飛翔ソフトウェア

新人研修

マイコン制御編

|  |  |  |
| --- | --- | --- |
| 日付 | 変更内容 | 変更者 |
| 2021//7/15 | 新規作成 | 庭 |
|  |  |  |

# 研修の目的

車載Grの業務に必要な知識・技術を少しでも多く習得することで、配属先でスムーズに作業を行えるようにする。

# 使用する機器・ソフト

* Raspberry Pi 3 Model B
* RX62N搭載基板
* WindowsがインストールされたPC

-Tera Term

-秀丸エディタやさくらエディタなどのテキストエディタ

-ルネサス フラッシュ開発ツールキット（無料版）

-ルネサス 統合開発環境CS+ for CC（無料版）

-ルネサス RXファミリ用C/C++コンパイラパッケージ（無料版）

* USBケーブル(Type-A to micro-B)
* USBケーブル(Type-A to mini-B)

# Raspberry Pi 3 Model Bとは

ARMプロセッサを搭載したシングルボードコンピュータ。

主要スペック

|  |  |
| --- | --- |
| CPU | ARM Cortex-A53 (クアッドコア/ 1.2 GHz) |
| GPU | Broadcom VideoCore IV 400 MHz (3D 300 MHz) |
| メモリ | 1GB (LPDDR2) |
| ストレージ | SDカード |
| 通信機能 | 有線LAN：Gigabit Ethernet over USB 2.0 (maximum throughput  300Mbps)  無線LAN：IEEE 802.11 b/g/n 2.4 GHz  Bluetooth：Bluetooth 4.1, Bluetooth Low Energy |
| I/O | HDMI(Type A)、USB 2.0(A端子×4)、GPIO40ピン |

# RX62Nとは

ルネサス エレクトロニクス株式会社によって開発された32ビットCISCマイコン。メモリ容量、CANコントローラ有無で数種類の型番が存在する。

|  |  |  |
| --- | --- | --- |
| CPU | | 最大動作周波数：100MHz  32ビットRX CPU |
| メモリ | ROM | 容量：最大512Kバイト  今回使用する型番は**384Kバイト** |
| RAM | RAM容量：最大96Kバイト  今回使用する型番は**64Kバイト** |
| データフラッシュ | ROM容量：32Kバイト |
| 通信機能 | | イーサネットコントローラ  USB2.0ホスト／ファンクションモジュール  シリアルコミュニケーションインタフェース  I2Cバスインタフェース  CANモジュール  SPI |

# CANとは

CAN とは Controller Area Network の略称(以下 CAN という)で、1986年ドイツの電装メーカBOSCH社が自動車向けの通信プロトコルとして開発。その後、ISO11898およびISO11519で規格化され、現在、欧州では自動車LANの標準プロトコルに位置付けられている。

# 機材構成図

今回の大まかな構成図を記載する。

PCから無線LANを経由してRaspberry Piへ接続し、CANを使ってRX62Nマイコンと通信を行う。CAN通信でRX62N基板に実装されているLEDの点灯・消灯の要求を行えるようにする。

WindowsPC

RX62N

Raspberry Pi

TeraTerm

wifiルーター

CAN通信

LED

# 回路図

CANトランシーバのVDDはマイコン側と合わせて5Vを使用。

RXラインは

(CANコントローラ)3.3V←5V(CANトランシーバ)

となり電流の回り込みを防ぐため分圧する

(33 / ((10+10)+33)) × 5 ≒ 3.11V

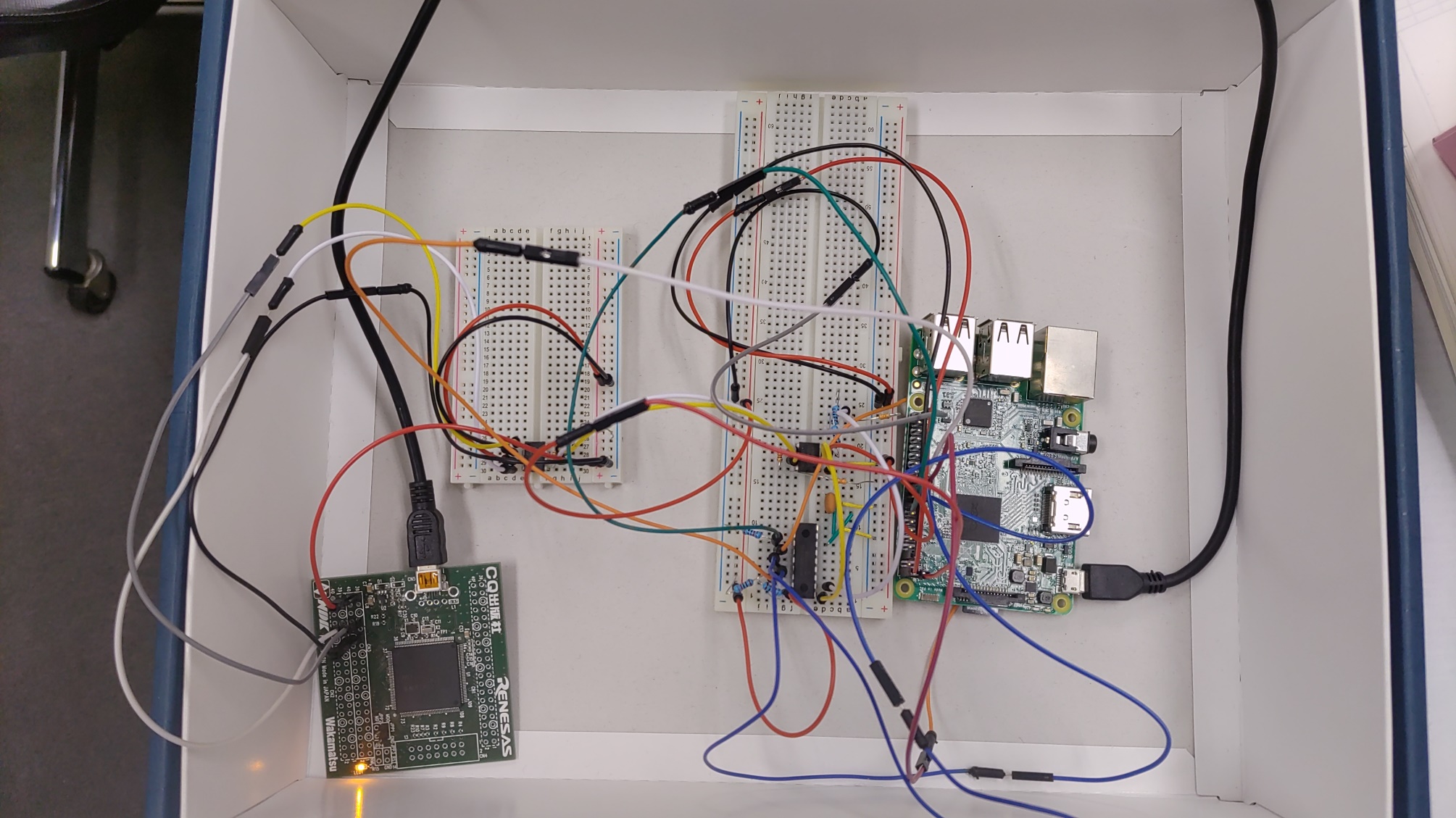
ラズパイとCANコントローラはSPI接続。ラズパイのGPIOの電圧が3.3Vなので、VDDには3.3Vを使用。

CANコントローラのRESETラインは3.3Vラインでpullupしておく（一定時間Lにすることでハードリセットできるが、ソフトでのリセットで十分なため常時Hとする）

CS、INTラインはGPIOの設定でpullup可能だが、今回は外部抵抗を使用。

トランシーバはISO-11898-5に準拠しているのでHigh Speed CANの接続にする

実物の写真

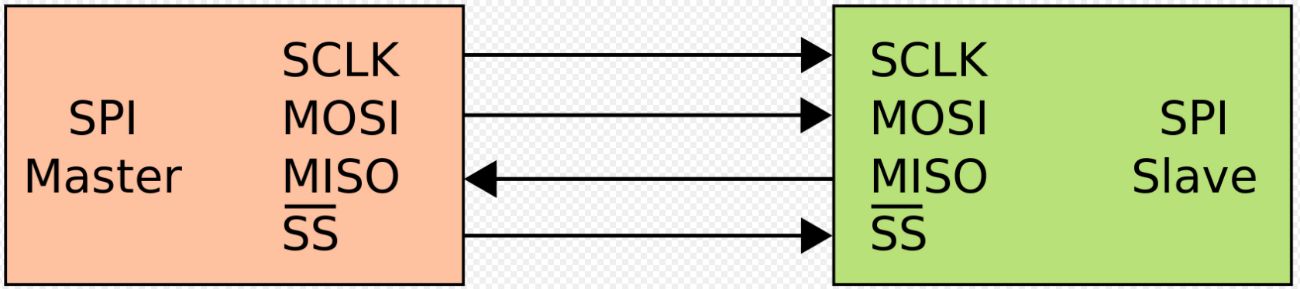


# SPI通信とは

シリアル・ペリフェラル・インタフェース(Serial Peripheral Interface, SPI)。

コンピュータ内部で使われるデバイス同士を接続するバスである。パラレルバスに比べて接続端子数が少なくて済むシリアルバスの一種で、比較的低速なデータ転送を行うデバイスに利用される。

wikipediaより抜粋



SCK：Serial Clock

MISO：Master In Slave Out

MOSI：Master Out Slave In

SS：Slave Select

# RX62N側の作業

## 準備

ルネサスのホームページから以下のツールを入手し、インストール

フラッシュ開発ツールキット（無料版）

統合開発環境CS+ for CC（無料版）

RXファミリ用C/C++コンパイラパッケージ（無料版）

## ソフトウェア作成

### CANメッセージ定義

今回の実験で使用するCANメッセージの仕様を定義する

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | 方向 | フレームタイプ | データサイズ | 意味 |
| 0x282 | Raspi→RX62N | データ | 1byte | マイコンLED制御  1:点灯/0:消灯 |
| 0x3C3 | Raspi→RX62N | データ | 1～8byte | テストデータ送信  送られたテストデータをマイコン内のRAMへ展開する。RAMの初期値はALL0xFFとし、0xFF以外の値が設定され次第、1秒毎にカウントアップしてID：0x5A5で送信する |
| 0x5A5 | Raspi←RX62N | データ | 8byte | マイコン内のテストデータを1秒毎にカウントアップしながら値を送信する |
| 0x3C4 | Raspi→RX62N | リモート | - | テストデータの値がいくつか確認する |
| 0x3C4 | Raspi←RX62N | データ | 8byte | 現在のテストデータの値を返却する |

### アドレス空間

RX62Nのアドレス空間は0000 0000h 番地からFFFF FFFFh 番地までの4G バイト。メモリマップドI/Oが採用されているため、マッピングされたメモリアドレスへアクセスすることで各種レジスタ操作が出来る。

RENESAS　RX62Nグループ、RX621グループユーザーズマニュアル　ハードウェア編

より抜粋



### スタートアップ処理

ルネサスの統合開発環境でプロジェクトを作成すると、雛形として最低限のスタートアップ処理が記載されている。

RX62Nではリセットを含む割り込みに対しては、配置場所が決められている固定ベクタテーブルを利用している。リセット・ベクタは0xFFFFFFFCとなり、PowerON\_Reset\_PC関数が登録されている。

固定ベクタテーブル



PowerON\_Reset\_PC

main関数

### PowerON\_Reset\_PC

ASISの処理

スタック領域の設定、CPU内部レジスタの初期化、静的変数の初期化を行った後、main関数を実行する。

変更点

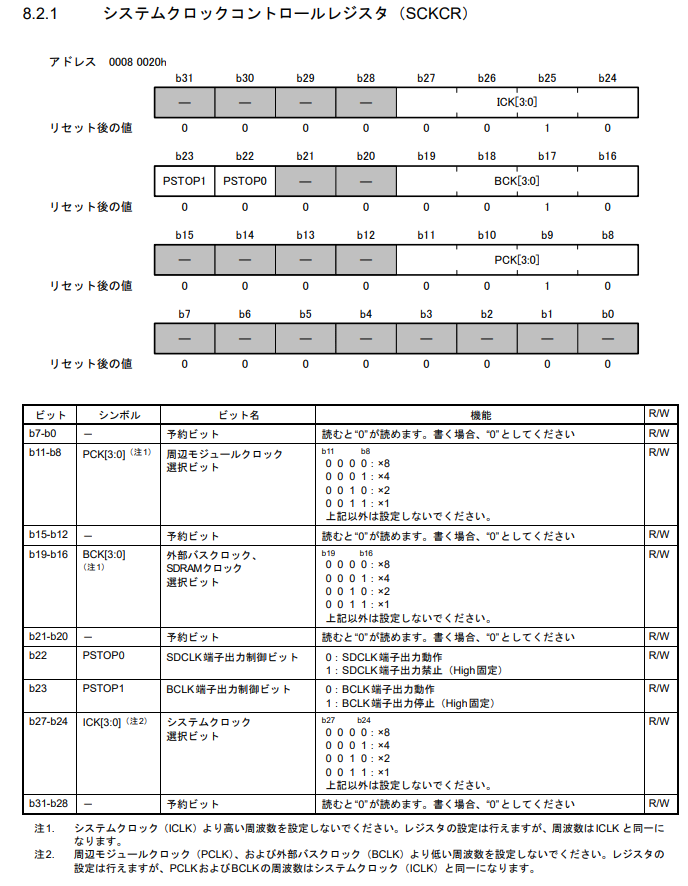
システムクロックの設定をハードの初期値から変更する。

CAN通信のクロックはPCLKが使用されるため、ICLK=96MHz,BCLK=24MHz,PCLK=12MHzにする。

SYSTEM.SCKCR.LONG = 0x20300;

RENESAS　RX62Nグループ、RX621グループユーザーズマニュアル　ハードウェア編

より抜粋



### main処理

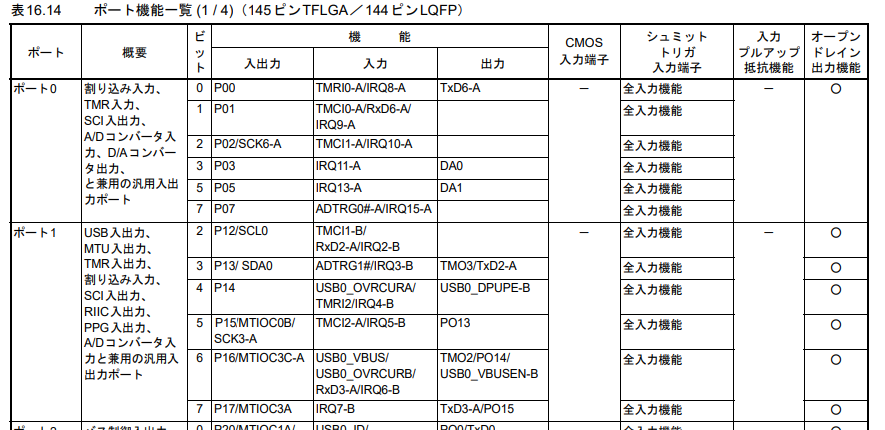
#### LED制御

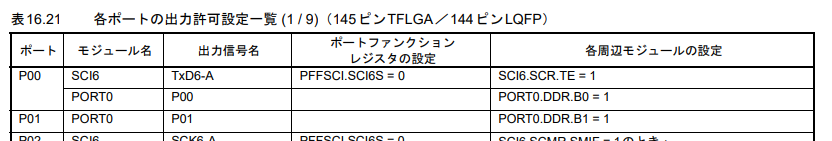
基板が付いていたインターフェース2011年5月号の情報によると、LEDは汎用I/Oポート15に接続されている。

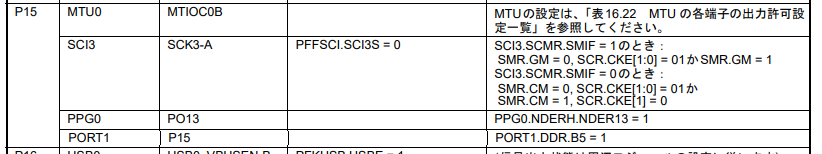
DDR（データディレクションレジスタ）で出力設定へ、DR（データレジスタ）で点灯状態を制御することができる。（LEDは負論理接続されているためLで点灯、Hで消灯）

RENESAS　RX62Nグループ、RX621グループユーザーズマニュアル　ハードウェア編

より抜粋







PORT1.DDR.BIT.B5=1 //出力設定

PORT1.DR.BIT.B5=1 //LED消灯

PORT1.DR.BIT.B5=0 //LED点灯

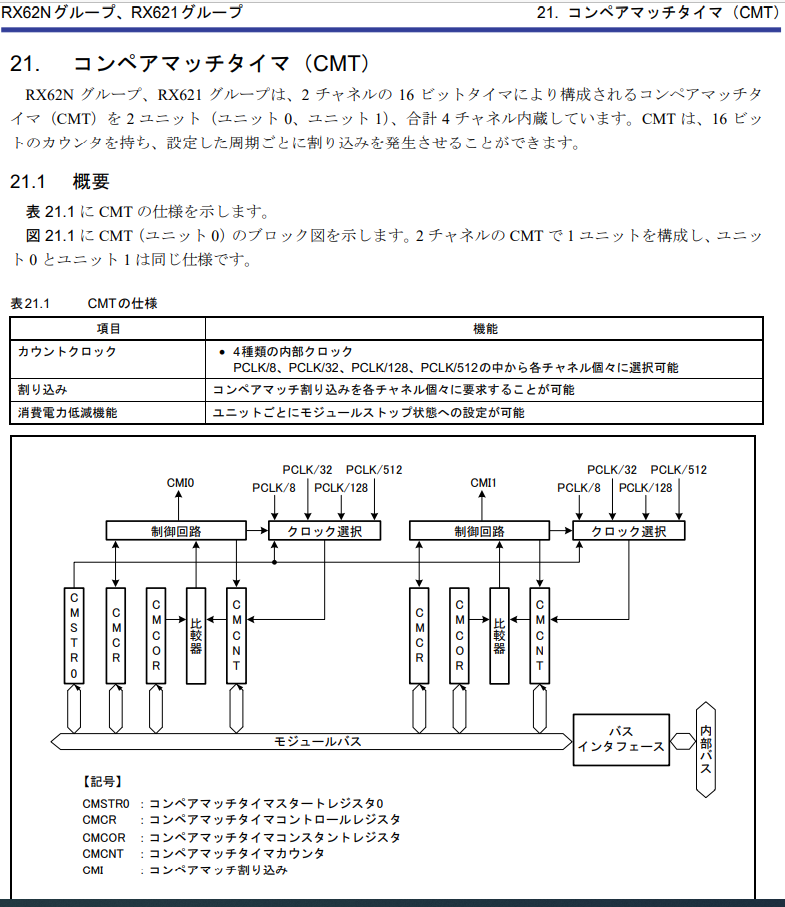
#### タイマ制御

タイマはコンペアマッチタイマ（CMT）を使用する。

コンペアマッチタイマはカウントアップで一定周期の時間経過を管理するタイマ。

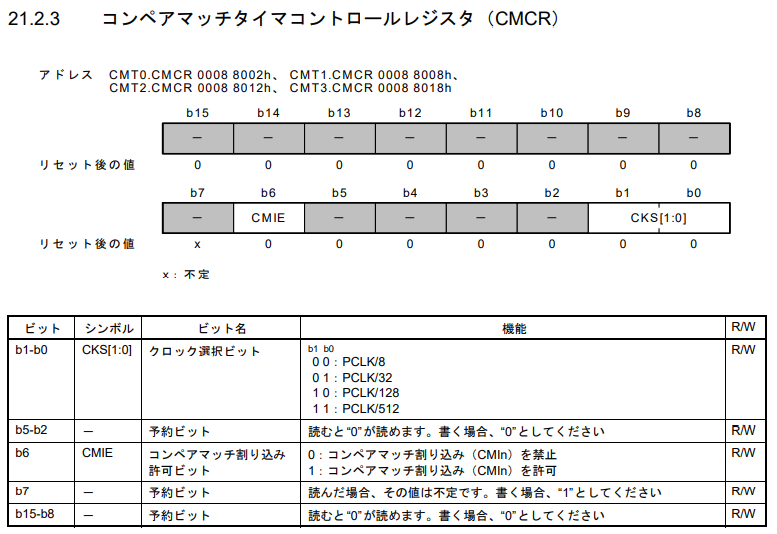
RENESAS　RX62Nグループ、RX621グループユーザーズマニュアル　ハードウェア編

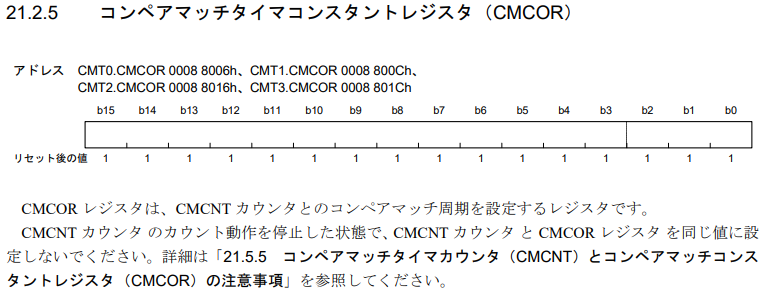
より抜粋



RENESAS　RX62Nグループ、RX621グループユーザーズマニュアル　ハードウェア編

より抜粋





今回は1秒の周期が必要なので、以下で使用する

CMT0.CMCR.BIT.CKS = 3; // PCLK/512

CMT0.CMCOR = 23437 - 1;

(PCLK = 12MHz、12000000 / 512 = 23437.5)

なお、今回は割り込みハンドラを登録せず、main関数で割り込みビットをポーリングしてチェックする方法とする。

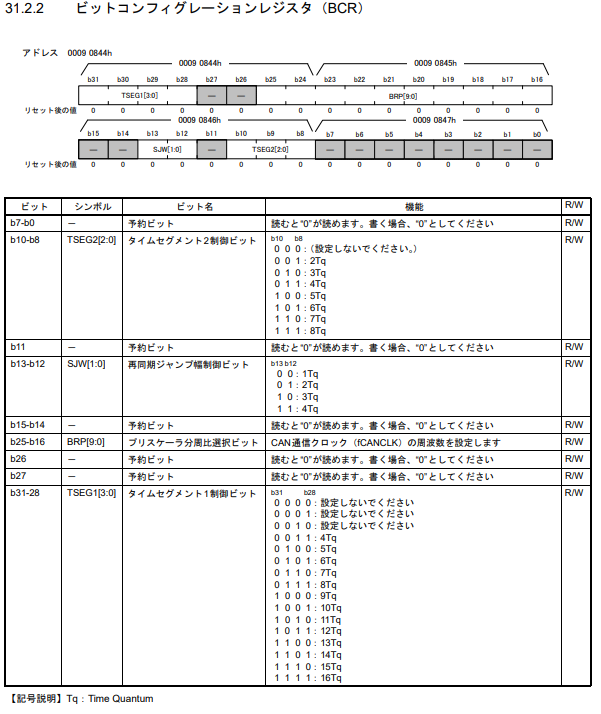
#### CAN制御

CAN通信の関数はインターフェース2011年6月号の付録から流用し、必要に応じて改造する。

ビットコンフィグレーションレジスタ（BCR）の設定

RENESAS　RX62Nグループ、RX621グループユーザーズマニュアル　ハードウェア編

より抜粋



今回は16TQに分割して、

Prop\_Seg＋Phase\_Seg1を9TQ

Phase\_Seg2を6TQ、SJWを4TQで設定する。うまく通信できない場合は要調整。

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Sync\_Seg | Prop\_Seg+ Phase\_Seg1 | | | | | | | | | Phase\_Seg2 | | | | | |

インターフェース2011年6月号によると、

CANクロック = PCLK / プリスケーラ分周比 / 1ビットタイム

の関係のようなので、

125kHz = 12MHz / プリスケーラ分周比 / 16

125000 = 12000000 / プリスケーラ分周比 / 16

プリスケーラ分周比= 6

BCRの設定で以下となる。

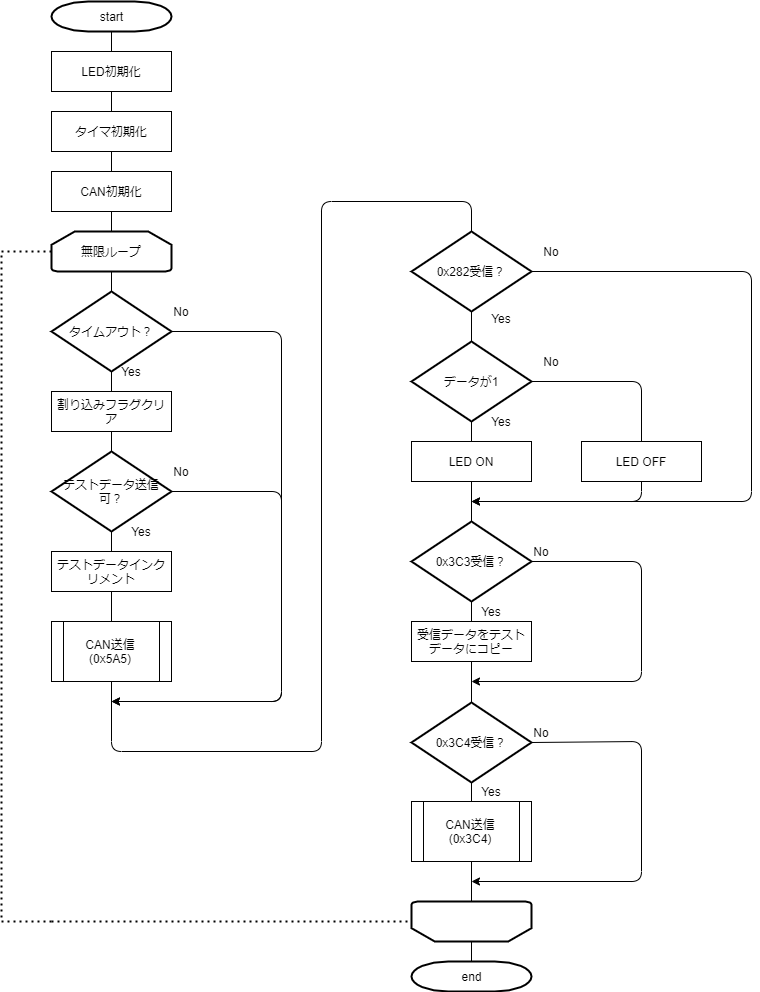
CAN0.BCR.BIT.BRP = 5; // プリスケーラ分周比 P+1=6

CAN0.BCR.BIT.TSEG1 = 8; // タイムセグメント 9TQ

CAN0.BCR.BIT.TSEG2 = 5; // タイムセグメント2 6TQ

CAN0.BCR.BIT.SJW = 3; // 再同期ジャンプ幅 4TQ

#### フローチャート



# Raspi側の作業

## ソフトウェア構成図

Raspberry Pi OS

can-utils

candump

cansend

mcp251xドライバ

SPIドライバ

ネットワークデバイス

ユーザースペース

カーネルスペース

ラズパイ側はmcp2515制御用のドライバを有効にし、既存のcan通信用ソフトウェアをインストールして使用する。

## 準備

1. Raspberry Pi OSをインストールする

<https://www.raspberrypi.org/software/>

具体的な手順は公式サイトやネット検索結果を参照。

1. mcp2515制御用のドライバを有効にする

sudo nano /boot/config.txt

以下の部分を編集

dtparam=spi=on

dtoverlay=mcp2515-can0,oscillator=12000000,interrupt=25

参考：<https://github.com/raspberrypi/firmware/blob/master/boot/overlays/README>

|  |
| --- |
| Name: mcp2515-can0  Info: Configures the MCP2515 CAN controller on spi0.0  Load: dtoverlay=mcp2515-can0,<param>=<val>  Params: oscillator Clock frequency for the CAN controller (Hz)  spimaxfrequency Maximum SPI frequence (Hz)  interrupt GPIO for interrupt signal |

spimaxfrequencyはdefaultで10MHz。今回はdefaultにしておく。

1. ユーザースペースからCAN通信を行うツール群can-utilsをインストール

sudo apt-get install can-utils

## 実験手順

1. ネットワークデバイスにCAN0を設定。bitrateは125kbpsとする

sudo ip link set can0 type can bitrate 125000

1. ネットワークデバイスの起動

sudo ip link set can0 up

1. テスト

受信データダンプ

candump -x -e can0

データ送信

LED点灯要求

cansend can0 282#01

→RX62NのLEDが点灯することを確認

LED消灯要求

cansend can0 282#00

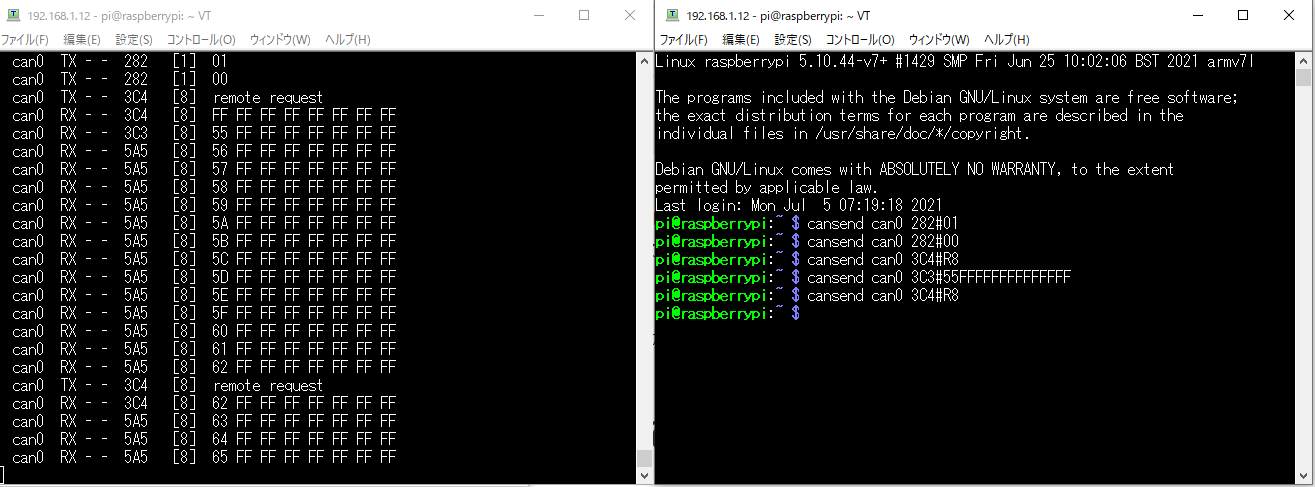
→RX62NのLEDが消灯することを確認

リモートフレーム送信

cansend can0 3C4#R8

テスト

cansend can0 3C3#55FFFFFFFFFFFFFF



# 参考資料

CAN関連

・RENESAS　CAN 入門書

・ベクタージャパン　はじめてのCAN/CAN FD

・30分で分かるCAN、設定とデザインのポイント

<https://eetimes.itmedia.co.jp/ee/articles/0912/15/news104.html>

・実装や試験で役立つ物理層から見るCANの仕組み

<https://monoist.atmarkit.co.jp/mn/articles/0809/10/news140.html>

RX62N関連

・RENESAS　RX62Nグループ/RX621グループ ユーザーズマニュアル ハードウェア編

・RENESAS　RXファミリ ユーザーズマニュアル ソフトウェア編

・インターフェース 2011年5月号

・インターフェース 2011年6月号

CANコントローラ、トランシーバ関連

・mcp2515\_j(MCP2515データシート　秋月電子の商品ページからダウンロード)

・mcp2561\_mcp2562\_j(MCP2561/2データシート　秋月電子の商品ページからダウンロード)

回路設計関連

・How to Connect Raspberry Pi to CAN Bus

<https://www.hackster.io/youness/how-to-connect-raspberry-pi-to-can-bus-b60235>