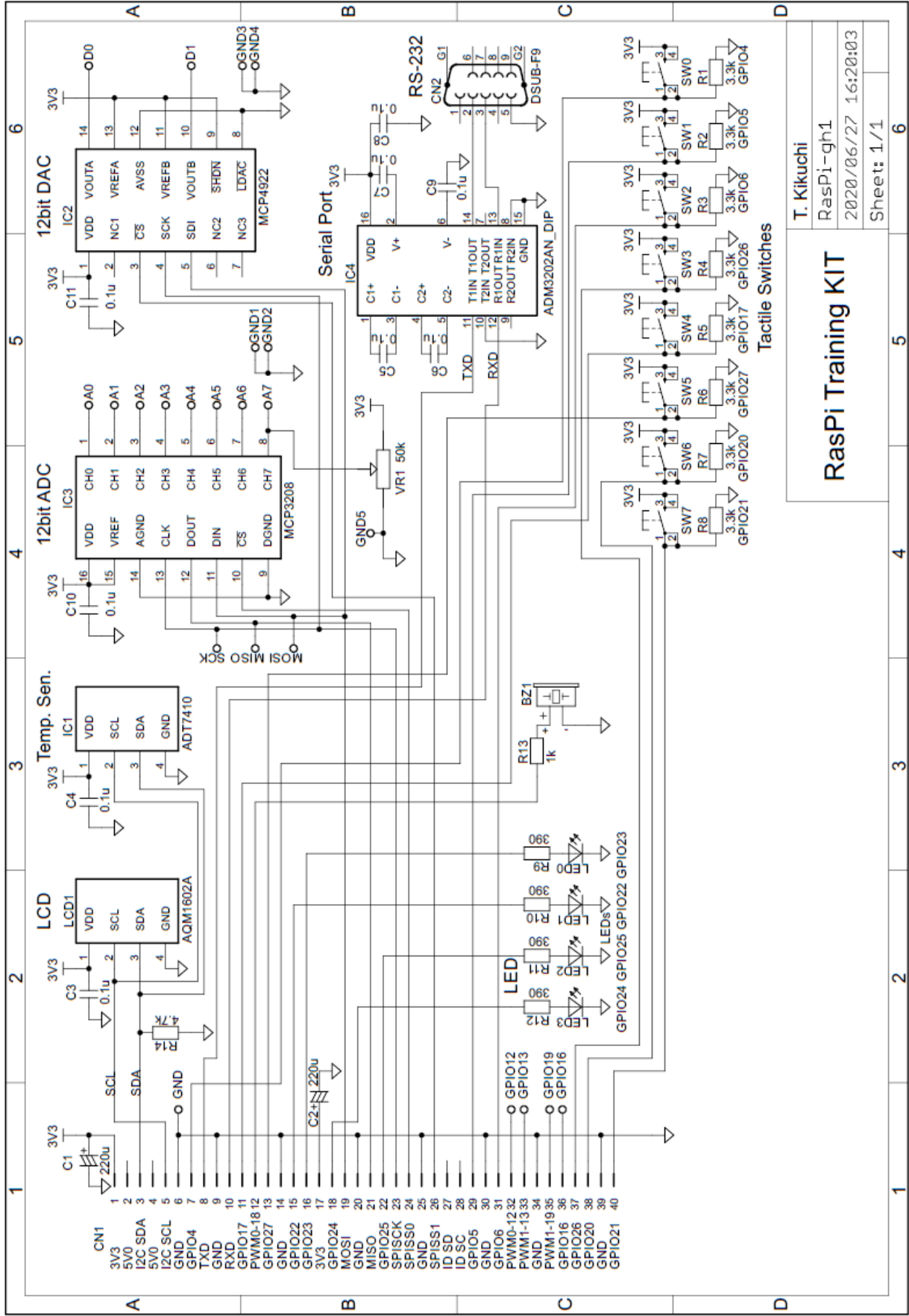
- 本章のソースコードと「徹底入門」のライブラリは、「徹底入門」の Chapter3 で作成したファイル保存用のフォルダ「MyApp」に保存して、使用してください。

/home/pi/MyApp

- ソースコードのコーディングに必要なハードウェアの情報は、「電子部品と GPIO の対応表」と回路図を参照してください。また、詳細な配線図については、表の備考に示した該当章を参照してください。

● 電子部品と GPIO の対応表

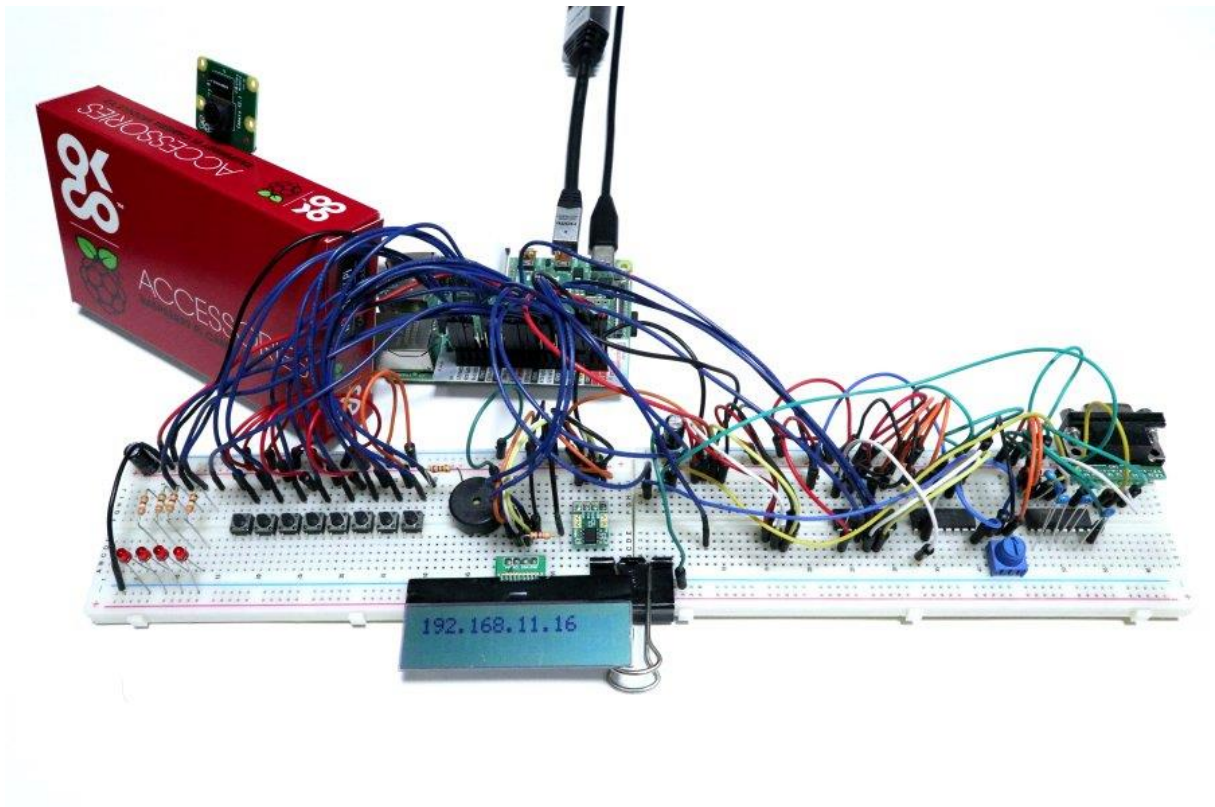
電子部品		GPIO	端子	方向	備 考
LED	LED0	GPIO <del>23</del>	16	出力	Chapter4 出力=HIGH→点灯 出力=LOW →消灯
	LED1	GPIO <del>22</del>	15		
	LED2	GPIO <del>25</del>	22		
	LED3	GPIO <del>24</del>	18		
タクトイル スイッチ	SW0	GPIO <del>4</del>	7	入力	Chapter5 押す→入力=HIGH 離す→入力=LOW
	SW1	GPIO <del>5</del>	29		
	SW2	GPIO <del>6</del>	31		
	SW3	GPIO <del>26</del>	37		
	SW4	GPIO <del>17</del>	11		
	SW5	GPIO <del>27</del>	13		
	SW6	GPIO <del>20</del>	38		
	SW7	GPIO <del>21</del>	40		
圧電サウナダ		GPIO <del>18</del> /PWM	12	出力	Chapter6 章末問題 Chapter6 から Chapter9
LCD モジュール AQM1602	GPIO <del>3</del> /SCL		5	入出力	Chapter7 I <sup>2</sup> C バス アドレス 0x3e 表示 16 文字×2 行
	GPIO <del>2</del> /SDA		3		
温度センサ ADT7410	GPIO <del>3</del> /SCL		5	入出力	Chapter7 I <sup>2</sup> C バス アドレス 0x48 測定範囲 -40℃～+105℃
	GPIO <del>2</del> /SDA		3		
D/A コンバータ MCP4922	GPIO <del>11</del> /SCK		23	出力	Chapter8 SPI バス 出力電圧範囲 0V～3.3V
	GPIO <del>10</del> /MOSI		19	出力	
	GPIO <del>7</del> /SS1		26	出力	
A/D コンバータ MCP3208	GPIO <del>11</del> /SCK		23	出力	Chapter8 SPI バス 入力電圧範囲 0V～3.3V
	GPIO <del>9</del> /MISO		21	入力	
	GPIO <del>10</del> /MOSI		19	出力	
	GPIO <del>8</del> /SS0		24	出力	
シリアルポート ADM3202ANZ	GPIO <del>14</del> /TXD		8	出力	Chapter14 通信速度 115,200bps
	GPIO <del>15</del> /RXD		10	入力	



T. Kikuchi	
RasPi-gh1	
2020/06/27 16:20:03	
Sheet: 1/1	6

# RasPi Training KIT

- 配線の写真



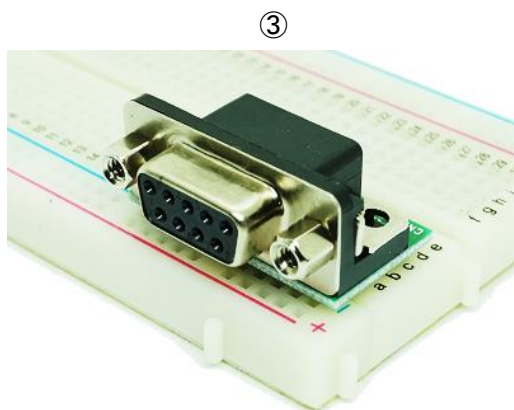
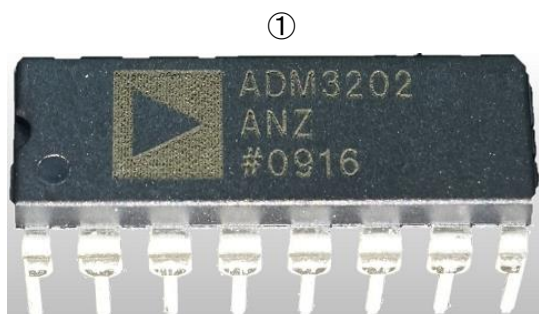


## 14.1. 本章で準備するもの

本章の電子工作で新たに必要とする主なものを表 14-1 に示します。参考品の欄に初心者の方がイメージしやすいように写真と入手情報等を載せましたが、持ち合わせの部品や相当品で結構です。なお、「徹底入門」で準備したものも使用します。

表 14-1 本章で準備する主なもの

名 称	個数	備 考	参 考 品 (秋月電子通商)
RS-232 インタフェース IC	1	アナログ・デバイゼズ社の ADM3202 を使用。ラズパイのシリアルポートの 3V 系電圧と RS-232 仕様の電圧を相互に変換します。	I-09001 (①)
積層セラミックコンデンサ 0.1μF	5	ADM3202ANZ に使用します。	P-10147 (②)
D サブコネクタ(メス)DIP 化キット	1	D サブコネクタの 2 列千鳥ピンを 1 列の 2.54 mm ピッチに変換します。はんだ付け作業があります。	K-05406 (③)
USB シリアル変換ケーブル	1	PC の USB ポートを RS-232 インタフェースに変換するケーブルです。	M-08343 (④)
RS-232 延長ケーブル(ストレート結線)	1	USB シリアル変換ケーブルの長さが足りない場合に使用します。	C-00004





## 14.2. シリアルポートとは

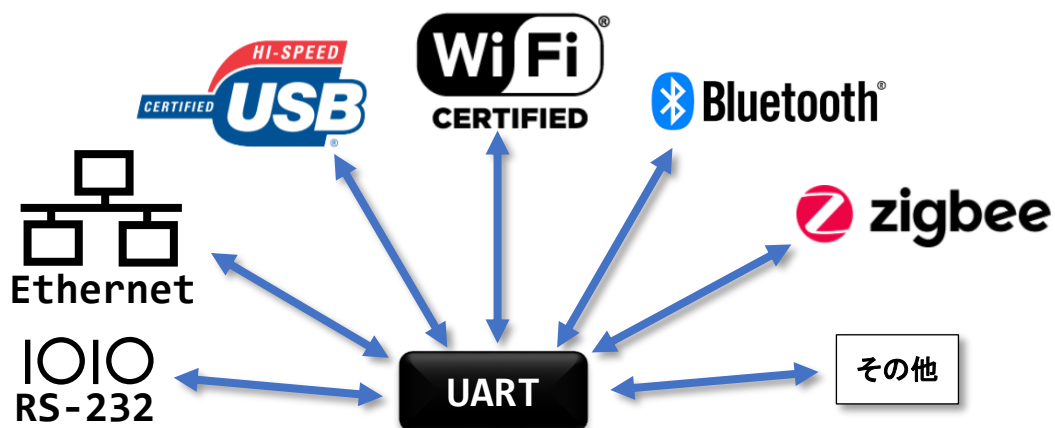
シリアルポートという用語は、かつてパソコン（PC）に標準的に装備されていた RS-232 規格に準拠したシリアル通信の接続端子を指します<sup>1</sup>。PC 内部のデータバスはパラレル信号ですが、外部との通信には経済性や利便性の理由から信号線を少なくしたシリアル通信が使用されます。そのため、パラレル信号とシリアル信号を相互に変換する回路が必要になります。その回路（デバイス）が、UART (Universal Asynchronous Receiver-Transmitter：汎用非同期送受信回路) です。UART は「ユーエーアールティ」または「ユーアート」と呼ばれています。

非同期とは**非同期通信**のことです。非同期通信では「始まり」と「終わり」を表す bit をデータに挟んで通信します。非同期は「調歩同期」とも呼ばれています。なお、シリアル通信には、同期 (Synchronous) 通信もあります。同期通信はクロック信号に同期してデータ信号を送受信する方式です。

UART は PC のみならず、ラズパイや Arduino Uno 等のマイコンでも UART を内蔵し、マイコン同士やセンサなどのデバイスとのシリアル通信に利用できます。また、UART と専用のデバイスを組み合わせることにより、RS-232、USB、イーサネット、Wi-Fi、Bluetooth、ZigBee などの様々なシリアル・インタフェースの使用が可能になります(図 14-1)。UART を活用することで、組込み機器の様々な可能性が広がります。実際、ラズパイの Bluetooth はシステムオンチップ (SoC) の UART を使用しています。

本書では、ラズパイの UART の送受信端子をシリアルポートと呼びます。以降の節では、UART の非同期通信のデータフォーマット、RS-232 規格の概要、ラズパイのシリアルポートの設定方法、RS-232 インタフェース回路の設計、Linux コンソールを使用したログイン、PC とのデータ伝送のプログラミングについて学びます。

図 14-1 各種シリアル・インタフェース

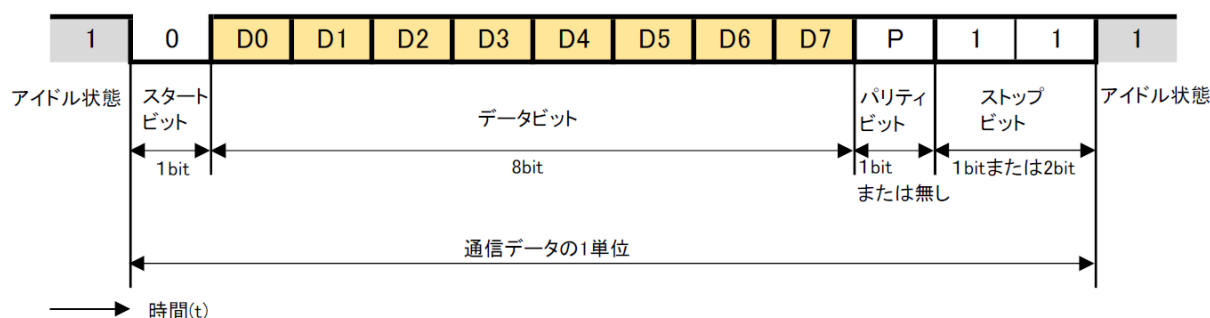


<sup>1</sup> 今日の PC のシリアル通信の機能は USB、イーサネット、Wi-Fi 等に置き換わりました。

## 14.2.1. 非同期通信のデータフォーマット

一般的な非同期通信のデータフォーマットを図 14-2 に示します。通信データの 1bit 目は、スタートビットから始まり、送信と受信のデータビット、パリティビット、ストップビットで構成されます。

図 14-2 非同期通信のデータフォーマット



### ● アイドル状態 (idle condition)

アイドル状態とは通信データが無い状態のことです。アイドル状態のときは「1」で、電圧は HIGH です。

### ● スタートビット (start bit)

通信データの先頭ビットを「スタートビット」と呼びます。1bit 分の「0」に定義され、電圧は LOW です。

### ● データビット (data bits)

データビットは送信と受信のデータです。データの LSB (最下位ビット) が先頭になります。データがアスキーコードの英数字記号だけなら 7bit、1 バイトのデータの場合は 8bit を指定します。

### ● パリティビット (parity bit)

parity とは奇偶のことで、パリティチェックは通信上の誤りを検出する方法の一つとして使用され、検査結果がパリティビットに入ります。パリティビットの設定には、「偶数 (even) パリティ」、「奇数 (odd) パリティ」、「パリティ無効」の 3 種類あります。

パリティビットは、データビットのすべてのビットを EXOR (EXclusive OR: 排他的論理和) の計算から求める 1bit の誤り検出符号です。参考に、2 入力の EXOR の論理式を式 14-1 に、真理値表を表 14-2 に示します<sup>2</sup>。

$$Y = A \oplus B = \bar{A} \cdot B + A \cdot \bar{B} \quad \text{式 14-1}$$

<sup>2</sup> EXOR は不一致論理とも呼びます。

表 14-2 2入力 EXOR の真理値表

入力		出力
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

パリティビットを付けて送信すると、受信側の端末でも受信したデータビットからパリティビットを計算します。送信側のパリティビットと受信側のパリティビットを照会して、一致しなければ、通信の過程で何らかの原因（ノイズ等）によりデータビットに誤りが発生したと推察できます。その場合、送信側にデータビットの再送を指示するなどのエラー処理が可能になります。

パリティビットの計算方法にはデータビットの「1」の数が偶数か、奇数かにより、偶数パリティ（式 14-2）と奇数パリティ（式 14-3）の計算式があります。偶数パリティでは、0を含めて EXOR の計算します。奇数パリティでは、1を含めます。

#### 偶数パリティ( $P_{\text{even}}$ )

$$P_{\text{even}} = D7 \oplus D6 \oplus D5 \oplus D4 \oplus D3 \oplus D2 \oplus D1 \oplus D0 \oplus 0 \quad \text{式 14-2}$$

偶数パリティはデータビットの「1」の個数が偶数なら「0」、奇数なら「1」になります。

【例】データビットがアスキーコード 'A' の 0 1 0 0 0 0 0 1 のとき、D7 から順番に EXOR を計算すると  $P_{\text{even}}$  は 0 と求まります。

$$P_{\text{even}} = 0 \oplus 1 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 1 \oplus 0$$

$$= 0$$

【例】データビットがアスキーコード 'C' の 0 1 0 0 0 0 1 1 のとき、D7 から順番に EXOR を計算すると  $P_{\text{even}}$  は 1 と求まります。

$$P_{\text{even}} = 0 \oplus 1 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 1 \oplus 1 \oplus 0$$

$$= 1$$

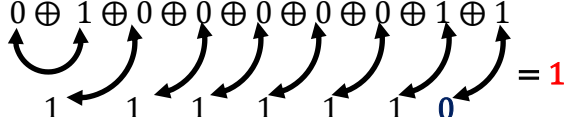
例えば、アスキーコード 'A' を送信して、伝送途中でノイズ等によりデータビットの D1 が 1 になって、受信側では 'C' を受信したとします。送れた通信データのパリティビットは、A' のパリティビット ( $P_{\text{even}}=0$ ) ですから、受信した 'C' から計算したパリティビット ( $P_{\text{even}}=1$ ) が異なるため、データビットに誤りが生じたと判断できます。

## 奇数パリティ(P<sub>odd</sub>)

$$P_{\text{odd}} = D7 \oplus D6 \oplus D5 \oplus D4 \oplus D3 \oplus D2 \oplus D1 \oplus D0 \oplus 1 \quad \text{式 14-3}$$

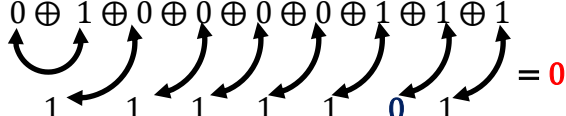
奇数パリティはデータビットの「1」の個数が奇数なら「0」、偶数なら「1」になります。

【例】データビットがアスキーコード 'A' の 0 1 0 0 0 0 0 1 のとき、D7 から順番に EXOR を計算すると P<sub>odd</sub> は 1 と求まります。

$$P_{\text{odd}} = 0 \oplus 1 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 1 \oplus 1$$


= 1

【例】データビットがアスキーコード 'C' の 0 1 0 0 0 0 1 1 のとき、D7 から順番に EXOR を計算すると P<sub>odd</sub> は 0 と求まります。

$$P_{\text{odd}} = 0 \oplus 1 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 1 \oplus 1 \oplus 1$$


= 0

なお、パリティチェックは偶数個のビットに誤りが発生した場合は誤りを検出できない欠点がありますが、パリティビットの情報は最小（経済的）で、高速に処理できるためデータをやり取り取りする様々な場面で利用されています。

## ● ストップビット(stop bit)

通信データの終了を示すビットを「ストップビット」と呼びます。「1」に定義されています。ビット幅は 1bit または 2bit を指定できます。

## 14.2.2. 通信速度の単位 bps

通信速度は 1 秒間に伝送できるビット数を定義します。単位はビット毎秒で **bps** (bits per second) です。通信速度の範囲は各 UART の仕様で異なります。

通信速度に大きな値を扱う場合は、「徹底入門」p. 77 に示した SI 接頭辞を使用します。1000 bps は 1kbps (キロ bps)、1000000bps は 1Mbps (メガ bps)、1000000000bps は 1Gbps (ギガ bps) になります。

ボーレート(baud rate)も通信速度の単位として使用する事例がありますが、ボーレートは「1 秒間に変復調できる値」を定義しています。電話回線(アナログ)でデジタルデータを伝送するモデムの性能を表す単位に使用されていました。1 回の変調で 1bit を送る場合はボーレートと bps の値は同じですが、1 回の変調で複数 bit を送る多値変調では値は異なります。そのため、通信速度の単位には bps が使用されます。

## 14.3. RS-232 とは

インターネットが普及する前、電話回線（アナログ）を利用したパソコン通信が行われていました（図 14-3）。パソコン通信では、電話回線のアナログ信号とパソコン(PC)のデジタル信号を相互に変換するモデムと呼ばれる装置が使われていました。モデムと PC とのインタフェースが RS-232 です[1]。

RS-232(Recommended Standard 232)は米国電子工業会（EIA : Electronic Industries Alliance）が 1960 年に策定した通信規格で「EIA-232」とも呼ばれます。規格の名称は「Interface Between Data Terminal Equipment and Data Circuit- Terminating Equipment Employing Serial Binary Data Interchange」です。テレタイプライタ、PC などの Data Terminal Equipment(DTE)と、モデムなどの Data Circuit- Terminating Equipment(DCE)を接続してデータ伝送するための電氣的・機械的な仕様を定義しています。

RS-232-C の名称を製品の仕様に見かけますが、「C」は改訂の版のことです。現在、RS-232 規格は米国電気通信工業会（TIA : Telecommunications Industry Association）に引き継がれ、最新規格は「TIA-232-F」です[2]。

RS-232 の規格では 25 ピンのコネクタで同期通信と非同期通信を定義しています。1984 年に IBM が発売した PC/AT では 9 ピンのコネクタで非同期通信のみをサポートした RS-232 のサブセットの仕様のシリアルポートを装備していました。その後、PC/AT 互換機の普及に伴いサブセット仕様がデファクトスタンダードとなり、混同を避ける目的で別規格として TIA-574 を制定しました。しかし、業界やユーザーは TIA-574 を引き続き RS-232 と呼んでいる背景から、本書でも RS-232 としています。

RS-232 はモデムだけでなく、計測器、PLC(Programmable Logic Controller)などの制御機器、産業ロボットなどの機械に装備されて、コンピュータとのシリアル通信に使用されてきました。現在、USB やイーサネットが普及していますが、RS-232 の通信方式はシンプルのため扱いやすく、比較的ノイズにも強いなどの理由から、産業界では使われ続けています。

RS-232 規格の概要を表 14-3 に示します[3]。最大伝送速度は 20kbps ですが、高性能な UART とドライバ/レシーバ IC を使用することで、さらに高速な通信が可能です。なお、RS-232 を改良した規格に RS-422 と RS-485 があります。これらの規格では、最大伝送速度は 10Mbps、最大ケーブル長は 1,200m になります。

図 14-3 電話回線(アナログ)を利用したパソコン通信

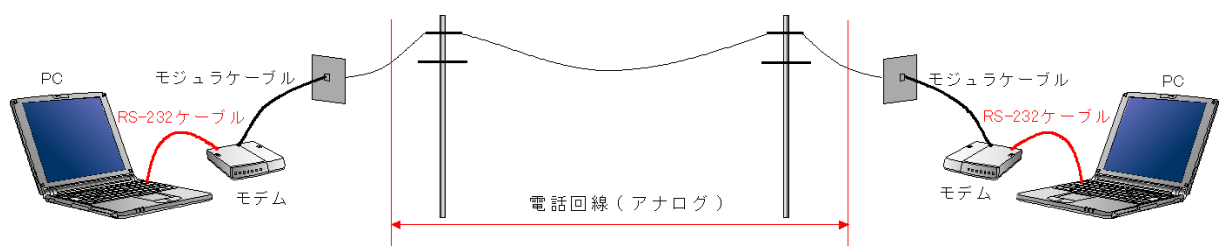


表 14-3 RS-232 の概要

項 目	内 容
通信モード	全二重(データの送信と受信を同時に行える)
接続の形態	1:1接続
最大伝送速度	20kbps
最大ケーブル長	約 15m(50ft)

### 14.3.1. 信号線と機能

9 ピンコネクタの RS-232 の信号名を表 14-4 に示します[4]。データを受信する信号は RXD(受信データ)で、送信する信号は TXD(送信データ)です。RS-232 は PC とモデム間で通信することを目的として策定されたため、送信信号と受信信号の他にモデムを制御する信号線があります。これらの信号は、送信側と受信側で送信の許可やデータ受信の完了など確認するための制御信号として使用されます。

これらの制御信号を使用する方式をハードウェアフロー制御と呼びます。また、制御信号を使用しないで、データの中にアスキーコード制御文字の XOFF (送信中断) と XON (送信再開) のコマンドを挿入する方式をソフトウェアフロー制御と呼びます。

表 14-4 RS-232 の信号名と機能

ピン番号	信号名	入出力	備 考
1	DCD (Data carrier detect)	入力	キャリア検出
2	RXD (Receive data)	入力	受信データ
3	TXD (Transmit data)	出力	送信データ
4	DTR (Data terminal ready)	出力	データ端末レディ
5	GND	—	グランド
6	DSR (Data set ready)	入力	データセットレディ
7	RTS (Request to send)	出力	送信リクエスト
8	CTS (Clear to send)	入力	送信可
9	RI (Ring indicator)	入力	リング表示

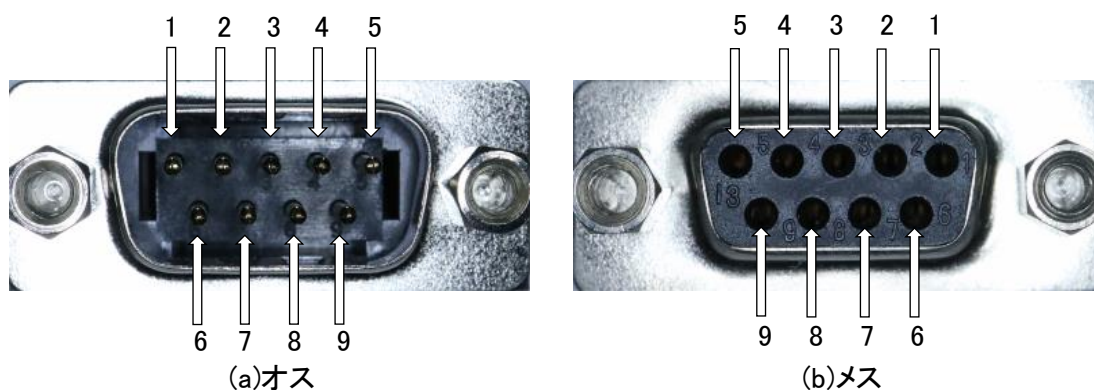


RS-232 ケーブルにはクロス結線とストレート結線がありますが、見た目には区別が付きません。デジタルマルチメータの導通ブザーまたは抵抗レンジで、両端の受信と送信のピンにテストピンを当てて電氣的に確認します。

### 14.3.2. コネクタのピン配置と結線の種類

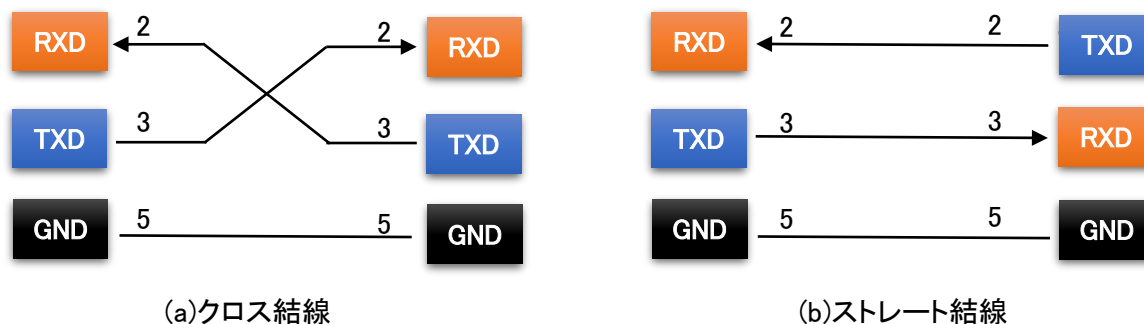
コネクタは台形状のコネクタ面をアルファベットの D に見立て、**D サブ**(D-Sub: D-Subminiature)と呼ばれています。D サブコネクタにはオス(male)とメス(female)があります(図 14-4)。オスのコネクタはピンで、メスはソケットで構成されています。なお、オスのピンやメスのソケットの番号の配列は異なります。PC 側のコネクタはオスになります。

図 14-4 RS-232 の D サブコネクタとピン番号



2 台の機器を RS-232 ケーブルで接続する場合、各機器の RS-232 コネクタの受信信号と送信信号のピン番号を機器の仕様書等で確認してください。図 14-5(a)のように、双方の受信信号と送信信号のピン番号が同じ場合、**クロス結線**の RS-232 ケーブルを使用します。また、図 14-5(b)のように、受信信号と送信信号のピン番号が入れ替わっている場合は、**ストレート結線**のケーブルを使用します。また、ケーブルのコネクタの形状にはオスとメスがあるので、機器側との組み合わせを確認してください。なお、コネクタのオスとメスを機械的に変換する**ジェンダーチェンジャー**と呼ばれる商品があります。

図 14-5 RS-232 ケーブルのクロス結線とストレート結線





### 14.3.3. 電気的特性

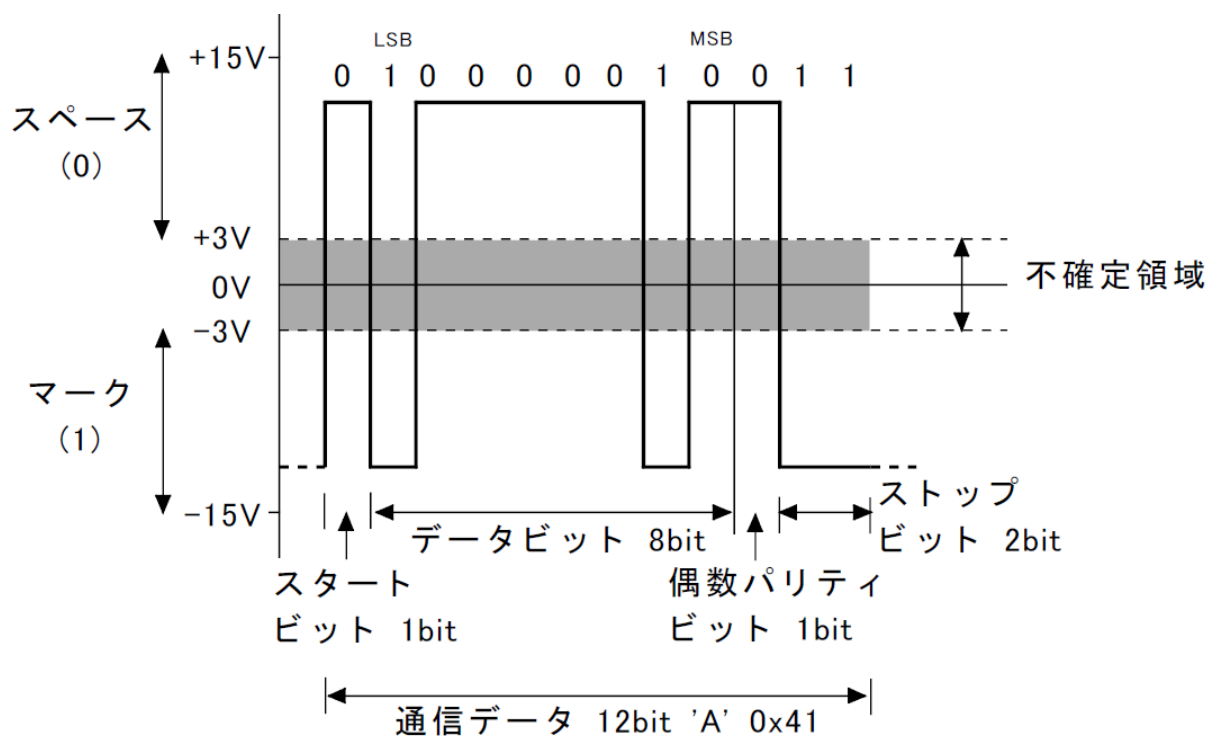
RS-232 信号の特徴は、論理値で電圧の極性が反転していることです（表 14-5）。論理値が 1 のときは**マーク**と呼ばれ、電圧範囲は-3V から-15V です。また、論理値が 0 のときは**スペース**と呼ばれ、電圧範囲は+3V から+15V です。-3V から+3V は不確定領域です。

例として、データビットにアスキーコード A(0x41)を伝送したタイミングチャートを図 14-6 に示します。データフォーマットは、スタートビット、データビット 8bit、偶数パリティ、ストップビット 2bit とし、通信データの 1 単位は 12bit になります。

表 14-5 RS-232 の論理値と電圧範囲

論理値	名 称	電圧範囲
1	マーク(Mark)	-15V から-3V
0	スペース(Space)	+3V から+15V

図 14-6 RS-232 のデータフォーマット



## 14.4. ラズパイのシリアルポートの設定と構成

ラズパイのシリアルポートを使用するには、「Raspberry Pi の設定」の画面で「有効」に設定します。また、シリアルポートの UART には、「mini UART」と「PL011」の2種類があります。シリアルポートの設定方法と2種類の UART について解説します。

### 14.4.1. シリアルポートの設定

ラズパイのシリアルポートの信号名とピン番号を表 14-6 に示します。TXD は送信信号で、RXD は受信信号です。表 14-4 に示した RS-232 の制御信号はありません。

表 14-6 シリアルポートの信号名とピン番号

信号名	拡張コネクタ		信号方向
	GPIO	ピン番号	
TXD	14	8	出力(送信)
RXD	15	10	入力(受信)

シリアルポートを有効にするために、メニュー⇒[設定]⇒[Raspberry Pi の設定]⇒[インターフェイス]のタブを開きます(図 14-7)。「Serial Port」と「Serial Console」の両方の「有効」にチェック⇒「OK」で再起動します。

#### ● Serial Poart (シリアルポート)(①)

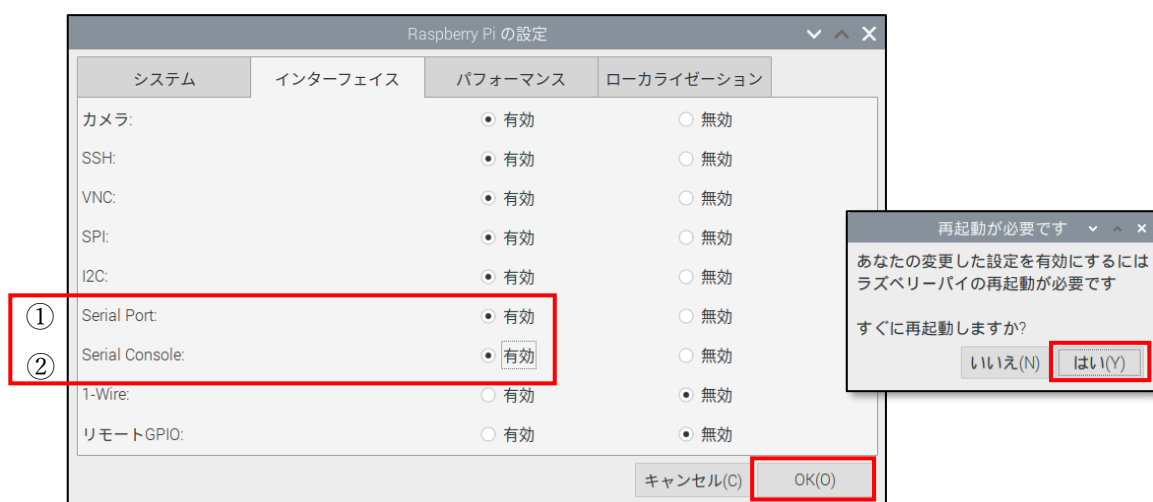
- ・「有効」の場合、シリアルポートの使用が可能になります。
- ・「無効」の場合、シリアルポートは無効になり、「Serial Console」はグレースアウトされ、操作の対象から外れます。

#### ● Serial Console(シリアルコンソール)(②)

シリアルコンソールとは Linux のコンソールのことです。コンソールでは、ターミナル(LXTerminal)と同じように、Linux のコマンドを実行できます。

- ・「有効」の場合、他の端末(PC等)からラズパイにログインして、CUI環境でシェルと対話的に作業ができます。
- ・「無効」の場合、コンソールの機能は停止します。

図 14-7 インターフェイスの設定画面



## 14.4.2. mini UART と PL011 の特徴

ラズパイで利用できる UART の型式には、mini UART と PL011 の 2 種類があり、SoC に内蔵されています(表 14-7)。PL011 は、シリアル通信用 IC の 16550<sup>3</sup>と互換性のある高機能な UART です。一方、mini UART はサブセット的な仕様で、パリティビット、ブ레이크信号の検出、フレーミングエラーの検出などの機能はありません[5], [6]。ラズパイでは、40 ピンの拡張コネクタに接続される UART をプライマリ UART と呼んでいます。信号線は GPIO14(送信)と GPIO15(受信)に接続されています。ラズパイ内部の Bluetooth に接続されている UART をセカンダリ UART と呼んでいます。Serial Port を「有効」にするとデフォルトでは、mini UART がプライマリ UART に接続され、PL011 がセカンダリ UART に接続されます。

表 14-7 mini UART と PL011 の仕様

型式	mini UART	PL011
UART 名	プライマリ UART	セカンダリ UART
接続先	拡張コネクタ	Bluetooth
スタートビット	1bit	1bit
データビット	7bit, 8bit	5bit, 6bit, 7bit, 8bit
パリティビット	無効	無効、奇数(odd)、偶数(even)
ストップビット	1bit	1bit, 2bit

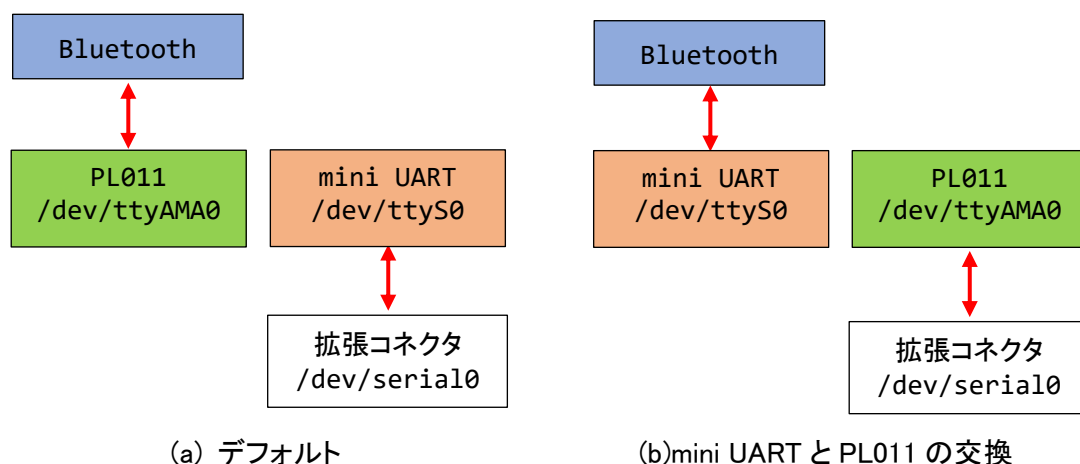
<sup>3</sup> 16550 は National Semiconductor 社が 1987 年に開発した UART で、PC/AT 互換機のシリアル通信用 IC として広く普及し、その仕様が UART のデファクトスタンダードになりました。

型式と UART 名のデバイス名を表 14-8 に示します。「/dev/serial0」と「/dev/serial1」は、「/dev/ttyS0」または「/dev/ttyAMA0」のどちらかのリンクです。デフォルトでは、「/dev/serial0」は「/dev/ttyS0」(mini UART)にリンクしています(図 14-8(a))。mini UART と PL011 の接続先を交換すると「/dev/serial0」は「/dev/ttyAMA0」(PL011)にリンクします(図 14-8(b))。なお、/boot/config.txt の設定により mini UART と PL011 の接続先を交換することができます。その手続きの詳細は 14.11 節で解説します。

表 14-8 型式・UART 名のデバイス名

型式・UART 名	デバイス名
mini UART	/dev/ttyS0
PL011	/dev/ttyAMA0
プライマリ UART	/dev/serial0
セカンダリ UART	/dev/serial1

図 14-8 mini UART と PL011 の内部接続



## Column

## フレーミング・エラーとは

データフォーマットのストップビットのあるべき位置にストップビットが無い場合をフレーミング・エラー(Framing Error)と言います。ストップビットは1と定義されていますが、0だった場合がフレーミング・エラーです。データは正しく伝送されないため文字化けなどの現象が起こります。原因は、通信速度、データビットの長さ、パリティビットの有無等のミスマッチが考えられます。

## 14.5. RS-232 インタフェース回路

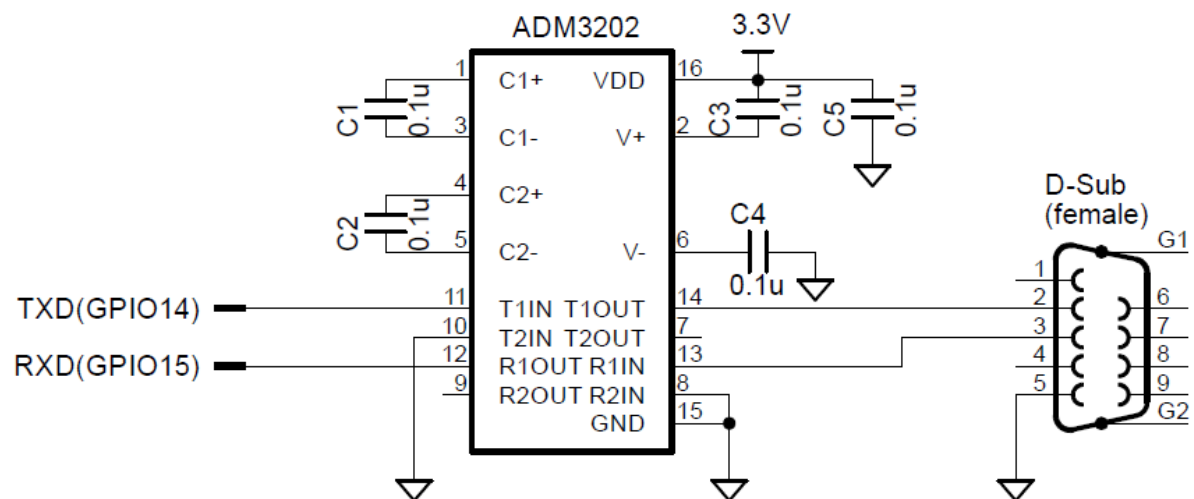
極性が反転する RS-232 信号の電圧レベルは、一般的なデジタル回路の電圧レベルと異なります。そこで、ラズパイに RS-232 の仕様（表 14-5）に準拠させたインタフェース回路が必要となります。

### 14.5.1. 回路図

アナログ・デバイセズ社の RS-232 ドライバ/レシーバ IC（ADM3202）を使用した回路図を図 14-9 に示します[7]。ADM3202 は外付けのコンデンサを利用したチャージ・ポンプによる倍電圧昇圧回路と電圧反転回路を内蔵しており、ADM3202 の VDD に供給される +3.3V から 2 倍の +6.6V とマイナス電圧の -6.6V を発生します。

図 14-9 の回路は、表 14-1 に示した USB シリアル変換ケーブル（秋月電子通商：M-08343）の仕様に合わせてあります。そのため、D サブコネクタ（メス）（図 14-4(b)）を使用し、ラズパイ側の RXD と TXD の信号線は D サブコネクタでクロス結線しています。なお、USB シリアル変換ケーブル（秋月電子通商：M-08343）の仕様と異なるケーブルを使用する場合は、使用されるケーブルの仕様に合わせてコネクタや配線を変更してください。

図 14-9 RS-232 のインタフェース回路



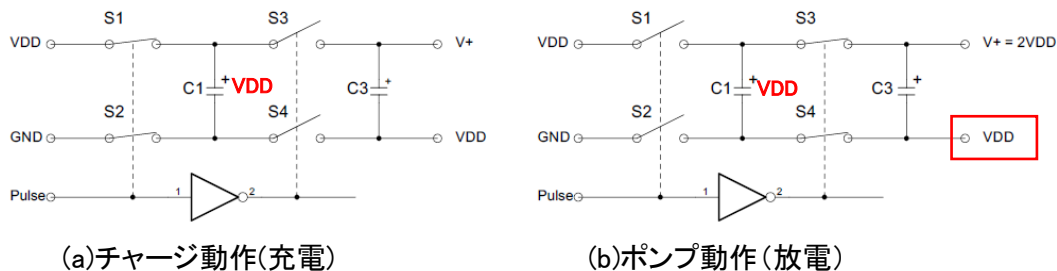
## チャージ・ポンプとは

チャージ・ポンプ(Charge Pump)とは、コンデンサとスイッチを組み合わせることによって、電圧を上昇させるための電子回路です。ADM3202 にはチャージ・ポンプを利用した倍電圧昇圧回路と電圧反転回路を内蔵しています[8]。

### ●倍電圧昇圧回路

図 14-10 のスイッチ (S1~S4) は半導体で構成され、Pulse (パルス) 信号により ON と OFF の動作をします。S1 と S2、S3 と S4 がペアになっています。最初に、図 14-10(a)の S1 と S2 が ON で、S3 と S4 が OFF のとき、コンデンサ C1 は充電されて電圧 VDD と等しくなります。次に、パルス信号の H と L が切り替わって、S1 と S2 が OFF で、S3 と S4 が ON のとき (図 14-10(b))、C1 のマイナス側は VDD に接続されるため、充電された VDD と直列接続になって出力 V+からは 2 倍の VDD が出力され、C3 を充電します。再び図 14-10(a)のチャージ動作に戻り、C1 が充電されている間、C3 は放電します。パルス信号により充電と放電を繰り返します。

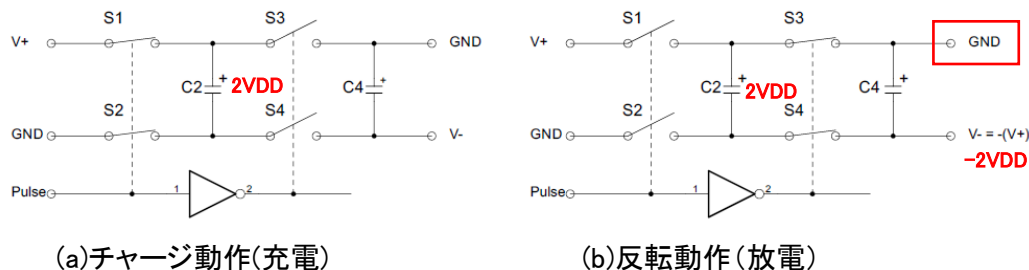
図 14-10 倍電圧昇圧回路 (Charge Pump Voltage Doubler)



### ●電圧反転回路

図 14-11(a)のとき、C2 は倍電圧昇圧回路の出力 V+により 2 倍の VDD 電圧で充電されます。次に、図 14-11(b)のとき、C2 のプラス側は GND に接続されるので、C2 のマイナス側は必然的に  $-(V+)$  となって出力 V-から出力され、C4 を充電します。

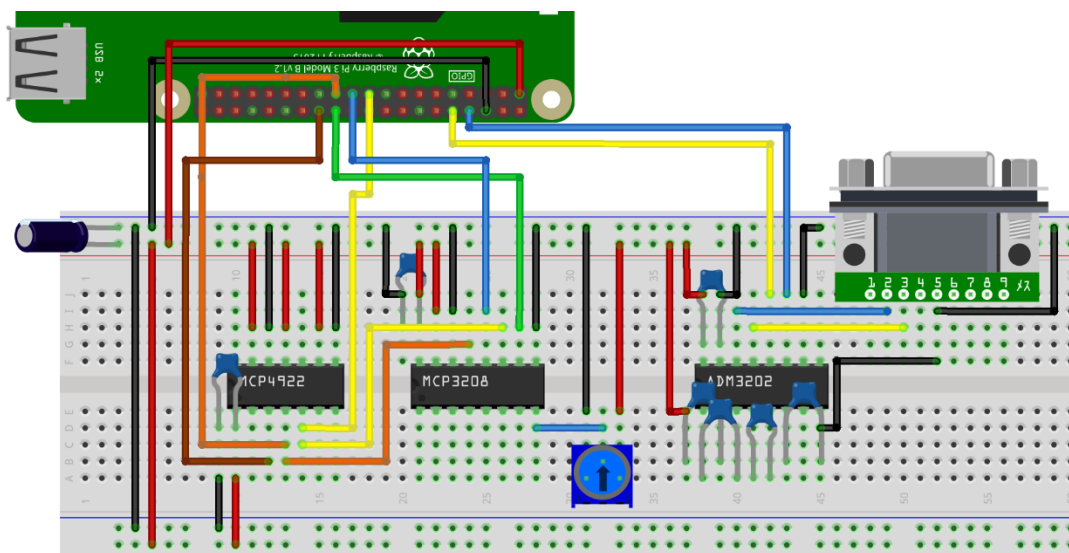
図 14-11 電圧反転回路 (Charge Pump Voltage Inverter)



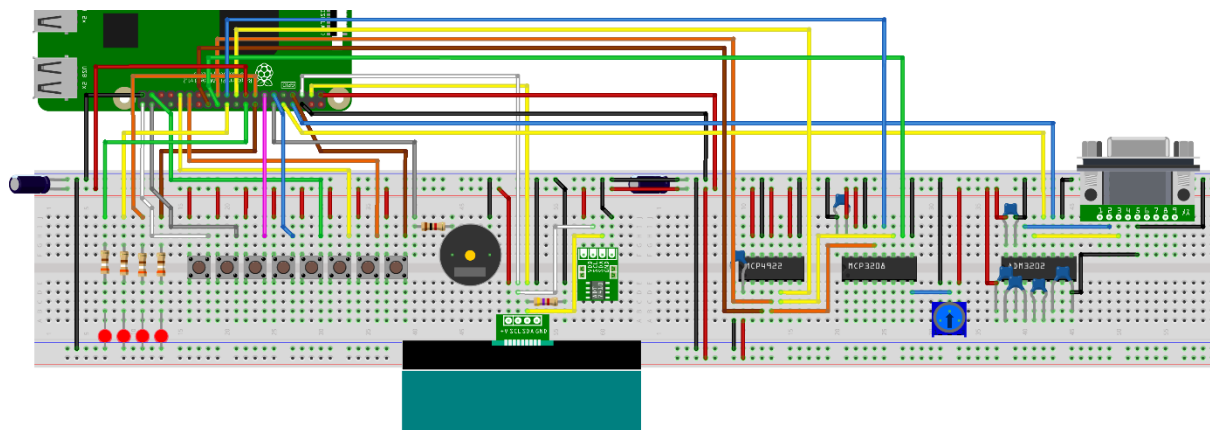
## 14.5.2. 配線図

ADM3202 の配線図を図 14-12(a)に示します。また、「Chapter8 SPI バスを使う」で製作した回路に、追加して配線した図を図 14-12(b)に示します。

図 14-12 RS-232 インタフェース回路の配線図



(a) ADM3202 の配線図



(b) 全体の配線図



## 14.6. PC のシリアルポートの通信設定

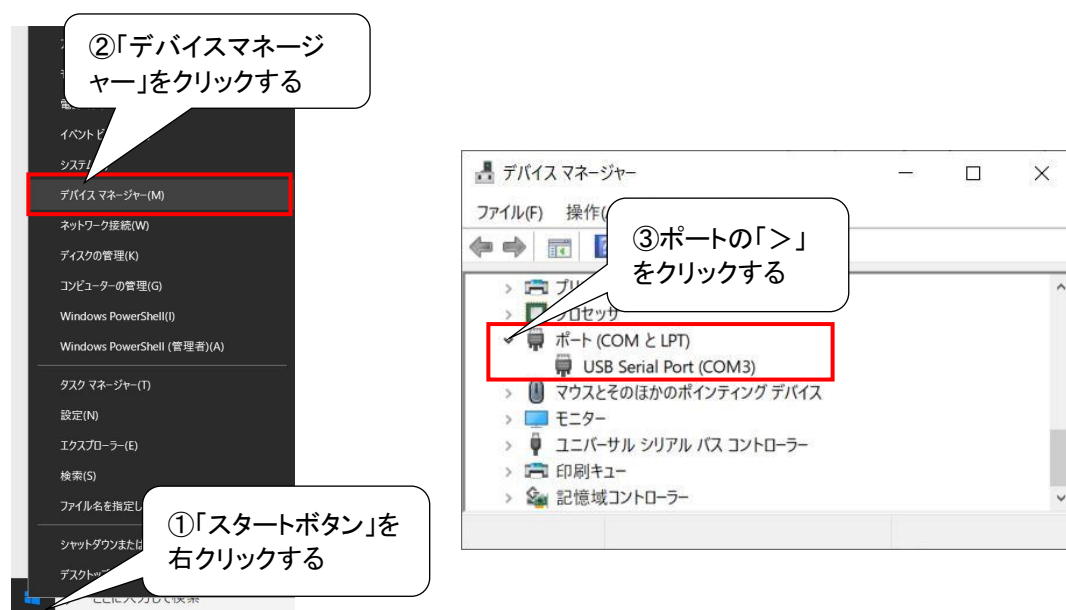
PC から RS-232 を利用してラズパイにログインしてみます。本節では、PC 側に RS-232 インタフェースがない場合を想定して、表 14-1 に示した USB シリアル変換ケーブル（秋月電子通商：M-08343）を使用します。本ケーブルを PC（Windows10）の USB コネクタに挿入すると、ドライバは自動的にインストールされます。他社の変換ケーブルのインストール方法については、取扱説明書を参照してください。

変換ケーブルのインストールが完了すると、RS-232 インタフェースは **COM ポート**（Communication Port）に割り当てられます。COM ポートとはシリアルポートのことです。PC に複数の COM ポートを割り当てることが可能で、COM ポートを識別するために COM に番号が振られます。本書では、**COM 番号**と呼びます。デバイスマネージャーで COM 番号を確認しましょう。なお、以降において、「USB シリアル変換ケーブル」を RS-232 ケーブルに含めて扱います。

### 14.6.1. COM 番号の確認方法

デバイスマネージャーは、図 14-13 のようにデスクトップ画面の左下、[スタートボタン（Windows マーク）]を右クリック ① ⇒ [デバイスマネージャー] ② をクリックします。デバイスマネージャーの中の [ポート（COM と LPT）] ③ を展開して表示される「USB Serial Port (COM3)」がインストールしたシリアルポートです。本書では COM 番号は「COM3」ですが、COM 番号は PC の環境で異なるので、自身で確認した COM 番号を使用してください。

図 14-13 デバイスマネージャーの表示

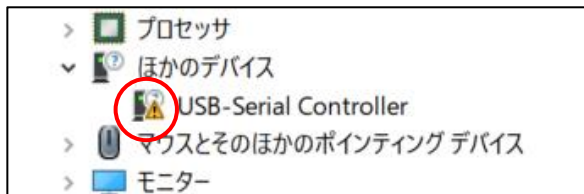




### <COM ポートのトラブルシューティング>

図 14-14 に示すように、USB-Serial Controller のアイコンに「!」が一緒に表示されている場合は、不具合の状態を示しています。PC を再起動すると修復される場合もありますが、このままの状態ではシリアル通信は機能しません。変換ケーブルのドライバが、Windows OS のバージョンや 32bit/64bit に対応しているかどうか確認してください。

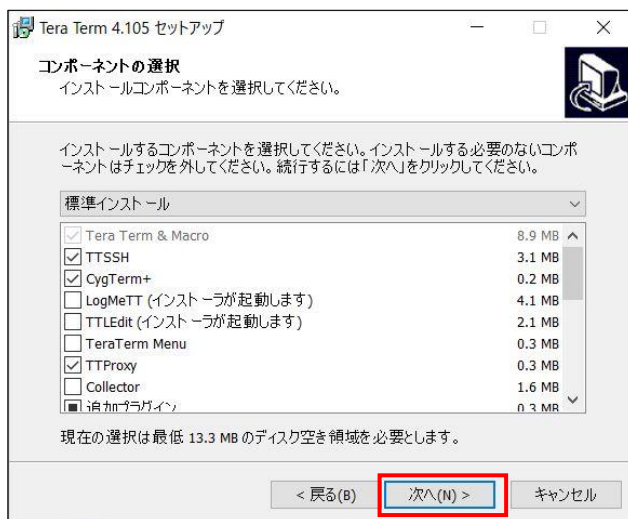
図 14-14 デバイスマネージャーでの不具合の表示



## 14.6.2. ターミナルソフト Tera Term の設定

次に、オープンソース・ソフトウェアの Tera Term (テラターム) というターミナルソフトの設定を行います。Tera Term をお持ちでない方は、Tera Term Project の Web サイト<sup>4</sup>からダウンロードできます。ダウンロードが完了したら、アイコンをダブルクリックするとインストールが始まります。インストールの途中、コンポーネントの選択画面 (図 14-15) が表示されますが、「次へ」ボタンをクリックしてください。

図 14-15 Tera Term のコンポーネントの選択画面



<sup>4</sup> Tera Term Project URL <https://ttssh2.osdn.jp/> 本書ではバージョン 4.105 で解説します。

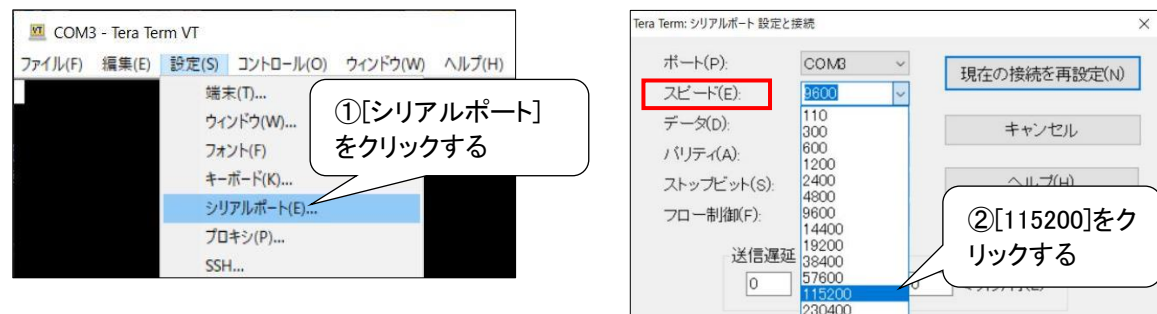
インストールが完了した後、Tera Term を起動させると「新しい接続」のダイアログボックスが表示されます（図 14-16）。[シリアル] (①) ⇒ [ポート] (②) からデバイスマネージャーで確認した COM 番号を選択し、[OK] (③) をクリックします。

図 14-16 COM 番号の選択



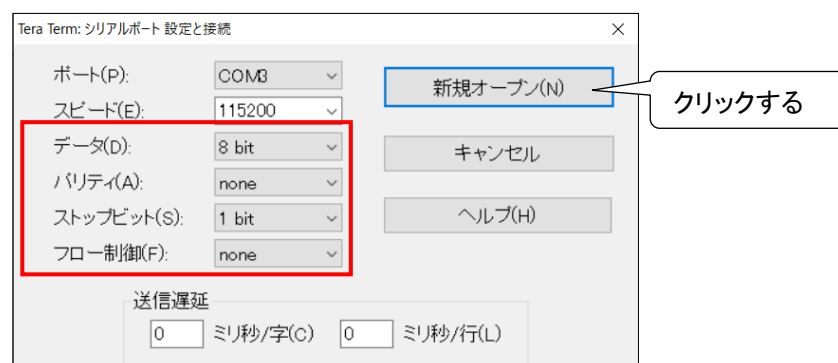
まず、通信速度を設定します。Tera Term のメニュー [設定] ⇒ [シリアルポート] (①) ⇒ [シリアルポート設定と接続] 画面 ⇒ [スピード] で「115200」 (②) に設定します（図 14-17）。

図 14-17 通信速度の設定



続いて、データは「8bit」、パリティは「none」（無効）、ストップ（ストップビット）は「1bit」、フロー制御は「none」に設定して、[新規オープン]または[現在の接続を再設定]をクリックします（図 14-18）。また、Tera Term のメニュー [設定] ⇒ [設定の保存] で、[TERATERM. INI] を上書き保存すると設定が記憶されます。

図 14-18 その他のシリアルポートの設定



次に、Tera Term の改行コードを設定しますが、ここで改行コードについて説明します。「徹底入門」の Chapter7 で触れたように、文字パターンはアスキーコードで標準化されています。改行のように画面に表示されない文字を制御文字またはエスケープシーケンスと呼び、C 言語ではバックスラッシュ (\) を使用して表現します[9]。主なエスケープシーケンスを表 14-9 に示します。表 14-9 の略号は「符号化文字集合用制御機能(JIS X0211-1994)」で規定されています。

略号 **NUL** の名称は空白ですが、スペース (0x20) のことではありません。**NUL** は「何もしない」という特殊な制御を表しています。C 言語では文字列の終端を表すヌル文字として用いられています[10]。

**BS** (後退) と **HT** (水平タブ) はキーボードのと同じで、**BS** はバックスペースキーとして文字の消去を行い、**HT** はタブ (Tab) キーで C 言語のインデントに使用しています。

**LF** (改行) は位置を次行に移動し、**CR** (復帰) は位置を行頭 (左端) に移動します。キーボードの「Enter」には改行の働きがありますが、OS によって改行コードが異なります。一般に、UNIX 系 OS やその上で動作するソフトウェアでは改行コードに **LF** が、macOS 系では **CR** が、Windows 系では **CR** と **LF** の 2 バイト (**CR+LF**) が、それぞれ標準的な改行コードとして用いられています[11]。

表 14-9 主なエスケープシーケンス

名 称	略号	16 進	エスケープ シーケンス	概 要
null 空白	NUL	0x0	\0	ヌル文字(null)
Backspace 後退	BS	0x08	\b	位置を 1 文字左に移動
Horizontal Tabulation 水平タブ	HT	0x09	\t	位置を次のタブストップに移動
Line Feed 改行	LF	0x0a	\n	位置を次行に移動
Carriage Return 復帰	CR	0x0d	\r	位置を行頭に移動

さて、Tera Term の改行コードの設定に戻ります。Tera Term のメニュー [設定] ⇒ [端末] より、[受信]と[送信]に **CR** を設定します (図 14-19)。この設定は「送受信する CR をどのように扱うか」という意味です。CR に設定することで、CR のままにする (何も変換しない) の指示になります。従って、Tera Term は次のように動作します。

- **CR+LF** を受信したとき、Tera Term のカーソルは行頭 (左端) に移動して改行されます<sup>5</sup>。
- Tera Term からキーボードの「Enter」を押すと、「Enter」のキーコード 0x0d (CR) を送信します。

図 14-19 Tera Term の改行コードの設定



<sup>5</sup> 受信 (R)に「CR+LF」を設定すると、受信された CR を CR+LF に変換します。CR+LF を受信すると、CR+LF+LF に変換され、LF が一つ余分になります。詳しくは Tera Term のヘルプを参照してください。

## 14.7. Linux のコンソールからログインする

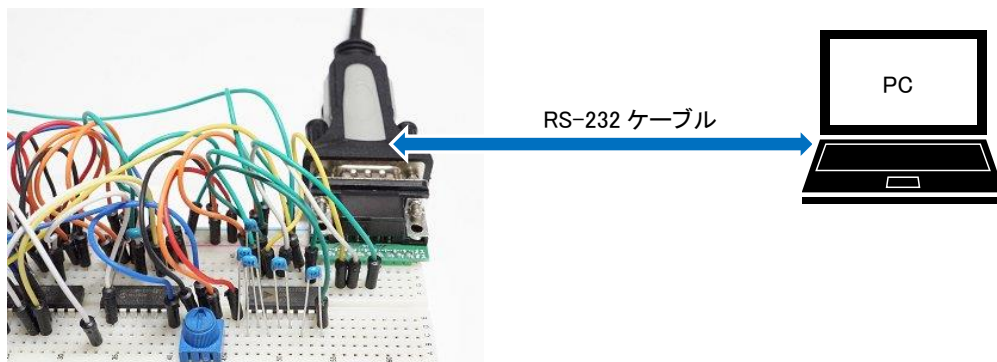
ラズパイの RS-232 のインタフェース回路が完成したので、通信実験をしましょう。この実験では、PC からラズパイの Linux のコンソールにログインします。Linux のコンソールでは、ターミナル(LXTerminal)と同じように Linux コマンドを実行できます。

例えば、ラズパイを機器に組み込む場合、ディスプレイやキーボードを外した状態になります。そこで、PC とラズパイのシリアルポート (RS-232) を接続することにより、Linux のコンソールからログインして、様々な作業が可能になります。Wi-Fi が使えない環境下では有効な手段です。Tera Term を利用して PC からラズパイにログインして、簡単なプログラムを作成して実行してみましょう。

### 14.7.1. RS-232 ケーブルで接続する

RS-232 ケーブルを試作した回路の D サブコネクタに接続します (図 14-20)。ケーブルの長さが足りない場合は、ストレート結線の延長ケーブルを使用してください。

図 14-20 RS-232 ケーブルを接続した写真



### 14.7.2. PC からログインする

ラズパイのインターフェイスの設定画面 (図 14-7) で、「Serial Port」と「Serial Console」の両方が「有効」に設定されていることを確認してください。再設定した場合は、再起動させます。PC 側の Tera Term を表示させて、キーボードの [Enter] を押してみてください。少し時間がかかる場合がありますが、「login」のプロンプトが表示されます (図 14-21)。ログイン名とパスワードはデフォルトでは「pi」と「raspberrry」ですが、Chapter2 で設定したパスワードを入力してください。なお、キーボードから入力したパスワードはセキュリティ保護のため Tera Term のターミナル画面には表示されません。

ログインが成功すると、プロンプト「\$」が表示され、ターミナル(LXTerminal)と同様の画面になります (図 14-22)。

図 14-21 Tera Term に表示されたログイン画面

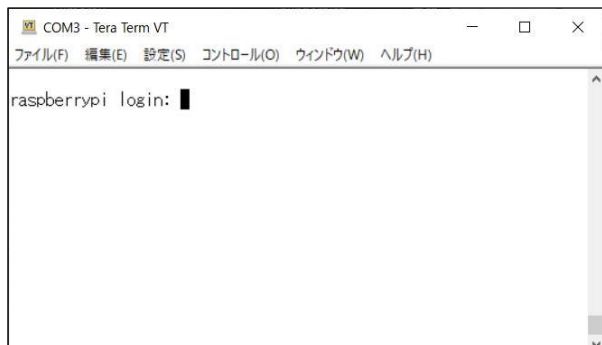
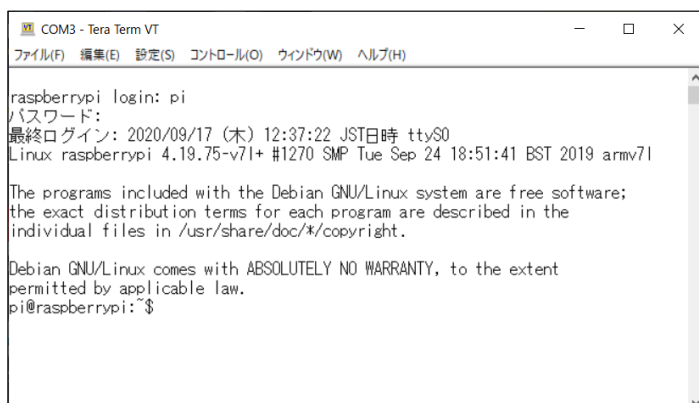


図 14-22 Linux のコンソール画面



#### <動作しないときのトラブルシューティング>

原因をラズパイ側と PC 側に、切り分けて探ります。

##### ●ラズパイ側

- ・ インターフェイスの設定画面で、「Serial Port」と「Serial Console」の両方が「有効」になっていますか？
- ・ 配線や部品の実装が正しいかチェックします。
- ・ ブレッドボードでの部品やジャンパーワイヤの接触不良をチェックします。

##### ●PC 側

- ・ Tera Term の COM 番号やスピードなどの設定を確認します。
- ・ Tera Term を一度終了して、再起動してください。
- ・ デバイスマネージャーに COM ポートの不具合が表示されていませんか？
- ・ RS-232 延長ケーブルを使用する場合、ストレート結線とします。不明な場合は双方のコネクタのピンをテスターで導通チェックして確認します。



### 14.7.3. コンソールでプログラムを作成し実行する

C 言語でハローワールドのプログラムをテキストエディタ「nano」で作成して、実行してみましょう。もし、nano エディタがインストールされていない場合は、次のコマンドでインストールしてください。

```
$ sudo apt install nano -y
```

cd コマンドで、「MyApp」のディレクトリへ移動します。プロンプトから「nano hello2.c」を入力して[Enter]を押すと、nano のテキストエディタが起動します。

```
$ cd MyApp
/MyApp$ nano hello2.c
```

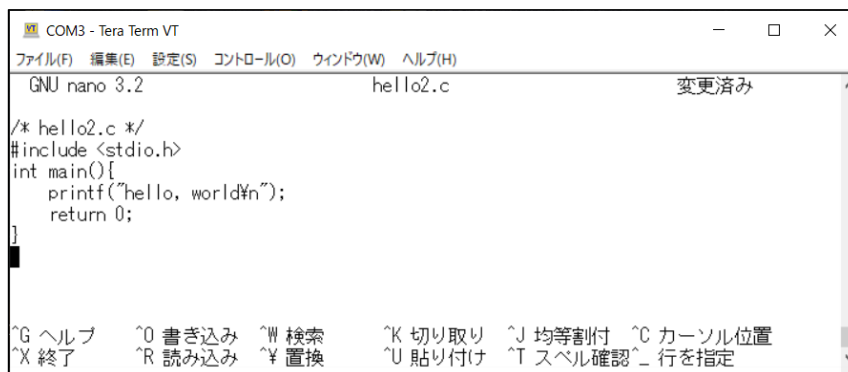
nano の操作コマンドは画面の下に表示され、[Ctrl]とコマンド名を同時に押すと実行されます(図 14-23)。例えば、[^G]ヘルプを表示するときは、[Ctrl]と[G] (Ctrl + G) を同時に押します。また、ヘルプを閉じる場合は、(Ctrl + X) を押します。nano では矢印キーが使えるので、テキストエディタの中では初心者でも比較的容易に扱うことができます。

Chapter3 でも作成しましたが、hello, world をターミナルに表示するプログラムのソースファイル「hello2.c」を作成します。リスト 14-1 のソースコード (6 行) を nano のエディタ画面に入力します。Tera Term では、printf 関数の改行 (\n) は¥n で表示されます。

リスト 14-1 hello2.c

```
/* hello2.c */
#include <stdio.h>
int main(){
    printf("hello, world¥n");
    return 0;
}
```

図 14-23 nano エディタの画面



ソースコードを保存する場合は、(Ctrl + O(オー))を押します。次に、nano を終了する場合は、(Ctrl + X)を押すと、nano を終了してコマンドプロンプトに戻ります。

gcc でコンパイルして、実行ファイルを作成します。カレントディレクトリを表す相対パス「./」を付けて、実行すると「hello, world」と表示されます。なお、ログアウトするときは「exit」コマンドを実行します。

```
/MyApp$ gcc -o hello2 hello2.c
/MyApp$ ./hello2
hello, world
/MyApp$ exit
```



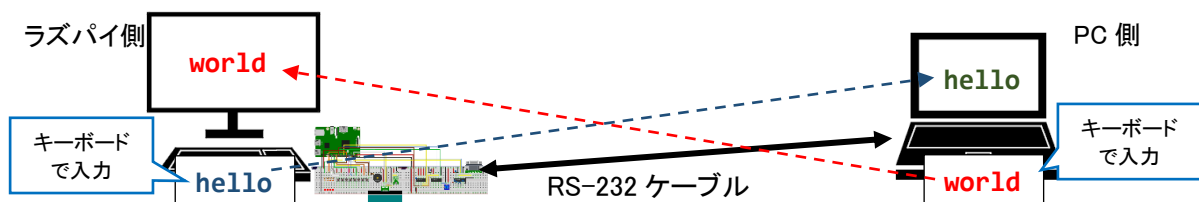
シリアルポートを利用した CUI 環境と GUI 環境の両方が使えるので、1 台のラズパイを 2 人で使用することもできます。UNIX 系 OS は、マルチユーザーでの使用を前提に設計されています。

## 14.8. ラズパイと PC でテレタイプする

前節では、Linux のコンソールを使用してラズパイにログインすることができました。次に、ラズパイと PC 間で、データ伝送を行います。データ伝送の送信と受信については、ユーザーがプログラムを作成してデータを操作します。本節では、PC とラズパイでテレタイプを行って、データ伝送の動作を確認しましょう。テレタイプとは、ユーザーが入力した文字を通信回線を利用して別の端末に送信したり、別の端末からの文字を受信して表示したりする装置や仕組みのことです。

例えば、図 14-24 に示すように、ラズパイのキーボードから「hello」とタイプすると、PC のディスプレイに表示され、次に PC 側から「world」とタイプするとラズパイのディスプレイに表示されます。各端末のキーボードから入力した文字は、自身のディスプレイには表示されず、相手のディスプレイに表示されます。

図 14-24 ラズパイと PC とのテレタイプ



### 14.8.1. Serial Console を無効にする

データ伝送中にシェルが応答しないように Serial Console を無効にします。メニュー ⇒ [設定] ⇒ [Raspberry Pi の設定] ⇒ [インターフェイス] のタブで [Serial Console] の「無効」にチェック⇒「OK」をクリックして、再起動します（図 14-25）。

図 14-25 Serial Console を無効に設定する



## 14.8.2. minicom のインストール

ラズパイ用のターミナルソフト minicom を次の手順でインストールします。インストールが完了したら、ラズパイを再起動させます。

```
$ sudo apt update
$ sudo apt install minicom -y
```



ラズパイがインターネットに接続していないと、アプリはインストールできません。

## 14.8.3. テレタイプの実験

PC とラズパイを RS-232 ケーブルで接続します。PC 側の Tera Term とラズパイ側の minicom の通信条件を設定し、テレタイプの実験を行います。

### ● Tera Term の設定

Tera Term のメニュー⇒[設定]⇒[シリアルポート]において、表 14-10 の設定値をセットします（図 14-16～図 14-18）。

表 14-10 Tera Term の設定

設定項目	設定値
スピード	115,200 bps
データ	8 bit
パリティ	none(無効)
ストップビット	1 bit
フロー制御	none

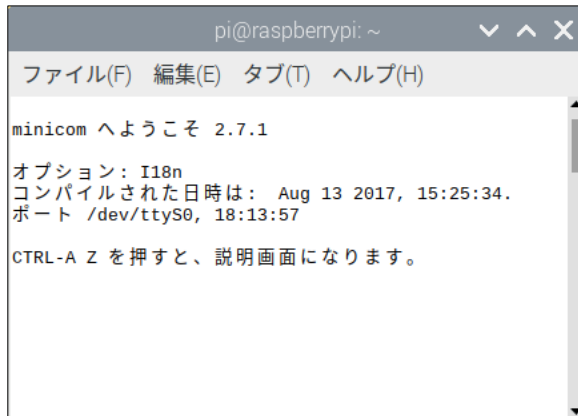
### ● minicom の起動と設定

ラズパイのターミナルから管理者権限で minicom を起動します(図 14-26)。minicom のオプション「-b」で通信速度を設定し、「-D」でプライマリ UART のデバイス名「/dev/serial0」を指定します。なお、minicom の通信条件のデフォルトでは、データビットは 8bit、パリティビット無効、ストップビットは 1bit です。

```
$ sudo minicom -b 115200 -D /dev/serial0
```

serial(小文字のエル)0(数字のゼロ)です。

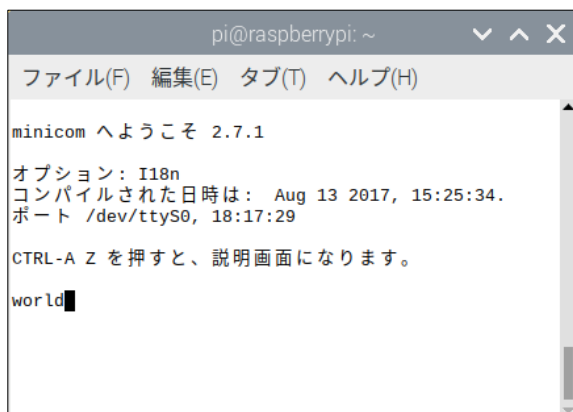
図 14-26 minicom の画面



### ● テレタイプの実験

ラズパイ側の minicom からキーボードで「hello」とタイプすると、文字データは通信ケーブルを経由して PC 側の Tera Term の画面に表示されます。次に、Tera Term から「world」とタイプすると minicom の画面に表示されます。

図 14-27 テレタイプの実験



(a)ラズパイ側の minicom で「hello」とタイプ



(b)PC 側の Tera Term で「world」とタイプ

### ● minicom の終了方法

[Ctrl]を先に押して、[A]を押します。画面下部に「CTRL-A Z for help…」のメッセージ表示されます。次に[Q]を押すと、「リセットせずに終了しますか?」「はい」「いいえ」が表示されます。「はい」を選択して[Enter]を押します。

## 14.9. WiringPi Serial Library 関連

Wiring Pi Serial Library 関連の概要について、使用例を交えて解説します。

### 14.9.1. wiringSerial.h ヘッダファイル

---

Wiring Pi Serial Library を使用する場合、ソースコードの先頭で wiringSerial.h のインクルードを記述します。

【例】 `#include <wiringSerial.h>`

### 14.9.2. シリアルポートをオープンする serialOpen 関数

---

【書式】 `int serialOpen(char *device, int baud);`

シリアルポートをオープンします。

- `*device` にはシリアルポートのデバイス名「`/dev/serial0`」を指定します。
- `baud` には通信速度を設定します。serialOpen 関数では通信速度を明記していませんが、後述する Linux の `termios` 構造体の仕様に準拠しているようです。Pi 3B と Pi 4B において、「14.8.3 テレタイプの実験」で確認したところ、下記の通信速度が有効でした。なお、ラズパイの通信速度のデフォルトは 115200 bps です。

1200  
2400  
4800  
9600  
19200  
38400  
57600  
115200  
230400  
460800

- 戻り値は `fd`(ファイルディスクリプタ)です。取得した `fd` は、他のシリアル通信の関数等で使用します。また、エラーの場合は -1 を返します。

【例】シリアルポートのデバイス名と通信速度 115200bps を設定し、戻り値を取得します。

```
fd = serialOpen( "/dev/serial0", 115200);
```

### 14.9.3. シリアルポートをクローズする serialClose 関数

---

【書式】 `void serialClose(int fd);`

`serialOpen` 関数で取得した `fd` を引数にして、シリアル通信をクローズします。

### 14.9.4. 1 バイトを送信する serialPutchar 関数

---

【書式】 `void serialPutchar(int fd, unsigned char c);`

`serialOpen` 関数で取得した `fd` を引数にして、1 バイトのデータを送信します。

【例】 文字 A はアスキーコード 0x41 に変換されて送信されます。

```
serialPutchar (fd, 'A');
```

### 14.9.5. バッファをクリアする serialFlush 関数

---

【書式】 `void serialFlush(int fd);`

`serialFlush` 関数を実行すると、受信されたデータ及び送信されるデータを破棄します。

### 14.9.6. 1 バイトを受信する serialGetchar 関数

---

【書式】 `int serialGetchar(int fd);`

1 バイトの受信データが戻り値になります。

【例】 受信すると、1 バイトのデータが変数 `rxid` に保存されます。

```
rxid = serialGetchar(fd);
```

### 14.9.7. 受信をチェックする serialDataAvail 関数

---

【書式】 `int serialDataAvail(int fd);`

データの受信を確認します。戻り値は受信したバイト数です。受信データがないときは戻り値は 0 です。また、エラーの場合は -1 を返します。



## 14.9.8. null 付き文字列を送信する serialPuts 関数

【書式】 `void serialPuts(int fd, char *s);`

文字列の終端を示す `null` を文字列の末尾に付加して送信します。`null` はヌル文字と呼ばれ値は 0 です。なお、`stdio.h` の `NULL` も 0 ですが、ナルポインタのことで意味は異なります。

## 14.9.9. 文字列を送信する serialPrintf 関数

【書式】 `void serialPrintf(int fd, char *message, ...);`

文字列を送信します。

【例】 "hello, world"を送信します。

```
serialPrintf(fd, "hello, world");
```

## 14.9.10. 通信パラメータを設定する termios 構造体

データビット、パリティビット、ストップビットの通信パラメータについては、Linux の `termios` 構造体で設定します。`termios` 構造体には多数のパラメータ<sup>6</sup>がありますが、データビット、パリティビット、ストップビットの設定には、`termios` 構造体のメンバ `c_cflag` を使用します。データビット、パリティビット、ストップビットは `c_cflag` に配置されています（表 14-11）。各ビットには、マクロと設定値が定義されています。マクロを利用して、AND(&)、OR(|)、NOT(~)のビット演算により `c_cflag` に設定します。なお、UART の仕様（表 14-7）の範囲外の設定は事実上無効です。例えば、mini UART にパリティビットを指定した場合など。

表 14-11 `c_cflag` のデータビット、パリティビット、ストップビットの配置

ビット目	9	8	7	6	5	4	3	2	1	0
項目	パリティビット			ストップビット	データビット					
マクロと 設定値	PARODD	PARENB		CSTOPB	CSIZE					
	偶数(even) 0	無効 0		1bit 0	8bit (CS8) 11					
	奇数(odd) 1	有効 1		2bit 1	7bit (CS7) 10					
					6bit (CS6) 01					
					5bit (CS5) 00					

<sup>6</sup> debian Linux Programmer's Manual TERMIOS(3)

URL <https://manpages.debian.org/unstable/manpages-ja-dev/termios.3.ja.html>

## ● データビット

- データビットを 00 にセット (クリア)  
`c_cflag &= ~CSIZE`
- データビット 8bit `c_cflag |= CS8`
- データビット 7bit `c_cflag |= CS7`
- データビット 6bit `c_cflag |= CS6`
- データビット 5bit `c_cflag |= CS5`

## ● パリティビット

- パリティビット無効 `c_cflag &= ~PARENB`
- パリティビット有効 `c_cflag |= PARENB`
- 偶数(even)パリティ `c_cflag &= ~PARODD`
- 奇数(odd)パリティ `c_cflag |= PARODD`

## ● ストップビット

- ストップビット 1bit `c_cflag &= ~CSTOPB`
- ストップビット 2bit `c_cflag |= CSTOPB`

【例】 データビット 8bit、パリティビット無効、ストップビット 1bit の通信パラメータを設定します。termios.h と unistd.h をインクルードします。serialOpen 関数で、プライマリ UART (mini UART) を使用し、通信速度を 115,200bps に設定します。fd(ファイルディスクリプタ)は serialOpen 関数の戻り値です。param は構造体変数です。tcgetattr 関数で現在のパラメータを取得し、構造体のメンバー c\_cflag にパラメータを設定したら、tcsetattr 関数で有効化します。TCSANOW は直ちに変更を有効にするオプションです。

```
#include <termios.h>           //ヘッダファイル
#include <unistd.h>             //ヘッダファイル
... 中略
fd = serialOpen( "/dev/ttyserial0", 115200);
struct termios param           //構造体変数
tcgetattr(fd, &param);        //パラメータの取得
    param.c_cflag &= ~CSIZE;   //データビットのクリア
    param.c_cflag |= CS8;      //データビットを 8bit にセット
    param.c_cflag &= ~PARENB;  //パリティビットを無効にセット
    param.c_cflag &= ~CSTOPB;  //ストップビットを 1bit にセット
tcsetattr(fd, TCSANOW, &param); //パラメータの有効化
```

## 14.10. Geany のビルドの設定

本書では、Chapter 7 で作成したライブラリ (libMyPi.a と MyPi.h) と数学関数を使用することがあるので、Geany のビルドの設定は下記のように設定してください。数学オプションの「-lm」は L と M の小文字です。

ライブラリの作成方法は、「徹底入門」の Chapter 7 を参照してください。また、ライブラリ・ファイルの入手情報は、本章の「はじめに」を参照してください。

【例:ビルド】 `gcc -Wall -o "%e" "%f" -lwiringPi -lpthread -g -O0 libMyPi.a -lm`

【例:実行】 `sudo "./%e"`

↑  
O 大文字

↑  
エルエム小文字

## 14.11. mini UART でデータ伝送する

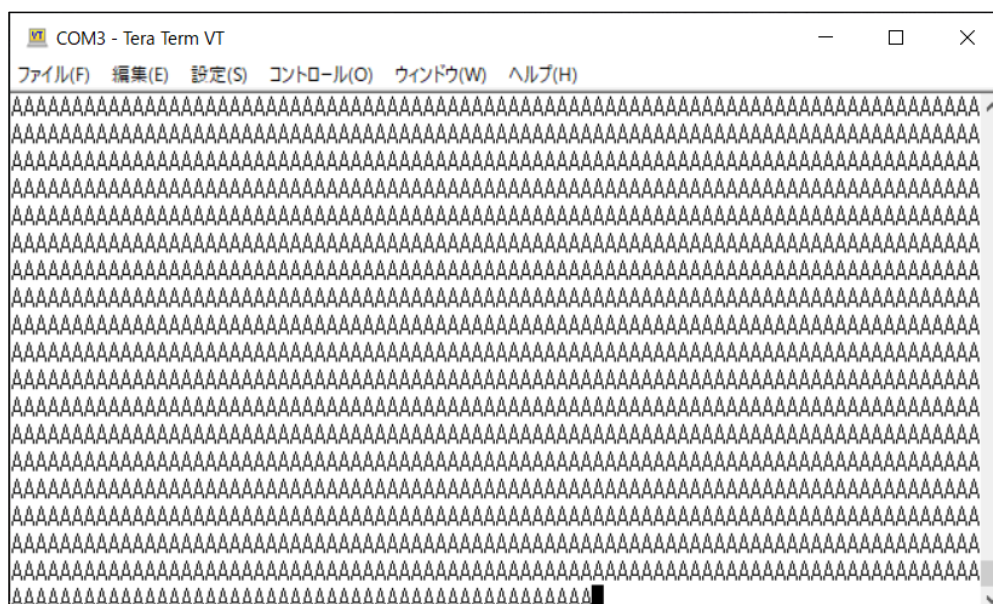
ラズパイ OS のデフォルトでは、mini UART がプライマリ UART に接続されています。mini UART を利用したデータ伝送を行きましょう。

### 14.11.1. データを送信する

ラズパイの mini UART から 1 文字「A」を繰り返し送信するプログラムを作成します。ラズパイと PC を RS-232 ケーブルで接続し、PC 側に Tera Term を使用して、ラズパイから受信したデータを表示させます（図 14-28）。ファイル名は「List14-1\_Txd01.c」とします。

- 条件
  - ・ 通信速度 115200bps、データビット 8bit、パリティビット無効、ストップビット 1bit

図 14-28 PC 側の Tera Term の画面

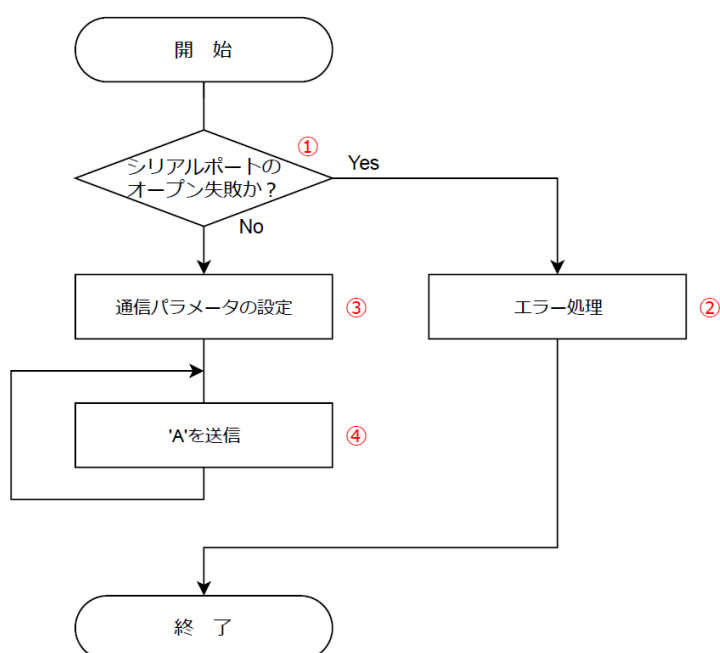


## 14.11.2. 送信のフローチャート

1 文字「A」を繰り返し送信するフローチャートを図 14-29 に示します。

- ① シリアルポートをオープンします。
- ② オープンに失敗した場合、エラー処理をして終了します。
- ③ データビット、パリティビット、ストップビットの通信パラメータを設定します。
- ④ 「A」を繰り返し送信します。

図 14-29 1 文字「A」を繰り返し送信するフローチャート



### 14.11.3. 送信のソースコード

ソースコード List14-1\_Txd01.c をリスト 14-2 に示します。

リスト 14-2 List14-1\_Txd01.c

```
#include <stdio.h>           //printf,etc
#include <stdlib.h>          // EXIT_SUCCESS,etc
#include <termios.h>         //tcgetattr,etc
#include <wiringSerial.h>    //serialOpen,etc ①

#define BPS 115200           //通信速度 ②

int main (void){
    char txdData ='A';      //送信データ
    int fd;
    struct termios param;   //構造体変数
    fd = serialOpen( "/dev/serial0", BPS); //シリアルポートのオープン ③
    if( fd == -1){
        fprintf(stderr,"Error serialOpen.\n");
        exit(EXIT_FAILURE);
    }
    tcgetattr(fd, &param);  //パラメータの取得
    param.c_cflag &= ~CSIZE; //データビットのクリア
    param.c_cflag |= CS8;    //データビットを 8bit にセット
    param.c_cflag &= ~PARENB; //パリティビットを無効にセット
    param.c_cflag &= ~CSTOPB; //ストップビットを 1bit にセット
    tcsetattr(fd,TCSANOW,&param); //パラメータの有効化 ⑤

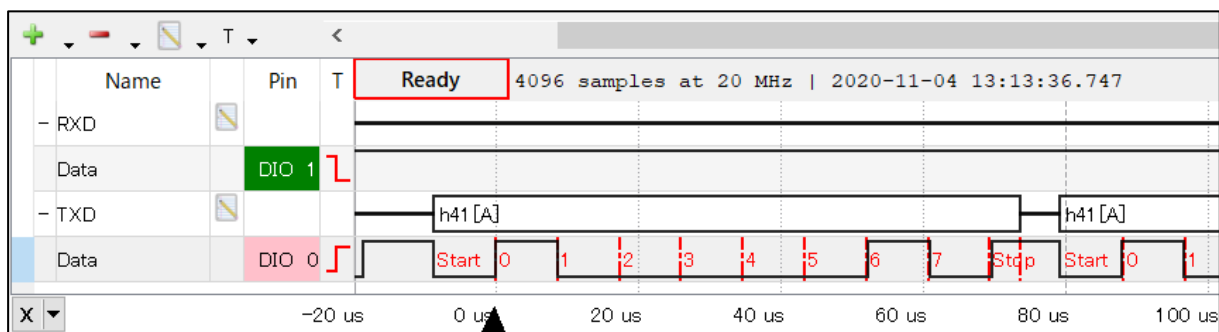
    printf("mini UART 送信テスト\n");
    serialFlush(fd);        //バッファのクリア ⑥
    for(;;){
        serialPutchar (fd, txdData); //1 文字の送信
    }
    serialClose(fd);
    return EXIT_SUCCESS;
}
```

- ① シリアル通信関連のヘッダファイルをインクルードします。
- ② 通信速度をマクロ定義します。
- ③ serialOpen 関数で通信速度を設定し、fd を取得します。
- ④ オープンに失敗した場合、エラーメッセージを表示して終了します。
- ⑤ 「データビット 8bit、パリティビット無効、ストップビット 1bit」の通信パラメータを設定します。
- ⑥ バッファをクリアします。
- ⑦ for 文で 1 バイトのデータ A を繰り返し送信します。

## 14.11.4. 送信の動作確認

リスト 14-2 をビルドして実行します。PC 側の Tera Term の画面に、アルファベットの A が図 14-28 のように表示されます。Digilent 社の Analog Discovery2 で測定したラズパイの拡張コネクタの送信信号 (TXD) を図 14-30 に示します。スタートビットは 1bit 分の「0」であることがわかります。また、データビットは 8bit で、最下位ビット (LSB) から出力されます。ストップビットは 1bit 分の「1」です。通信速度が 15,200bps のときの 1bit 幅は約  $8.68 \mu s$  です。通信データは 10bit で構成されるので、1 通信データは約  $86.8 \mu s$  と求まり、図 14-30 から確認できます。

図 14-30 mini UART が 'A' を送信したときの送信信号



### <動作しないときのトラブルシューティング>

- 「Raspberry Pi の設定」の「Serial Console」を無効にしましたか？(図 14-25)
- 本書のサンプルコードをダウンロードして、実行してみましょう。正常に動作した場合は、ユーザーのソースコードに不具合がある可能性があります。
- エラーメッセージが表示された場合、デバイス名 /dev/serial0 のスペルを確認します。serial(小文字のエル)0(数字のゼロ)です。
- PC からラズパイにログインできたらなら、回路やケーブルに問題はありますが、ブレッドボードに実装した部品や配線の浮き・接触不良、D サブコネクタで緩みがないか、確認してください。
- Tera Term の COM ポート番号と通信速度 (115200bps)を確認します。

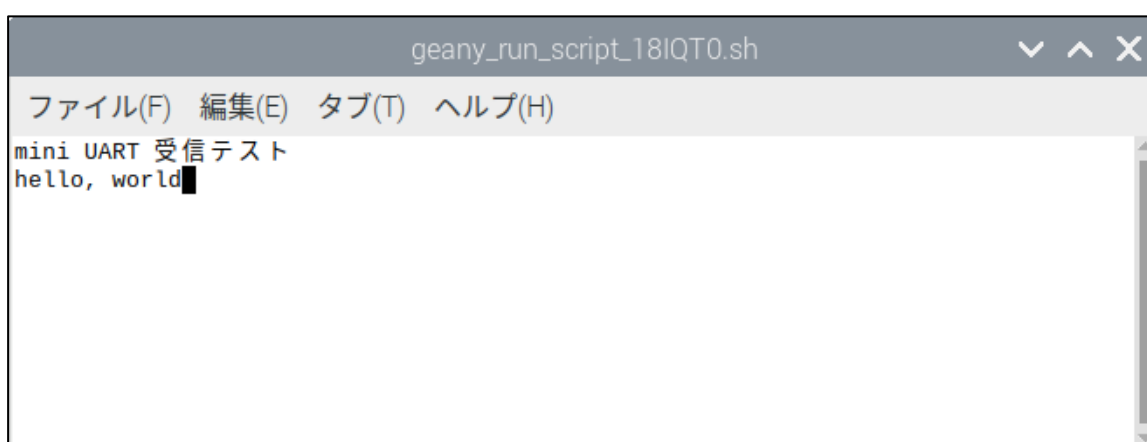
### 14.11.5. データを受信する

---

PC 側の Tera Term から、例えば「hello, world」とキーボードでタイプして送信します。ラズパイで受信データをターミナルに表示させるプログラムを作成します (図 14-31)。ファイル名は「List14-2\_Rxd01.c」とします。

- 条件
  - ・ 通信速度 115200bps、データビット 8bit、パリティビット無効、ストップビット 1bit

図 14-31 mini UART の受信画面



### 14.11.6. 受信のフローチャート

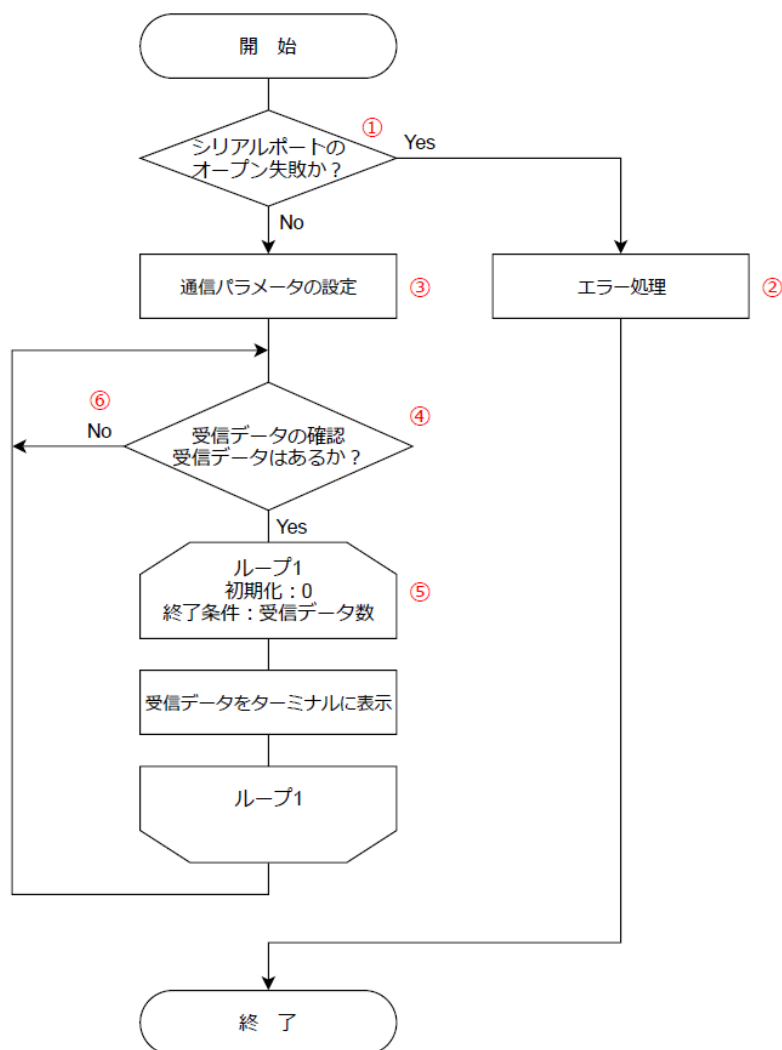
---

受信したデータをターミナルに表示するフローチャートを図 14-32 に示します。

- ① シリアルポートをオープンします。
- ② オープンに失敗した場合、エラー処理をして終了します。
- ③ データビット、パリティビット、ストップビットの通信パラメータを設定します。
- ④ 受信データを確認します。
- ⑤ 受信データがあれば、「Yes」で受信データをターミナルに表示します。
- ⑥ 受信データが無ければ、「No」で④のデータ受信の確認へ戻ります。



図 14-32 受信したデータをターミナルに表示するフローチャート



### 14.11.7. 受信のソースコード

ソースコード List14-2\_Rxd01.c をリスト 14-3 に示します。

リスト 14-3 List14-2\_Rxd01.c

```

#include <stdio.h>           //printf,etc
#include <stdlib.h>          //EXIT_FAILURE,etc
#include <termios.h>         //tcgetattr,etc
#include <unistd.h>          //usleep
#include <wiringSerial.h>    //serialOpen,etc

#define BPS 115200          //通信速度

int main (void){
    int i;
    int rxdData;             //受信データを格納する変数

```

```

int c;                //受信データ数
int fd;
struct termios param; //構造体変数

fd = serialOpen( "/dev/serial0", BPS); //シリアルポートのオープン ①
if( fd == -1){
    fprintf(stderr,"Error serialOpen.\n");
    exit(EXIT_FAILURE);
}
tcgetattr(fd, &param); //パラメータの取得
param.c_cflag &= ~CSIZE; //データビットのクリア
param.c_cflag |= CS8; //データビットを 8bit にセット
param.c_cflag &= ~PARENB; //パリティビットを無効にセット
param.c_cflag &= ~CSTOPB; //ストップビットを 1bit にセット
tcsetattr(fd, TCSANOW, &param); //パラメータの有効化

printf("mini UART 受信テスト\n");
serialFlush(fd); //バッファをクリア ④
while(1){
    c = serialDataAvail(fd); //受信データ数の取得 ⑤
    if(c>0){
        for (i=0;i<c;i++){
            rxdData = serialGetchar(fd); // 文字の受信
            //serialPutchar(fd,rxdData); //エコーバック
            printf("%c",rxdData); fflush(stdout); //文字の表示
        }
    }else{
        usleep(1000); ⑦ //CPU 使用率の抑制のため
    }
}
serialClose(fd);
return EXIT_SUCCESS;
}

```

- ① serialOpen 関数で通信速度を設定し、fd を取得します。
- ② オープンに失敗した場合、エラーメッセージを表示して終了します。
- ③ 「データビット 8bit、パリティビット無効、ストップビット 1bit」の通信パラメータを設定します。
- ④ バッファをクリアします。
- ⑤ 受信データ数を取得します。0 のときは受信データはありません。
- ⑥ 受信データがあれば、for 文で受信データをターミナルに表示します。受信データが無ければ、⑤へ戻り受信データ数を取得します。なお、printf 関数の引数である文字は一時的にバッファ<sup>7</sup>に保存されます。バッファの文字は通常改行コードが書かれたときだけ、またはバッファが一杯のときだけ転送されます。“%c”には改行コードが含まれていないため、直ぐには表示されません。そこで、fflush(stdout)を実行すると、バッファの文字を stdout(標準出力)に(強制的に)出力します。stdout と stderr(標準エラー出力)はディスプレイに結び付けられています。

<sup>7</sup> 異なるデバイス間でデータを転送するときに、処理速度や転送速度の差を緩衝するために一時的にデータを保存するメモリ領域のこと。

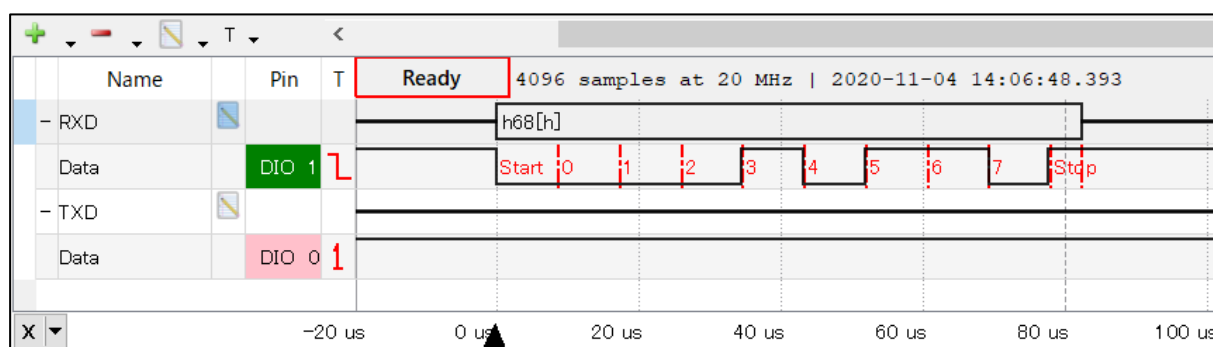
- ⑦ `usleep` 関数は  $\mu\text{s}$  単位で値を設定できます。ここでは、`usleep` 関数で `1ms` の時間待ちを挿入して、ループ回数を減らして CPU 使用率を抑制します。

### 14.11.8. 受信の動作確認

リスト 14-3 をビルドして実行します。PC 側の Tera Term から「hello, world」とタイプすると、ラズパイ側のターミナルに「hello, world」が表示されます (図 14-31)。Digilent 社の Analog Discovery2 で測定したラズパイの拡張コネクタの受信信号 (RXD) を図 14-33 に示します。図 14-33 は、「h」を受信したときの受信信号です。図 14-33 から、伝送データが連続しない場合、ストップビットを受信後にアイドル状態「1」になることが分かります。

なお、[Backspace]、[Delete]、[上下左右のカーソル]等のコントロールキーについては使用できません。

図 14-33 「h」を受信したときの受信信号



Column

## エコーバックとは

受信したデータをそのまま送り返すことをエコーバック (echoback) と言います。

## 14.12. パリティビットを付加してデータを伝送する

本節では、拡張コネクタのプライマリ UART を PL011 へ変更して、パリティビットを付加したシリアル通信を行います。ラズパイの Serial Port は「有効」、Serial Console は「無効」に設定してください（図 14-25）。

### 14.12.1. mini UART と PL011 の接続先を交換する

ラズパイ OS のデフォルトでは、mini UART がプライマリ UART として拡張コネクタに接続されています。しかし、通信相手の機器から、パリティビットを含めたデータフォーマットを要求された場合、mini UART ではパリティビットを処理する機能がないために通信エラーが発生して、通信が中断する恐れがあります。

そこで、パリティビットを処理できる PL011 をプライマリ UART（拡張コネクタ）に設定し、mini UART をセカンダリ UART（Bluetooth）へ、接続先を交換することができます。なお、mini UART により、Bluetooth の通信速度が低下する場合があります。

- ① config.txt を管理者権限で編集します。

```
$ sudo mousepad /boot/config.txt
```

- ② config.txt ファイルの末尾に、下記の 2 行を追加して保存します。miniuart-bt は、mini UART を Bluetooth に接続して、PL011 をプライマリ UART に設定します。また、mini UART のクロック信号の元となる VPU（Visual Processing Unit）のコア周波数が変動しないように 250MHz に固定します。

```
dtoverlay=miniuart-bt  
core_freq=250
```

- ③ OS を「再起動」すると mini UART と PL011 の接続先が入れ替わります（図 14-8）。

なお、mini UART をプライマリ UART に戻す場合は、上記 2 行を config.txt から削除またはコメントアウトして、再起動します。

ここでは、PL011 の通信パラメータをデータビット 7bit、奇数パリティ、ストップビット 2bit に設定します。そのため、Tera Term のシリアルポートの設定も図 14-34 のように変更します。なお、mini UART と同じ課題で送信と受信を確認しますので、フローチャートは図 14-29 と図 14-32 を参照してください。

図 14-34 Tera Term のシリアルポートの設定



### 14.12.2. データを送信する

ラズパイの PL011 から 1 文字「A」を繰り返し送信するプログラムを作成します。ラズパイと PC を RS-232 ケーブルで接続し、PC 側に Tera Term を使用して、ラズパイから受信したデータを表示させます（図 14-28）。ファイル名は「List14-3\_Txd02.c」とします。

- 条件

- 通信速度 115200bps、データビット 7bit、パリティビット奇数、ストップビット 2bit

### 14.12.3. 送信のソースコード

ソースコード List14-3\_Txd02.c をリスト 14-4 に示します。

リスト 14-4 List14-3\_Txd02.c

```
#include <stdio.h>           //printf,etc
#include <stdlib.h>           // EXIT_SUCCESS,etc
#include <termios.h>          //tcgetattr,etc
#include <wiringSerial.h>     //serialOpen,etc

#define BPS 115200           //通信速度

int main (void){
    char txData ='A';        //送信データ
    int fd;
    struct termios param;     //構造体変数
```

```

fd = serialOpen( "/dev/serial0", BPS); //シリアルポートのオープン ①
if( fd == -1){
    fprintf(stderr, "Error serialOpen.\n");
    exit(EXIT_FAILURE);
}
tcgetattr(fd, &param); //パラメータの取得
param.c_cflag &= ~CSIZE; //データビットのクリア
param.c_cflag |= CS7; //データビットを 7bit にセット
param.c_cflag |= PARENB; //パリティビットを有効にセット
param.c_cflag |= PARODD; //奇数パリティにセット
param.c_cflag |= CSTOPB; //ストップビットを 2bit にセット
tcsetattr(fd, TCSANOW, &param); //パラメータの有効化 ③

printf("送信テスト\n");
serialFlush(fd); //バッファのクリア ④
for(;;){
    serialPutchar (fd, txdData); //1 文字の送信 ⑤
}
serialClose(fd);
return EXIT_SUCCESS;
}

```

- ① serialOpen 関数で通信速度を設定し、fd を取得します。
- ② オープンに失敗した場合、エラーメッセージを表示して終了します。
- ③ 「データビット 7bit、パリティビット有効、奇数パリティ、ストップビット 2bit」の通信パラメータを設定します。
- ④ バッファをクリアします。
- ⑤ for 文で 1 バイトのデータ A を繰り返し送信します。

#### 14.12.4. 送信の動作確認

リスト 14-4 をビルドして実行します。アルファベットの A が PC 側の Tera Term の画面に表示されます (図 14-28)。測定したラズパイの拡張コネクタの送信信号 (TXD) を図 14-35 と図 14-36 に示します。「奇数パリティではデータビットの「1」の個数が奇数なら「0」、偶数なら「1」になる」 んでしたね。

図 14-35 は PL011 から 'A' を送信したときの送信信号です。図 14-35 より、データビットが 7bit であることが確認できます。'A' の「1」の個数は偶数なので、奇数パリティではパリティビットは 1 になります。ストップビットは 2bit あります。

次に、図 14-36 は PL011 から 'C' を送信したときの送信信号です。図 14-36 より、データビットが 7bit であることが確認できます。'C' の「1」の個数は奇数なので、奇数パリティではパリティビットは 0 になります。ストップビットは 2bit あります。

図 14-35 PL011 が 'A' を送信したときの送信信号

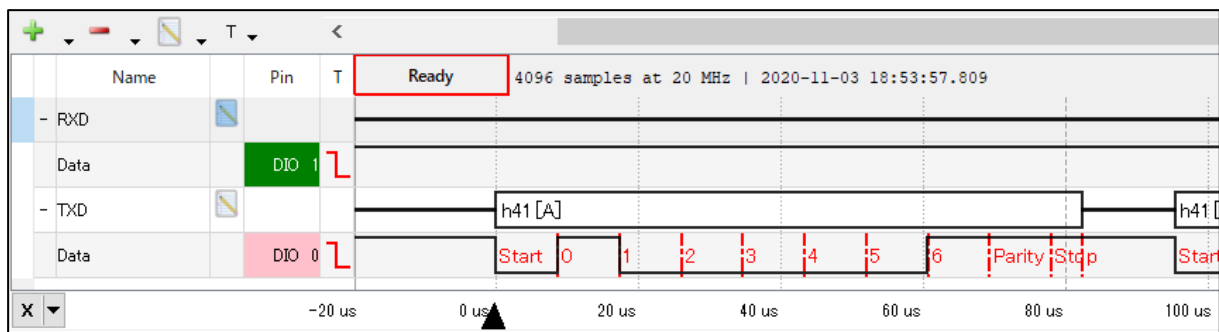
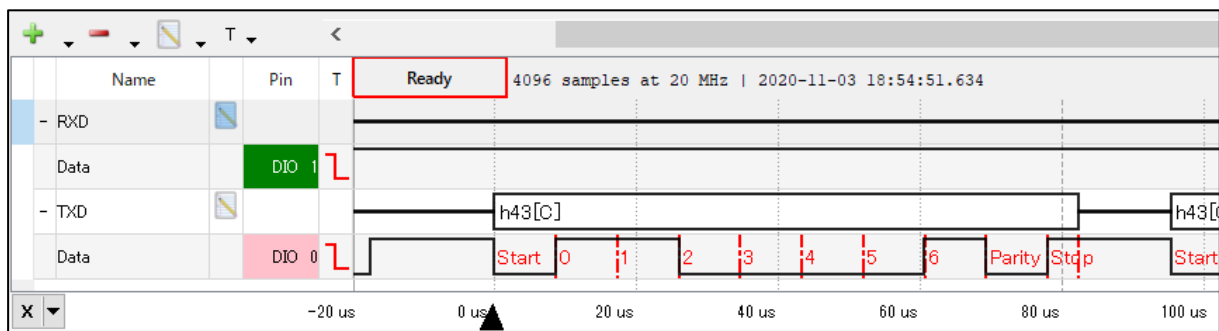


図 14-36 PL011 が 'C' を送信したときの送信信号



#### ＜動作しないときのトラブルシューティング＞

- 「/boot/config.txt」の末尾に下記の 2 行を追加して、再起動しましたか？  
dtoverlay=miniuart-bt  
core\_freq=250
- 追加した 2 行にスペルミスはありませんか？
- Tera Term のシリアルポートの設定を図 14-34 のように設定しましたか？

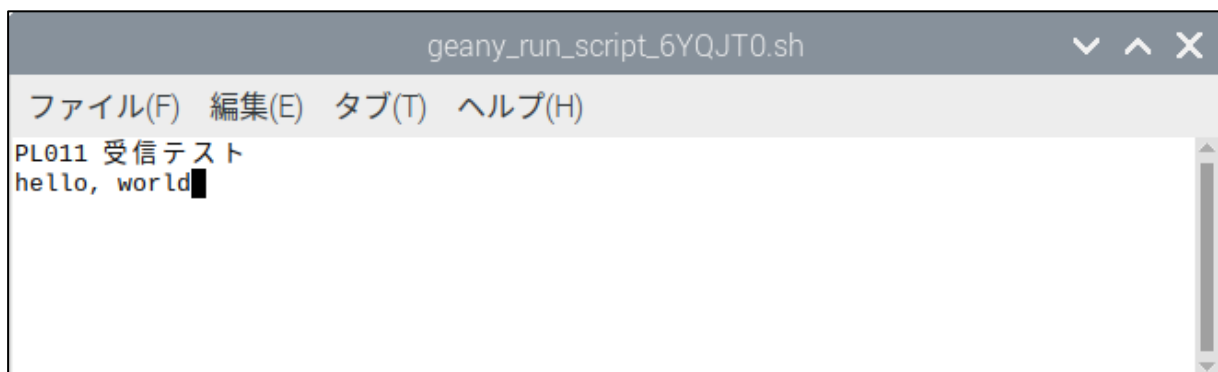
### 14.12.5. データを受信する

---

PC 側の Tera Term から、例えば「hello, world」とキーボードでタイプして送信します。ラズパイで受信データをターミナルに表示させるプログラムを作成します (図 14-37)。ファイル名は「List14-4\_Rxd02.c」とします。

- 条件
  - ・ 通信速度 115200bps、データビット 7bit、パリティビット奇数、ストップビット 2bit

図 14-37 PL011 の受信画面



### 14.12.6. 受信のソースコード

---

ソースコード List14-4\_Rxd02.c をリスト 14-5 に示します。

リスト 14-5 List14-4\_Rxd02.c

```
#include <stdio.h>           //printf,etc
#include <stdlib.h>          //EXIT_FAILURE,etc
#include <termios.h>         //tcgetattr,etc
#include <unistd.h>          //usleep
#include <wiringSerial.h>    //serialOpen,etc

#define BPS 115200           //通信速度

int main (void){
    int i;
    int rxdData;              //受信データを格納する変数
    int c;                   //受信データ数
    int fd;
    struct termios param;    //構造体変数
```



```

fd = serialOpen( "/dev/serial0", BPS); //シリアルポートのオープン ①
if( fd == -1){
    fprintf(stderr,"Error serialOpen.\n");
    exit(EXIT_FAILURE);
}
tcgetattr(fd, &param); //パラメータの取得
param.c_cflag &= ~CSIZE; //データビットのクリア
param.c_cflag |= CS7; //データビットを 7bit にセット
param.c_cflag |= PARENB; //パリティビットを有効にセット
param.c_cflag |= PARODD; //奇数パリティにセット
param.c_cflag |= CSTOPB; //ストップビットを 2bit にセット
tcsetattr(fd,TCSANOW,&param); //パラメータの有効化 ③

printf("PL011 受信テスト\n");
serialFlush(fd); //バッファをクリア ④
while(1){
    c = serialDataAvail(fd); //受信データ数の取得 ⑤
    if(c>0){
        for (i=0;i<c;i++){
            rxdData = serialGetchar(fd); // 文字の受信
            //serialPutchar(fd,rxdData); //エコーバック
            printf("%c",rxdData); fflush(stdout); //文字の表示
        }
    }else{
        usleep(1000); //CPU 使用率の抑制のため
    }
}
serialClose(fd);
return EXIT_SUCCESS;
}

```

- ① serialOpen 関数で通信速度を設定し、fd を取得します。
- ② オープンに失敗した場合、エラーメッセージを表示して終了します。
- ③ 「データビット 7bit、パリティビット有効、奇数パリティ、ストップビット 2bit」の通信パラメータを設定します。
- ④ バッファをクリアします。
- ⑤ 受信データ数を取得します。0 のときは受信データはありません。
- ⑥ 受信データがあれば、for 文で受信データをターミナルに表示します。受信データが無ければ、⑤へ戻り受信データ数を取得します。なお、usleep 関数で 1ms の時間待ちを挿入して、ループによる CPU 使用率を抑制します。

## 14.12.7. 受信の動作確認

リスト 14-5 をビルドして実行します。PC 側の Tera Term から「hello, world」とタイプすると、ラズパイ側のターミナルに「hello, world」が表示されます (図 14-37)。測定したラズパイの拡張コネクタの受信信号 (RXD) を図 14-38 と図 14-39 に示します。

図 14-38 は「h」を受信したときの受信信号です。「h」の「1」の個数は奇数なので、奇数パリティではパリティビットは0になります。

次に、図 14-39 は「e」を受信したときの受信信号です。「e」の「1」の個数は偶数なので、奇数パリティではパリティビットは1になります。

図 14-38 「h」を受信したときの受信信号

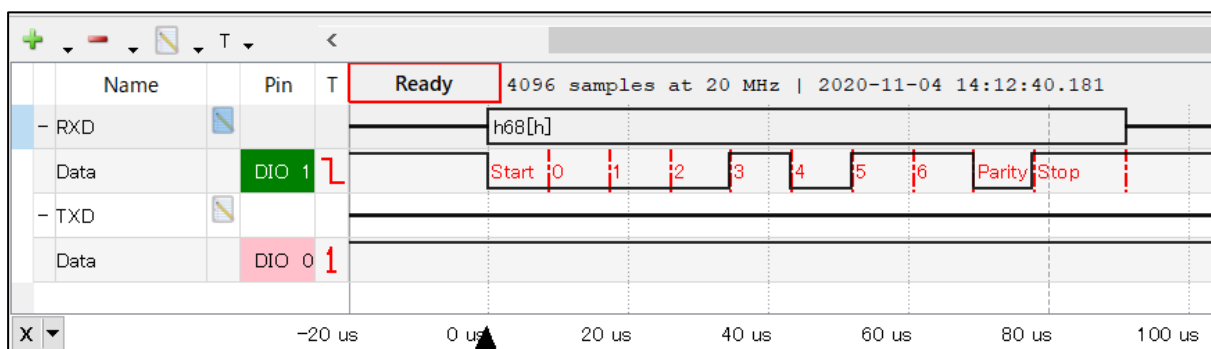
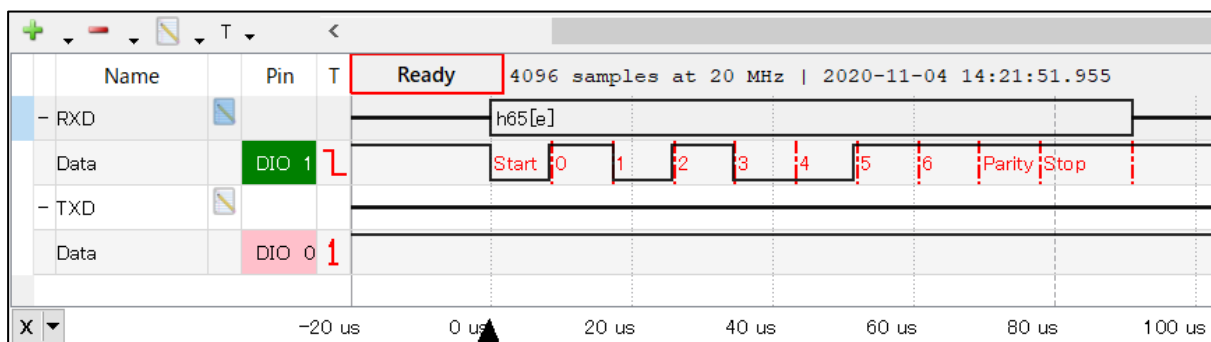


図 14-39 「e」を受信したときの受信信号



## 14.13. 章末問題

章末問題は、これまでの学習した内容を踏まえて応用力を習得するために用意しました。ここでは、図 14-12(b)で使用した RS-232 のインタフェース回路の配線図を使用します(図 14-40)。ラズパイと PC を RS-232 ケーブルで接続します(図 14-41)。ラズパイの Serial Port は「有効」、Serial Console は「無効」に設定してください(図 14-25)。

なお、章末問題ではプライマリ UART に mini UART を使用しますので、「14.11 パリティビットを付加してデータ伝送する」で config.txt を変更した場合は、追記した 2 行をコメントアウトし、上書き保存して再起動してください。また、Tera Term の「シリアルポートの設定」も、図 14-18 のように再設定してください。

```
#dtoverlay=miniuart-bt
#core_freq=250
```

章末問題の通信パラメータ等は次のとおりです。

- 条件
  - ・ 通信速度 115200bps、データビット 8bit、パリティビット無効、ストップビット 1bit
  - ・ プライマリ UART mini UART

図 14-40 章末問題の配線図

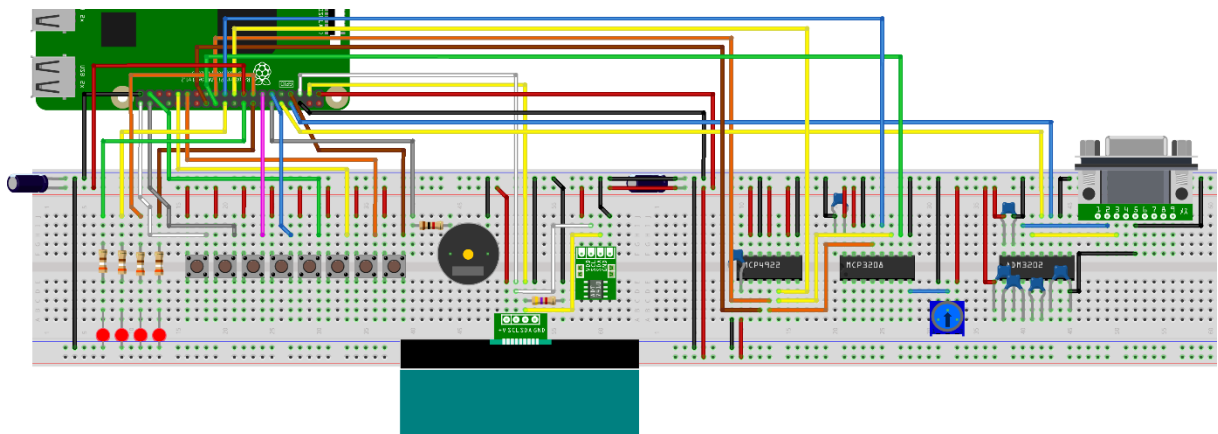
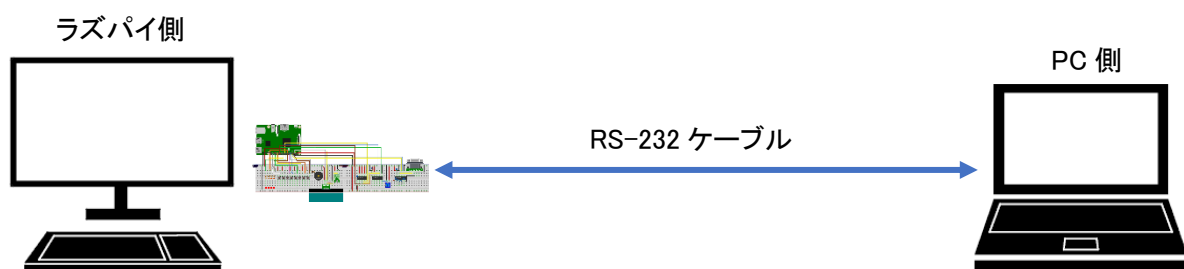


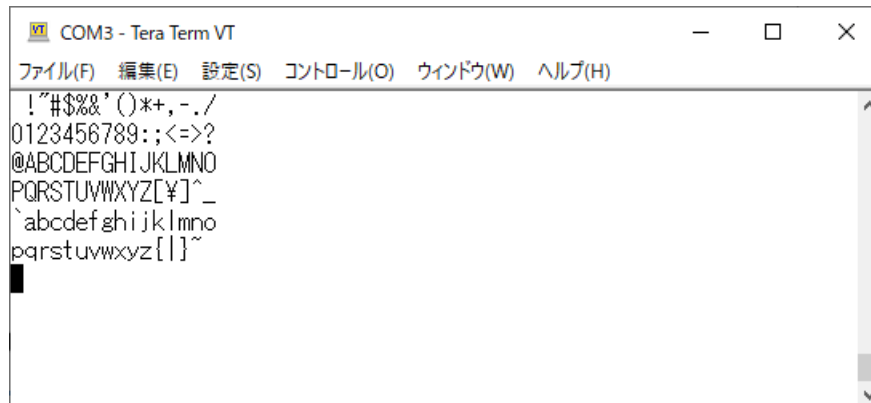
図 14-41 RS-232 ケーブルで接続したラズパイと PC



## 【問題 14-1】 アスキーコードを送信するプログラム

ラズパイから 0x20 から 0x7f までのアスキーコードを 16 文字毎に改行して PC 側へ送信します。PC 側の Tera Term では、図 14-42 のように表示させます。ファイル名は「List14\_Ex01.c」とします。

図 14-42 PC 側の Tera Term の表示画面



- ヒント
  - Linux と Windows では改行コードが異なります。Linux では改行コードは LF(\n) 1 バイトですが、Windows では CR+LF(\r\n) の 2 バイトとなりますので、ラズパイ側から CR+LF を送信します。

## 【問題 14-2】 SerialPuts 関数を使用して文字列を送信するプログラム

---

送信に serialPuts 関数だけを使用して、実行例のように hello, world に連番と改行を挟んで送信します。ファイル名は「List14\_Ex02.c」とします。

- 実行例

```
1 hello, world
2 hello, world
3 hello, world
. . . (略) . . .
```

- ヒント

- ・ 改行のエスケープシーケンスを文字列に含めます。  
"hello, world\r\n"

## 【問題 14-3】 SerialPrintf 関数を使用して文字列を送信するプログラム

---

今度は、送信に serialPrintf 関数だけを使用して【問題 14-2】と同様に、hello, world に連番と改行を付加して送信します。ファイル名は「List14\_Ex03.c」とします。

## 【問題 14-4】 スイッチデータを送信するプログラム

---

SW0～SW7 までのタクトイルスイッチの状態を 8bit のデータとして、10 進数で 100 ミリ秒毎に、PC 側へ送信します。スイッチが押されていないときは 0 で、押されたときは 1 とします。MSB は SW7 です。全てのスイッチを押した時は、データは 255 となります。ファイル名は「List14\_Ex04.c」とします。

### ● 実行例

```
SW data = 0
SW data = 1
SW data = 255
. . . (略) . . .
```

## 【問題 14-5】 温度データを送信するプログラム

---

温度センサ ADT7410 から 13bit フォーマットで温度を 0.5 秒ごとに取得し、PC 側へ送信します。温度のデータは小数点第一位まで表示します。ファイル名は「List14\_Ex05.c」とします。

### ● 実行例

```
23.9 C
24.6 C
25.1 C
25.5 C
. . . (略) . . .
```

## 【問題 14-6】 可変抵抗の電圧値を送信するプログラム

---

12bit A/D コンバータ MCP3208 の CH7 に、可変抵抗器 (50  $\Omega$ ) で 0V から 3.3V 範囲で任意の電圧を入力します。入力電圧を 0.5 秒ごと A/D 変換して、実行例のように 16 進数で ADC コードを、「徹底入門」式 8-5 (p. 188) を使用して、10 進数で電圧を PC 側へ送信します。ファイル名は「List14\_Ex06.c」とします。

### ● 実行例

CH7 = 5H	電圧 = 0.004 V
CH7 = 3B4H	電圧 = 0.764 V
CH7 = 7CAH	電圧 = 1.606 V
CH7 = 80FH	電圧 = 1.662 V
. . . (略) . . .	

## 【問題 14-7】 受信したデータを LED に表示するプログラム

---

PC 側に「0 から 15 までの値を入力してください>>>」と表示させます。PC のキーボードから 0 ~ 15 の数値を入力して「Enter」が押されたら、入力された数値が 2 進数でラズパイの LED に表示させます。また、0 ~ 15 以外が入力されたら「入力の値が範囲外です」と表示して、すべての LED を消灯します。ファイル名は「List14\_Ex07.c」とします。

### ● 実行例

#### LED の表示

0 から 15 までの値を入力してください>>>5

○●○●

0 から 15 までの値を入力してください>>>20

○○○○

入力の値が範囲外です

### ● ヒント

- PC 側で「Enter」を押したとき、ラズパイ側で CR(\r)を受信したらデータの区切りとします。
- 受信したデータの末尾に null を加えて、文字列にします。
- 文字列を数値に変換するには atoi 関数を使用します。

## 【問題 14-8】受信したデータを LCD に表示するプログラム

PC 側の Tera Term から受信したデータが 0x20 から 0x7f までの範囲とき、LCD (AQM1602) に文字パターンを表示します。[Enter]を受信したら、LcdNewLine (fd) 関数を実行して LCD を改行します。なお、0x20 から 0x7f 以外のデータが受信された場合、PC 側に「文字コードが範囲外です」と送信します。ファイル名は「List14\_Ex08.c」とします。

図 14-43 は、「hello, world[Enter]Raspberry Pi」と入力したときの表示例です。キーボードの「Esc」を押した場合は、PC 側の Tera Term に「文字コードが範囲外です」と表示されます (図 14-44)。また、キーボードの「Delete」を押した場合は、アスキーコードで 0x7f がラズパイへ送信され、LCD (AQM1602) では「←」が表示されます。「徹底入門」図 7-6: 文字パターンとアスキーコード (p. 142) を参照してください。

図 14-43 LCD の表示例

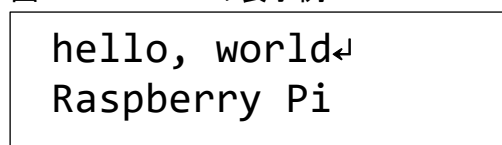
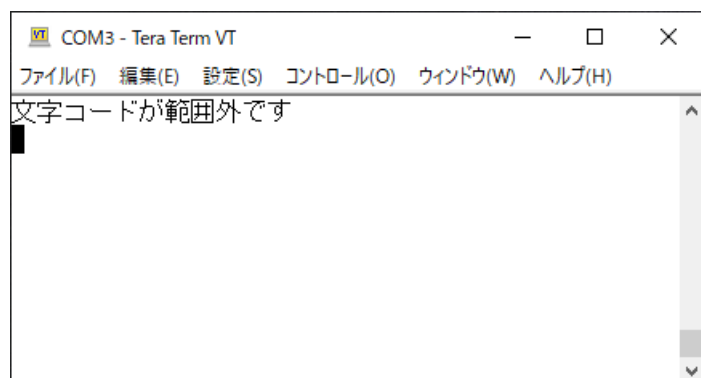


図 14-44 Tera Term に表示されるエラーメッセージ



- ヒント
  - ・ 「徹底入門」の Chapter 7 で、LCD 用関数をライブラリ(libMyPi.a, MyPi.h)に登録しました。LCD 用関数を利用すると便利です。LCD 用関数の詳細については「徹底入門」の Chapter 7 を参照してください。また、出版社の Web サイトからダウンロードできます。詳しくは、本章の「はじめに」を参照してください。
  - ・ ソースコードでヘッダファイル MyPi.h をインクルードします。  
#include "MyPi.h"



## 参考文献

- [1] 芹井滋喜, “RS-232 インターフェースの詳細と実例,” トランジスタ技術 2006 年 6 月号, CQ 出版社, pp.115-139, (June 2006)
- [2] “Interface Circuits for TIA/EIA-232-F,” Texas Instruments Design Notes SLLA037A, (Sep. 2002)
- [3] “RS-232 アプリケーションにおける電源要件について,” National Semiconductor Application Note 914, (Oct. 1993)
- [4] “データ通信用コネクタの標準ピン割り当て,” National Semiconductor Application Note 917, (Jul. 1998)
- [5] “BCM2835 ARM Peripherals,” Broadcom Corporation, (Feb. 2012)
- [6] “BCM2711 ARM Peripherals,” Raspberry Pi Trading Ltd, (Oct. 2020)
- [7] “ローパワー、+3.3V、RS-232 ライン・ドライバ/レシーバ ADM3202/ADM3222/ADM1385,” アナログ・デバイセズ, (May 2011)
- [8] 赤羽一馬, “チャージ・ポンプのしくみと低電圧動作の便利 IC,” トランジスタ技術 2008 年 6 月号, CQ 出版社, pp.151-154, (June 2008)
- [9] “エスケープシーケンス【escape sequence】エスケープ文字,” IT 用語辞典 e-Words, URL <https://e-words.jp/w/エスケープシーケンス.html> (2018.3.14 更新)
- [10] “制御文字,” ウィキペディア日本語版, URL <https://ja.wikipedia.org/wiki/制御文字> (2020 年 9 月 11 日 (金) 03:35 UTC)
- [11] “改行コード【newline code】改行文字,” IT 用語辞典 e-Words, URL <https://e-words.jp/w/改行コード.html> (2018.6.5 更新)

シーげんご ラズベリーパイ てっていにゅうもん  
C言語ではじめるRaspberry Pi 徹底入門

Chapter14 シリアルポートを使う

2020 年 11 月 18 日 初版

2021 年 3 月 20 日 第 2 版 受信用ソースコード改良

2021 年 5 月 13 日 第 2.1 版 GPIO 対応表、Tera Term 補足の追記

2023 年 4 月 1 日 第 2.2 版 しおり付き PDF

©2020 菊池達也