

# ライントレースカーによる 開発実習 <sup>マイコン基礎編</sup>

## 本テキストを使用する為に必要なモノ

- 実習機器
  - ・パソコン
  - PK-LTC
  - ・ドック
  - 基礎編の部品一式
  - 単三電池×2
  - オーバルコース(A3サイズ)

## 推奨する動作環境

- OS : Windows7/8/8.1/10
- CPU : Intel Core i5以上(2.0GHz以上を推奨)
- RAM :4GB以上
- •I/F : USBポート(3個以上)
- 画面 : HD1080(1920×1080)以上
- 使用するソフトウェア
  - LPCXpresso IDE V8.2.2
  - TeraTerm V4.98

#### マイコン基礎編の目的と目標

- 目的
  - ・マイコン制御の基本であるIO制御やAD変換などの知識・技術の習得
- 目標
  - C言語を用いてGPIOおよびAD変換制御ができるようになること
  - 習得した技術を活用してライントレース制御ができるようになること









## LPC1343マイコンの概要

- •LPC1343のスペック概要
  - CPU :ARM Cortex-M3 72MHz
  - Flash :32KB …今回は開発環境 からの制限で8KBまで使用可
  - SRAM :8KB
  - GPIO :42本(共用pinあり)
  - ADC :8本 分解能10bit
  - •通信 :UART,USB,I2Cほか

- VS-WRC103LVからの制約
  - 動作電圧は3.3V
  - プログラムの書き込みはUSB経由

- データシートのDescriptionやFeaturesを読 もう!
   マイコンの機能概要が書かれている ここを読
  - マイコンの機能概要が書かれている。ここを読めばどんなマイコンなのかザックリ分かる
    - 複数の型番のマイコンの事が1つのデータシー
       トに書かれている事が多いので注意すること



## 開発環境の構築・使い方

- VS-WRC103LV\_取扱説明書を合わせて使用しながら実施
  - 今回使用するIDEはLPCXpresso 8.2.2
- LPCXpressoのダウンロード
  - 共有フォルダからDLしてください
- LPCXpressoのインストール方法
  - 取説 P17 6-2 LPCXpressoのインストール を参照
    - インストールするだけでは8KB制限がかかってしまうが、今回の開発 規模ではプログラムのサイズが8KB以上超える事は稀であると予想される為、アクティベーションは行わない
- LPCXpressoの起動
  - 取説 P20 6-3 LPCX pressoの起動と認証
    - アクティベーションに関する作業は不要
- プロジェクトの作成・ビルド手順
  - 本テキストの次ページ以降を参照
- プログラムの書き込み手順
  - 取説 P33 6-5 VS-WRC103LVへのプログラムの書き込み方法 を参照



| terry 10.00.002 and the second file strategy   |  |                    |
|--|--|--------------------|
| um parte Meder heigen barte Build des Brens und -  |  |                    |
| C. DERAKAN CONTRACTOR OF THE   | 6+0+\$+\$+\$ # (#21), C #, # (#, *)  | (access) a lange   |
| Approximate of A. A. Sources, and an approximate of the second se | THE TT P COME COMPANY COUNTY AND A COMPANY COME COMPANY OF THE PARTY O |                    |
| i minoritano   | The second  |                    |
| A New WOLFSCH, Same and Add  |  |                    |
| al Assess  |  |                    |
| of technic   | Contraction of the second seco |                    |
| al m   |  |                    |
| 1. WHERE MADE AND A STREET AND  | The second  |                    |
| · Figend   |  |                    |
| < 8 million  |  |                    |
| 1.0.4.4750   |  |                    |
| E-mail   |  |                    |
| 11 TO 10   | and the second s |                    |
| to Terrary   | statigied among terms  |                    |
|  | The second  |                    |
| 1.0  |  |                    |
| and the second se  |  |                    |
| al house   |  |                    |
| of latence.  |  |                    |
|  | The second  |                    |
| Contraction and the second   |  |                    |
|  | The state is the balance of the state of the state of the state of the state of the  |                    |
| and the second se  |  |                    |
|  |  |                    |
| - Designed and the second s  |  |                    |
| 1.0.00000000   | and the intervention of the second states and the second states of the second states and |                    |
| D DAVE   | and the second se  |                    |
| a mean   |  |                    |
| 1 W  |  |                    |
|  | <ul> <li>mid Ministeri Art Lond</li> </ul>   |                    |
| the second se  |  |                    |
| and the second se  | And a second sec |                    |
| ter ne   | and a second sec |                    |
| heri pandi   |  |                    |
| New address of the second s  | and the second se  |                    |
| And a second   | the second design of the second  |                    |
|  | and the second se  |                    |
| And the star were presented by the present   |  |                    |
| And the Million County David Service   | 10 NA  |                    |
| taken to shift the investor data in the set  | and  |                    |
|  |  |                    |
|  |  |                    |
| the state of the second st | 1.47   |                    |
| debut territory for  | 1  |                    |
|  | The second se  |                    |
|  | A CONTRACT OF THE OWNER OW  |                    |
|  | and the second sec   |                    |
| and shareholder and because  |  |                    |
|  | North Control Street Providence P | 1 日間 法法公司 医骨骨的 化丁丁 |
|  | and the second sec   |                    |
|  | 22.25 (C. 102) Incompany half if and galling has been been been been been been   |                    |
|  |  |                    |
|  | sense where the sense has been been been been been been been bee   |                    |
|  |  |                    |
|  | and the second se  |                    |
|  |  |                    |
|  |  |                    |
|  |  |                    |
|  |  |                    |
|  |  |                    |

#### プロジェクトの作成とビルド手順



#### ワークスペースの選択

- デスクトップより「LPCXpresso」を起動
- ワークスペースフォルダの変更
  - 変更先: C:¥WorkSpace¥ARM\_C
  - •【変更の理由】LPCXpressoのワークスペースはパスに日本語などの 全角文字が含まれると正常にビルドできない為

| CXpresso    | orkspace<br>stores your projects in a folder called a v | vorkspace. |        |
|-------------|---|------------|--------|
| hoose a w   | orkspace folder to use for this session.                |            |        |
| orkspace:   | C:¥WorkSpace¥ARM_C                                      | ~]         | Browse |
|             |   |            |        |
|             |   |            |        |
|             |   |            |        |
| ]Use this a | s the default and do not ask again                      |            |        |
|             |   | 1          |        |
|             |   | OK         | Cancel |

#### プロジェクトの新規作成1

#### ① QuickStart PanelよりNewProjectを選択

② NewProject…画面よりLPC1300/1500>LPC13xx>C Projectを選択

| Project_ 13 7, Pergine, 10 Reports & Synthesi, **  | 0 Welcome II  | = (                                |
|--|---|------------------------------------|
| e 11   | Ille:///C/mp/LPCXpresso_822_650/tpcxpresso/pages/registeredReeEdition.htm   |                                    |
|  | NP  |                                    |
|  | LPCXpresso IDE (Free Edition) is fully activated  |                                    |
|  | Produces to LPC/Ignorem IDE (From Bellins). The arthreast is non-fails authorized, and can be used for produc<br>(Free Edition), day in used to generate and downlast applications containing up to 25002 of come to at LPC<br>Predict. Decomposite | don. LPCIgravies<br>Diaget         |
|  | This UPCX present RC User Quells providing installation for using UPC2 present. This is one available from the<br>+ Redge-UPC2 present their Gastle   | te Hittp menu                      |
|  | Purfere product Apparentiation is proveled within the LPC/Ipmann EDE we the Help reserve<br>+ Help-Help Contants  |                                    |
|  | Resources   |                                    |
|  | The ADP commonly is serificitively lead for ADD MCU using, including LPC generate EC, and is an<br>remarked for petrogeneity and antice an ann of the ADP MCDs, tasks and branchs. So, pet connection<br>ADP connection.                            | forestimation<br>formed perior Hos |
| or Quice, 🕮 💷 Vanas, 🍬 Binas, 🏦 Quitta, 🥂 Espis, 🤎 | To view the forum, and to regardle to that you can part, nut INEP Common by Param.     Dank the following pages of ADP can for our MOP MCD software ettaken, 19 Ca, and internation   |                                    |
| E Start here E                                     | <ul> <li>LPCOses serve and secondare</li> <li>LPCOses serve the follow</li> <li>LPC FAGe</li> </ul>   |                                    |
| Mew project.                                       | <ul> <li>DEGameses IDE setunts.</li> </ul>  |                                    |
| a had all second as                                | Links   | 172.0                              |
| S Frid (1, 1)                                      | <ul> <li>Val be Exhaust Artists LPCApresso rebills.</li> </ul>  |                                    |
| Chean ( )  |   |                                    |
| Deming 1     Deming 1     Deming 1                 | Constant I furtheren B Mersony Construction Inco 22 WO Inco Confer 40 Press Mersone   | ment line)                         |
| C Edit " project artifical                         |   | 10.00                              |
| Duick Settings •                                   | No consoles to display at this time.  |                                    |
| S Depart projects to accive faust                  |   |                                    |
| DEport projects and references to archive (alph    |   |                                    |
| IN Internet successful former WHE Proceedings      |   |                                    |

| X                    |                            |         |        | × |
|----------------------|----------------------------|---------|--------|---|
| New project          |                            |         |        |   |
| LPC13xx -> C Project |                            |         |        |   |
| Wizard               |                            |         |        | ^ |
| > Generic-M          |                            |         |        |   |
| > LPC1100 / LPC120   | 0                          |         |        |   |
| ~ LPC1300 / LPC150   | 0                          |         |        |   |
| > LPC13xx (12bit     | ADC)                       |         |        |   |
| LPC13xx              |                            |         |        |   |
| LPCOpen - C          | Project                    |         |        |   |
| LPCOpen - C          | Static Library Project     |         |        |   |
| LPCOpen - C          | ++ Project                 |         |        |   |
| LPCOpen - C          | .++ static Library Project |         |        |   |
| C Project            | (2)                        |         |        |   |
| C Project (Se        | emihosted)                 |         |        | ~ |
| C Canal - 1 lb -     | n:+                        |         |        | Ť |
|                      |                            |         |        |   |
| (?)                  | Park Next                  | Theorem | Cancel |   |

# プロジェクトの新規作成2

③ Project nameに「test01」と入力

④ Target selectionよりLPC13xx>LPC1343を選択

⑤ CMSIS Core libraryより「CMSIS\_CORE\_LPC13xx」を選択後、importを押

| X                            |  |   |
|------------------------------|--|---|
| New project                  |  | New project   |
| LPC13xx -> C Project         |  | CMSIS Library Project Selection   |
| Project name: 3              | New project  | Select the CMSIS project that you want your new project to link against, or select 'None' |
| Use default location         | Select the target MCU  | The selected library project(s) must be present in this workspace. If                     |
| Location: C:¥WorkSpace¥ARM_C | 5-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0  | they are not, then click the 'Import' button to run the Import Wizard.                    |
|                              | Target selection   | Library project can be found within your install directory at '/<br>lpcxpresso/Examples/' |
|                              | NXP LPC1343           > LPC11xx           > LPC11xxLV           > LPC1200           ~ LPC13xx           LPC1311           LPC1311           LPC1311           LPC1313           LPC13130 | CMSIS Core library to link project to CMSIS_CORE_LPC13xx                                  |
| e Back Next >                | LPC1343<br>> LPC1343<br>> LPC13xx (12bit ADC)<br>> LPC15xx   | Cancel  |
|                              | ? < Back Next > Finish   | Cancel  |

# プロジェクトの新規作成3

⑥ Project archive(zip)よりBrowseボタンを押下

- 「C:¥nxp¥LPCXpresso 8.2.2\_650¥lpcxpresso¥Examples¥Legacy¥CMSIS\_CORE¥ CMSIS\_CORE\_ Latest.zip」を開いて、Next押下
- ⑦ ProjectよりCMSIS\_CORE\_LPC13xx(CMSIS\_CORE\_LPC13xx)を選択し、Finish
- ⑧ CMSIS DSP Library Project Selectionは「none」でNext

⑨ SWO Trace Output TraceはチェックしてNext

| -  |                            |  |  |
|--|----------------------------|--|--|
| M Import   |                            |  |  |
| Import<br>Select the examples archive file to import.  | C                          | N - C  |  |
| Projects are contained within archives (.zip) or are unpacked within a directory. Select your project archive or root directory and press <next>. On the next page, select those projects</next> | s you                      | New project<br>CMSIS DSP Library Project Selection   | New project  |
| wish to import, and press <finish>. Project archives for LPCOpen and 'legacy' examples are provided.  Project archive (zin)  6</finish>  | )                          | If required, select the CMSIS DSP library project that you want your new project to link a   | Serial Wire Output Trace   |
| Archive  | Browse                     | The selected library project must be present in this workspace. If it is not, then click the 'Import' button to run the Import Wizard. CMSIS | Enable SWO trace clock     When enabled, code is included in the startup code which will enable clocking for SWO trace,                                      |
| Project directory (unpacked)<br>Root directory   | Browse                     | DSP library projects can be found in your LPCXpresso install<br>directory at '/Ipcxpresso/Examples/CMSIS_DSPLIB'                             | which would otherwise be left disabled as the MCU comes out of RESET.<br>Note - your board support must also set up pinmuxing such that SWO is output on the |
| LPCOpen<br>LPCOpen is the recommended code base for Cortex-M based NXP LPC Microcontrollers.   |                            | CMSIS DSP Library to link project to None  | appropriate GPIO pin for your hardware.  |
| LPCXpresso includes the LPCOpen packages which can be imported directly by pressing t<br>button in the Project archive (zip) section, above, and navigating to the Examples/LPCOp                | he Browse<br>en directory. | For more information on the CMSIS DSP Library, visit <u>https://community.nxp.com/messa</u><br>630590.                                       |  |
| Alternatively, press the button below to Browse the nxp.com website for latest resources.<br>Browse LPCOpen resources on nxp.com   |                            |  |  |
|  |                            | ⑦ < <u>B</u> ack <u>N</u> ext > Einish C   |  |
| Sack Next > Finish   | Cancel                     |  | C < Back Next > Finish Cancel  |

#### プロジェクトの新規作成4 ⑩ Other optionsはincにチェックを入れてFinish ⑪ プロジェクトが完成 ■ Develop-Welcome page-LPCXpress

| X   |                                      |                |                 |        |         | ×  |
|---|--------------------------------------|----------------|-----------------|--------|---------|----|
| New project                                 |                                      |                |                 |        |         |    |
| Other options                               |                                      |                |                 |        |         |    |
| Code Read Protection                        | n (CRP)                              |                |                 |        |         |    |
| Enable linker supp                          | ort for CRP                          |                |                 |        |         |    |
| See NXP documentati<br>MCUS for more inform | on for your MCU at<br>nation on CRP. | http://www.nxp | com/pages/:LPC- | ARM-CO | ORTEX-N | 1- |
| Project Structure                           |                                      |                |                 |        |         |    |
| User source directory                       | SFC                                  |                |                 |        |         |    |
| Create 'inc' directo                        | ry and add to path                   | (10)           |                 |        |         |    |
| Compiler language di                        | ialect                               | -              |                 |        |         |    |
| C Dialect Default                           |                                      |                |                 |        |         | ~  |
|   |                                      |                |                 |        |         |    |
|   |                                      |                |                 |        |         |    |
|   |                                      |                |                 | _      |         |    |

| Project 22 🐕 Periphe 🕮 Registers 🕼 Symbol 🖱  | Welcome  Helcome  He | - 0       |
|--|---|-----------|
| <ul> <li>CMSIS_CORE_LPC13xx</li> <li>test01</li> <li>Includes</li> <li>inc</li> <li>s or_startup_lpc13xx.c</li> <li>c crp.c</li> <li>test01.c</li> </ul> | Image:                               | × 🔺       |
| Start here Import project(s) New project Build all projects [Debug] Suild it est01' [Debug] Clean 'test01' [Debug]                                       | Check the following pages on KXP, com for new XXP MCU software releases, FAQs, and information.   |           |
| Clean test01 [Debug]   |   |           |
| Debug testor (Debug)     Preminate, Build and Debug 'test01' [Debug]   | 🖸 Console 🕄 🕐 Problems 🕕 Memory 🚳 Instruction Trace 🖾 SWO Trace Config 🚥 Power Measurement Tool   | = 0       |
| Edit 'test01' project settings   |   | d 🛛 🕶 🔂 🕶 |
| @ Quick Settings ▼   | No consoles to display at this time.  |           |
| J Export projects to archive (zip)   |   |           |
| B Forest and and advances to making (sin)  |   |           |

\_

## binファイル生成の為の設定1

プロジェクトの作成は完了しましたが、実際にコードを開発するにあたって、もう1つ設定する項目があります。
通常、axfと呼ばれるデバッガ用のファイルしか生成されない。そこで、axfからbin ファイルを生成するコマンドを設定する

ProjectExplorerよりtest01プ
 ロジェクトを右クリック
 >Propertiesをクリック

| 🔀 Deve  | lop -       | test01/src/test01.c - LPCXpresso   | 5                               |  |            | ×       |
|---|-------------|--|---------------------------------|--|------------|---------|
| File Edit   | t So        | urce Refactor Navigate Sea   | rch Project R                   | in Window Help   |            |         |
| <b>13 • 1</b> 8   |             | 8 • <b>%</b> • 8 × 10 11 11  | N 🕩 🛍 🖷 🤋                       | · · · · · · · · · · · · · · · · · · ·  | 3 ×        | Develop |
| Projec  |             | 😓 Periph IIII Regist 😓 Svi   | mb 🖻 🗖                          | Welcome     Betst01.c □  |            |         |
| > 🐸 CN  | ISIS_C      | CORE_LPC13xx   | 8 🐞 🔻                           | 18 #define GPIO0_DIR (*(volatile unsigned int *)0x50008000)<br>19 #define GPIO0_DATA (*(volatile unsigned int *)0x50003FFC)  |            | ^ -     |
| v ⊮ tes   | 1           | New<br>Go Into<br>Open in New Window   | >                               | 20 // main関数<br>22@int main(void) {<br>23 unsigned int wait = 0; // 待ち時間用の変数   |            |         |
| )<br>)<br>)<br>@  |             | Copy<br>Paste<br>Delete<br>Source<br>Move<br>Rename                                    | Ctrl+C<br>Ctrl+V<br>Delete<br>> | 24<br>25 // ポートの入出力設定<br>26 GPIO0_DIR = 0x08;<br>27<br>28 // 無限ループ<br>29 while(1) {<br>30 GPIO0_DATA = 0x08; // LEDをOFF<br>31 GPIO0_DATA = 0: wait < 2000000; wait++); // 待ち         |            |         |
|   | in<br>La    | Import<br>Export   |                                 | 31         For(wait = 0; wait < 3000000; wait++);         // 得ら           32         GPIO0_DATA = 0x00;         // LEDをON           33         for(wait = 0; wait < 100000; wait++); |            | ~       |
| U Qui   | ٤           | Build Project<br>Clean Project<br>Refresh<br>Close Project<br>Close Unrelated Projects | F5                              | Console S  Problems  Memory  Instruction Trace  SWO Trace Config  Power Measurement Tool ♦ ♥ S   | <b>:</b> 8 | • • •   |
| Start Start   | t<br>4<br>5 | Build Configurations<br>Make Targets<br>Index  | ><br>><br>>                     | <pre>make -r all<br/>Building file:/src/test01.c<br/>Invoking: MCU C Compiler<br/>arm=none=eabi=gcc =DDEBUG =D_CODE RED =DCORE M3 =D_USE CMSIS=CMSIS_CORE_LEC13xx =L</pre>           | ) LP       | C130    |
| <ul> <li>Build</li> <li>Build</li> <li>Clear</li> </ul> | 1           | Run As<br>Debug As<br>Profile As   | ><br>><br>>                     | <pre>/src/test01.c: In function 'main':<br/>/src/test01.c:31:10: error: 'waita' undeclared (first use in this function)<br/>for(waita = 0; wait &lt; 3000000; wait++); // 嬬?縺。</pre> |            |         |
| 参 Debu<br>参 Termi<br>⑧ Edit 1                           | u<br>t      | Team<br>Compare With<br>Restore from Local History<br>Launch Configurations            | >                               | <pre>/src/test01.c:31:10: note: each undeclared identifier is reported only once for ea<br/>make: *** [src/test01.o] Error 1<br/>03:49:22 Build Finished (took 363ms)</pre>          | ich f      | uncti   |
| Quic<br>Quic<br>Expo<br>Expo                            | r<br>r ag   | Smart update<br>Utilities<br>Tools   | >                               |  |            | ~       |
| ***<br>> test0'   | 1           | Properties   | Alt+Enter                       | © NXP LPC1343 (test01)   |            | >       |

### binファイル生成の為の設定

Note - Co (no - A ( - Af

13 C/C++ Build > Settings J り Build steps > Post-Build steps > Editボタン押下

⑭ 既存の3行のうち、2~3行 目の冒頭にある#コメントを 削除

| Resource<br>Builders<br>C/C++ Bui<br>Build V<br>Environ<br>Loggin<br>MCU se<br>Setting<br>Tool Ch<br>> C/C++ Ge<br>Project Re<br>Run/Debu<br>C/C++ Ge<br>Run/Debu<br>C/C++ | ce<br>rs<br>Build<br>d Variables<br>ronment                            | Configuration: Debu                                       | ng [Active]                  |                         |                    | An a sub-                 |
|--|--|---|------------------------------|-------------------------|--------------------|---------------------------|
| st-build steps<br>st-build steps   | o variables<br>ronment   |   | by [Active]                  |                         | Ŷ                  | Manage Configurations     |
| <ul> <li>&gt; C/C++ Ge<br/>Project Re<br/>Run/Debu</li> <li>st-build steps</li> <li>st-build steps</li> <li>st the Windows command pr<br/>comment character (#) disab<br/>ter one command per line.</li> <li>ter editing, commands are co<br/>-none-eabi-size "\${BuildArtin<br/>m-none-eabi-objcopy -v -O<br/>ecksum -p \${TargetChip} -d 1</li> </ul>  | ging<br>J settings<br>ings<br>I Chain Editor                           | Tool Settings /*<br>Pre-build steps<br>Command:           | Build steps 🕊 Build          | Artifact 📓 Binary Parse | rs 🥹 Error Parsers |                           |
| t-build steps<br>mmands are executed by a l<br>t the Windows command pr<br>omment character (#) disab<br>er one command per line.<br>er editing, commands are co<br>none-eabi-size "\${BuildArtin<br>n-none-eabi-objcopy -v -O<br>ecksum -p \${TargetChip} -d  | General<br>References<br>ebug Settings                                 | Description:  |                              |                         |                    | Edit                      |
| t-build steps<br>mmands are executed by a l<br>t the Windows command pr<br>omment character (#) disab<br>er one command per line.<br>er editing, commands are co<br>none-eabi-size "\${BuildArtin<br>n-none-eabi-objcopy -v -O<br>:cksum -p \${TargetChip} -d  |  |   |                              |                         |                    | ~                         |
| t-build steps<br>mmands are executed by a l<br>t the Windows command pr<br>omment character (#) disab<br>er one command per line.<br>er editing, commands are co<br>none-eabi-size "\${BuildArtin<br>n-none-eabi-objcopy -v -O<br>ecksum -p \${TargetChip} -d  |  | Post-build steps<br>Command:                              |                              |                         |                    |                           |
| t-build steps<br>nmands are executed by a l<br>the Windows command pr<br>omment character (#) disab<br>er one command per line.<br>er editing, commands are co<br>none-eabi-size "\${BuildArtin<br>n-none-eabi-objcopy -v -O<br>teksum -p \${TargetChip} -d  |  | arm-none-eabi-siz   | e "\$(BuildArtifactFile      | Name}"; # arm-none-eabi | -objcopy -v -O bin | ary "\${BuildArtifactFi ~ |
| nmands are executed by a l<br>the Windows command pr<br>omment character (#) disab<br>er one command per line.<br>er editing, commands are co<br>none-eabi-size "\${BuildArtin<br>n-none-eabi-objcopy -v -O<br>cksum -p \${TargetChip} -d "  |  |   |                              | ×                       | 1                  | Edit                      |
| none-eabi-size "\${BuildArtii<br>n-none-eabi-objcopy -v -O<br>cksum -p \${TargetChip} -d '   | a Linux compa<br>processor).<br>sables ALL FOL<br>a.<br>e concatenatec | atible shell<br>LOWING COMMANDS<br>d with a '' separator. | S.                           |                         |                    | ~                         |
|  | rtifactFileNam<br>-O binary "\${Bu<br>-d "\${BuildArti                 | ne}"<br>uildArtifactFileName)"<br>factFileBaseName}.bir   | " "\${BuildArtifactFil<br>n" | BaseName).bin"          | 14)                |                           |
|  |  |   |                              |                         |                    |                           |
|  |  |   |                              | 2                       |                    |                           |

(13)

# 初めてのマイコンプログラミング

- 初めてのマイコンプログラミングと言えばLEDをチカチカと点滅させるのが定番です
  - 組込みにおける「Hello World」
     です
- test01.cに右のソースコード を打ち込んで動作を確認して みましょう

```
#ifdef __USE_CMSIS
#include "LPC13xx.h"
#endif
```

#include <cr\_section\_macros.h>

#### // レジスタ定義

#define GPIO0\_DIR (\*(volatile unsigned int \*)0x50008000)
#define GPIO0\_DATA (\*(volatile unsigned int \*)0x50003FFC)

```
// main関数
```

```
int main(void) {
    volatile unsigned int wait = 0;// 待ち時間用の変数
```

```
// ポートの入出力設定
GPIO0_DIR = 0x08;
```

```
// 無限ループ
while(1) {
    GPIO0_DATA = 0x08;// LEDをOFF
    for(wait = 0; wait < 1000000; wait++);// 待ち
    GPIO0_DATA = 0x00;// LEDをON
    for(wait = 0; wait < 1000000; wait++);// 待ち
}
return 0 ;</pre>
```

#### プロジェクトのビルド

- ProjectExplorerより test01プロジェクトを選 択
- メニューの
   Project>Build Project
   をクリック
  - これにより該当のプロ ジェクトのビルドがス タート

| 🔀 Develop - test01/src/test01.c - LPCXpresso  | - 0   | ×       |
|---|---|---------|
| ile Edit Source Refactor Navigate Search Project  | t Run Window Help   |         |
| G ▼ Lar (A)   (8) Open Project<br>Close Project   | 2. 0  | Develop |
| Projec 😫 🕌 Build All Ctrl+B   | Welcome     id test01.c      I  | - 0     |
| <ul> <li>See CMSIS_COR</li> <li>See CMSIS_COR</li> <li>See test01</li> <li>See test01&lt;</li></ul> | 18 #define GPI00_DIR (*(volatile unsigned int *)0x50008000)         19 #define GPI00_DATA (*(volatile unsigned int *)0x50003FFC)         20         21 // main関数         22=int main(void) {         23 unsigned int wait = 0; // 待均時間用の変数         24         25         26         27         28         29       while(1) {         30       GPI00_DATA = 0x08;         27         28         29       while(1) {         31       for(wait = 0; wait < 3000000; wait++); // 待ち |         |
|   |   | 2       |
|   | 🖸 Console 🛱 📓 Problems 🕕 Memory 🚳 Instruction Trace 📓 SWO Trace Config 📼 Power Measurement Tool   | - 0     |
| 🕑 Qui 🕄 🕪 Vari 🗞 Bre 🗄 Outl 🕅 Exp 🧮 I   | CDT Build Concole (tert01)  | - 🖬 -   |
|   | MFlash32: 1040 B 32 KB 3.17%  | ^       |
| Start here *  | RamLoc8: 0 GB 8 KB 0.00%  |         |
| Import project(s)   | Finished building target: test01.axf  |         |
| New project   | makeno-print-directory post-build   |         |
| Build all projects [Debug]  | Performing post-build steps   |         |
| Suild 'test01' [Debug]  | arm-none-eabi-size "test01.axf" ; arm-none-eabi-objcopy -v -O binary "test01.axf" "test0  | 1.bir   |
| 🖌 Clean 'test01' [Debug]  | text data DSS dec nex filename  |         |
| Debug 'test01' [Debug]  | copy from `test01.axf' [elf32-littlearm] to `test01.bin' [binary]   |         |
| Terminate, Build and Debug 'test01' [Debug]   | Written checksum 0xefffd2b2 at offset 0x1c in file test01.bin   |         |
| Edit 'test01' project settings  | Previous value 0xefffd2b2 at offset 0x1c in file test01.bin   |         |
| Quick Settings  |   |         |
| Export projects to archive (zip)  | 03:56:33 Build Finished (took 1s.336ms)   |         |
| Export projects and references to archive (20)  |   | ~       |
| William and anniantial from VSAI Description  | v c   | >       |
| Writable  | Smart Insert 31:17  |         |

#### ビルド正常終了と失敗事例

- ビルドが正常に終了すると、Consoleの左記のようなログが流れます。Binファイルが生成されて、Build Finishedの文字が見えれば成功です
- ビルドが失敗に終わると、問題点について赤くマークアップされる。

🖻 Console 😫 🔜 Problems 🕕 Memory 🛞 Instruction Trace 🛄 SWO Trace Config 📟 Power Measurement Tool CDT Build Console [test01] MFlash32: 1040 B 32 KB 3.17% RamLoc8: 0 GB 8 KB 0.00% Finished building target: test01.axf make -- no-print-directory post-build Performing post-build steps arm-none-eabi-size "test01.axf" ; arm-none-eabi-objcopy -v -O binary "test01.axf" "test01.bir text bss dec hex filename data 1040 0 0 1040 410 test01.axf copy from 'test01.axf' [elf32-littlearm] to 'test01.bin' [binary] Written checksum 0xefffd2b2 at offset 0x1c in file test01.bin Previous value 0xefffd2b2 at offset 0x1c in file test01.bin

03:56:33 Build Finished (took 1s.336ms)



| 🖻 Console 😫 👔 Problems 🔋 Memory 🦚 Instruction Trace 🔛 SWO Trace Config 🚥 Power Measurement Tool                           |
|---|
|   |
| CDT Build Console [test01]  |
| 03:49:22 **** Incremental Build of configuration Debug for project test01 ****<br>make -r all                             |
| Building file:/src/test01.c   |
| Invoking: MCU C Compiler  |
| arm-none-eabi-gcc -DDEBUG -DCODE_RED -DCORE_M3 -DUSE_CMSIS=CMSIS_CORE_LPC13xx -DLPC13xx/src/test01.c: In function 'main': |
| /src/test01.c:31:10: error: 'waita' undeclared (first use in this function)   |
| for(waita = 0; wait < 3000000; wait++); // 蠕?縺。<br>^  |
| /src/test01.c:31:10: note: each undeclared identifier is reported only once for each funct:                               |
| make: *** [src/test01.0] Error 1  |
| 03:49:22 Build Finished (took 363ms)  |

## プログラムの書き込み手順

取説 P33 6-5 VS-WRC103LVへのプログラムの書き込み方法を

#### ようこそ!組込みの世界へ

- 基板上のオレンジ色のLEDが点滅することを確認できれば正常 に動作した事になります。
- •ようこそ!組込みの世界へ!

#### 01:GPIOによるLED制御



#### はじめに

- 最初のサンプルプログラムは、マイコンの基本であるLED制御です。
- このLEDと次章で説明するボタンスイッチ、スライドスイッチ などは全てマイコン内蔵のGPIOと呼ばれるIOポートを利用し ています。
- •よって、ここではGPIOの基本的な使い方について学習します。
- GPIOは、ほとんどのマイコンに付属する最も基本的な機能になります。それゆえに、LEDをはじめ多くのハードウェアの制御に使用されています。



- GPIOによるLED回路
  - LED1:緑のLED
  - LED2:オレンジのLED
- シンク電流による点灯制御





- LEDに流れる電流値を計算
  - $V = R^*I \rightarrow 3.3[V] 2[V] = 136[\Omega] + I[A]$
  - I=(3.3-2)/136=9.55…[mA]
  - LPC1343のシンク電流の最大値
    - 1pinあたり:4mA以下
      - データシートのIOH=-4mAより
    - しかし、IOHS=45mAである為、非保障な がら最大で45mAまで可

#### 回路の簡単な説明

- 基板上には緑と橙(オレンジ)の2つのLEDが搭載されており、 緑LEDはPIO0\_3に、橙LEDはPIO0\_7に繋がっています。
- ここでは、シンク電流によるLED点灯回路が構成されており、 IOポートから'0'を出力するとLEDの両端に3.3Vの電位差が発生 し、LEDが点灯、逆に'1'を出力するとLEDの両端に電位差が生 じずLEDは消灯する仕組みになっています。
  - ・緑LEDを点灯させるには、\_\_\_\_\_から\_\_\_\_を出力する
    ・緑LEDを消灯させるには、\_\_\_\_\_から\_\_\_\_を出力する
    ・橙LEDを点灯させるには、\_\_\_\_\_から\_\_\_\_を出力する

#### GPIOとは

- GPIOはGeneral Purpose I/Oの略です。
- GPIOは外部に対して、'1'か'0'の信号を出力する事ができます。
- LPC1343マイコンには、12bitのポートが3セットと4bitのポートが1セット内蔵 されており、それぞれPIO0, PIO1, PIO2, PIO3と名前が付けられています。
- またポートをビットごとに表現するにはポート1のビット0の場合、PIO1\_0と表します。
- 同じLPC1343マイコンでも、ピン数の少ないHVQFN33パッケージの場合、PIO2 およびPIO3は1bitのポート割付になる為、注意が必要です。

| Table 146. GPIO configuration |         |                   |                   |                   |                           |                       |  |  |
|-------------------------------|---------|-------------------|-------------------|-------------------|---------------------------|-----------------------|--|--|
| Part                          | Package | GPIO port 0       | GPIO port 1       | GPIO port 2       | GPIO port 3               | Total<br>GPIO<br>pins |  |  |
| LPC1311,<br>LPC1311/01        | HVQFN33 | PIO0_0 to PIO0_11 | PIO1_0 to PIO1_11 | PIO2_0            | PIO3_2; PIO3_4;<br>PIO3_5 | 28                    |  |  |
| LPC1313,<br>LPC1313/01        | LQFP48  | PIO0_0 to PIO0_11 | PIO1_0 to PIO1_11 | PIO2_0 to PIO2_11 | PIO3_0 to PIO3_5          | 42                    |  |  |
| LPC1313                       | HVQFN33 | PIO0_0 to PIO0_11 | PIO1_0 to PIO1_11 | PIO2_0            | PIO3_2; PIO3_4;<br>PIO3_5 | 28                    |  |  |
| LPC1342                       | LQFP48  | PIO0_0 to PIO0_11 | PIO1_0 to PIO1_11 | PIO2_0 to PIO2_11 | PIO3_0 to PIO3_3          | 40                    |  |  |
|                               | HVQFN33 | PIO0_0 to PIO0_11 | PIO1_0 to PIO1_11 | PIO2_0            | PIO3_2                    | 26                    |  |  |
| LPC1343                       | LQFP48  | PIO0_0 to PIO0_11 | PIO1_0 to PIO1_11 | PIO2_0 to PIO2_11 | PIO3_0 to PIO3_3          | 40                    |  |  |
|                               | HVQFN33 | PIO0_0 to PIO0_11 | PIO1_0 to PIO1_11 | PIO2_0            | PIO3_2                    | 26                    |  |  |

#### GPIOを扱う時の注意点

- 注意点としては、これらの ポートのほとんどが他の機能 の端子と兼用になっている点 です。
- マイコンの回路設計をする際には、ADやPWMなど数の少ない有用な機能を優先的にセンサやアクチュエータに割り当てて、GPIOなどは後の方で残ったものを利用する傾向があります。



#### ユーザマニュアルの参照

• GPIOを制御するプログラムを作成する場合は、ユーザマニュア ルの「Chapter 9: LPC13xx General Purpose I/O (GPIO)」を参 照します

### PIOOのレジスタ構成

| 名称       | アドレス<br>オフセット | R/W        | 初期値  | [base address]  |
|----------|---------------|------------|------|---|
| GPIODIR  | 0x8000        | R/W<br>R/W | 0x00 | port 0: 0x5000 0000<br>port 1: 0x5001 0000<br>port 2: 0x5002 0000 |
| GPIODATA | 0x3FFC        |            | n/a  | port 3: 0x5003 0000   |

- ここでいうレジスタとは、汎用レジスタとは異なり、GPIOを制御する為の機能が割りついた専用のレジスタとなります。
- 他のADコンバータやPWMといった機能も、全て制御レジスタが用意されており、このレジスタに値を書き込んだり、逆に値を読み込むことで制御を行います。
- マイコンを制御する場合、レジスタの制御を覚える事は非常に重要なポイントとなります。
- GPIOの各レジスタは、base address + アドレスオフセットで算出が可能です。
  - PIOOのDIRレジスタのアドレス : 0x5000 0000 + 0x8000 = 0x5000 8000
  - PIOOのDATAレジスタのアドレス: 0x5000 0000 + 0x3FFC = 0x5000 3FFC

## GPIODIR (GPIO data direction register)

- 該当のポートの各ビット(各端子)が入力ポートであるか、出力 ポートであるかを設定します。
- •対応するビット、それぞれにバラバラの設定が可能
  - '0'の場合は入力ポート
  - '1'の場合は出力ポート
- •初期値は0x00である為、電源投入時は入力ポートの設定。

#### GPIODATA(GPIO data register)

- •出力ポートの場合
  - 該当のポートの本レジスタに値を書き込むことで、値によってビット が立ったポート端子PIOn\_0~11に出力されます。
- •入力ポートの場合
  - 該当のポートの本レジスタの値を読み込むことで、ポート端子の状態 を取得する事が出来ます。

例題:test01.c

#ifdef \_\_USE\_CMSIS
#include "LPC13xx.h"
#endif

#include <cr\_section\_macros.h>

#### // レジスタ定義

#define GPIO0\_DIR (\*(volatile unsigned int \*)0x50008000)
#define GPIO0\_DATA(\*(volatile unsigned int \*)0x50003FFC)

#### // main関数

```
int main(void) {
    volatile unsigned int wait = 0;// 待ち時間用の変数
    // ポートの入出力設定
    CPI00 PID 0:00:
```

```
GPIOO_DIR = 0x08;
```

```
// 無限ループ
```

```
while(1) {
    GPIO0_DATA = 0x08;// LEDをOFF
    for(wait = 0; wait < 1000000; wait++);// 待ち
    GPIO0_DATA = 0x00;// LEDをON
    for(wait = 0; wait < 1000000; wait++);// 待ち</pre>
```

```
return 0 ;
```

- GPIOの制御レジスタの定義
  - #define GPIO0\_DIR ····
  - #define GPIO\_DATA ···
- GPIOの制御レジスタへのアクセス
- → 制御レジスタに対して変数のよう にアクセスする
  - $GPIO0_DIR = 0x08;$
  - $GPIO_DATA = 0x08;$
- 待ち時間の作成→空 $\nu$ ープ
  - for(…); …何もしない処理
- volatile修飾子
  - コンパイラの最適化を防ぐ

#### CMSIS-COREの活用

- 今後、新たなレジスタを使用する度に、レジスタをdefineで定義する必要があります。レジスタの数は非常に多岐に渡り、 define文で定義するだけでも一苦労です。
- ARM Cortex-MシリーズにはCMSIS-COREと呼ばれる便利な規 格が存在する
  - #include<LPC13xx.h> の中身を見てみよう!

# CMSIS-COREを活用したLチカ

#### // CMSIS-COREを活用した場合

#ifdef \_\_USE\_CMSIS
#include "LPC13xx.h"
#endif

#include <cr\_section\_macros.h>

```
int main(void) {
    unsigned int wait = 0;
```

```
LPC_GPIOO->DIR = 0x08;// ポート初期化
```

```
while(1) {
  LPC_GPI00->DATA = 0x08;// LEDをOFF
  for(wait = 0; wait < 100000; wait++);// 待ち
  LPC_GPI00->DATA = 0x00;// LEDをON
  for(wait = 0; wait < 1000000; wait++);// 待ち
  }
  return 0 ;</pre>
```

- ファイル名:rei01\_01.c
- 制御レジスタを構造体を活用し、 ペリフェラル別に構造化してのア クセス
  - LPC\_GPIO0->DIR = 0x08;
  - LPC\_GPIO0->DATA = 0x08;
- LPC13xx.h内に全て記述済み

- ARM Cortex-M間での移植性を高 める為の仕組み
- 今後は、CMSIS-COREで定義され たレジスタを活用する形でコード を書いていく。



#### ファイル名は「kadai01\_xx.c」とすること。

- •課題01:緑LEDだけを点滅制御するプログラム
- 課題02: 橙LEDと緑LEDを同時に点滅制御するプログラム 尚、点滅周期は従来の2倍とする
- •課題03:橙LEDと緑LEDを**交互**に点滅制御するプログラム
- •課題04:LED制御を行う関数を作成
  - void init\_led(void);
  - void set\_led\_orange(unsigned char value);
    - 橙LEDの点灯制御を行う関数
    - 引数が0の時、消灯。1の時、点灯。
  - void set\_led\_green(unsigned char value);
    - 緑LEDの点灯制御を行う関数
    - 引数が0の時、消灯。1の時、点灯。

ヒント:ビット演算
## フルカラーLEDを制御しよう!



- CN14のPIO3\_0~PIO3\_2にフル カラーLEDを繋ぎ、点灯制御し てみましょう!
- 課題
  - 05:赤→緑→青の順で点灯するプロ グラムを作成
  - 06:白色に光るプログラムを作成
  - 07:紫→黄→水の順で点灯するプロ グラムを作成
  - 08:時間があれば…関数化
    - void init\_exled(void);
    - void set\_exled\_full\_color(unsigned char color);
      - 引数により…0:消灯、1:赤、2:緑、4: 青といった制御を行う



fritzing

# 実体配線図(LED無し)

#### 拡大図



. .

.

•••••

.

# 精密な点滅周期を作る

- rei01\_01.cなどをベースとして行ってください
- オシロスコープを用いて現在の点滅周期を測定する
- 空ループの回数を調整し、1msのwait関数を作成する
  - ファイル名:kadai01\_08.c





- 1ms以上の任意の時間のwaitする事ができる関数、wait\_ms()を作成せよ。 Wait時間は引数\*1msとする。
  - ファイル名:kadai01\_09.c

## 用語:デバイスドライバ

### • デバイスドライバとは

デバイスドライバ(略称:ドライバ、ドライバー、デバドラ)とは、画像ディスプレイモニター、 プリンターやイーサネットボード、拡張カードやその他周辺機器など、パソコンに接続されている ハードウェアなどをOSによって制御可能にするために用意された、ソフトウェアである。 【引用元:Wikipedia】 <u>https://ja.wikipedia.org/wiki/デバイスドライバ</u>



# 02:シリアル通信によるprintfデバッグ





- PCやスマホのアプリケーション開発とは異なり、マイコンには キーボードも無ければ液晶画面も存在しないものが大半である
- そんな中でも、簡単なデバッグ・動作確認を行う為に、シリア ル通信を活用し、マイコンに対してデータの入力/出力を行う 事が多い。
- 今回は、redlibとシリアル通信用のデバイスドライバを活用し、 stdio.hを活用できるようにする
  - つまり、printfやscanfが活用できる!



# USB-Serialセットアップ&動作確認





- 1. USB-Serialの3pin(TXD)と4pin(RXD)をショートさせる
- 2. USB-SerialとPCを接続
- 3. (ドライバ導入)
  - 既に導入済みの場合は不要
- 4. TeraTerm起動
  - シリアル設定はデフォルト
  - ターミナル設定よりLocalEcho:切
- 5. 何か文字を打ち込みターミナル上に表示されれば成功

#### これまでに作った回路に 新たに追加する形で実現すること

# 回路図&実装配線図



- PIO1\_6はRxDと兼用ピン
- PIO1\_7はTxDと兼用ピン



# RedLibとシリアルのデバドラ導入



- シリアル通信のソースコードを共有フォルダ からダウンロードする
  - uart.h, uart\_io.hをプロジェクトのincに入れる
  - uart.c, uart\_io.cをプロジェクトのsrcに入れる
- RedLibを使用する設定を行う
  - Quickstart -> Quick Settings -> Set library/header type -> redlib(semihost) を選択



#### 【RedLib】

CodeRed社が実装した標準Cライブラリ→printfやscanfを提供

【シリアルのデバドラ】 マイコンのメーカーや評価ボードのメーカーの提供コードを利用するか、独自実装する

【システムコール関数】 RedLibのprintfやscanfから呼び出される関数. シリアルのデバドラとRedLibの橋渡し。自身で実装する

# 例題:rei02\_01.c (ソースコード提供)

#ifdef \_\_USE\_CMSIS
#include "LPC13xx.h"
#endif

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <cr_section_macros.h>
#include "uart.h"
#include "uart io.h"
```

return uart getchar();

}

```
// システムコールの実装
int __sys_write(int iFileHandle, char *pcBuffer, int iLength)
{
    int n;
    for (n = 0; n < iLength; n++) {
        if (pcBuffer[n] == '¥n'){
            uart_putchar('¥r');
            }
            uart_putchar(pcBuffer[n]);
        }
    return iLength;
}
int sys readc(){</pre>
```

```
// main関数
```

```
int main(void) {
    char str[16];
    int a;
```

```
// シリアル通信の初期化
init_uart_io(9600);
```

```
printf("Hello World.¥n");
```

```
// scanfによる文字列の取得
printf("input your name:");
scanf("%s", str);
printf("-> %s¥n", str);
```

```
// scanfによる整数の取得
printf("input num:");
scanf("%d", &a);
printf("-> %d¥n", a);
```

```
printf("\n\n--- bye.\n");
```

```
return 0 ;
```

# 例題:rei02\_01.cの動作確認

#### TeraTermの表示される結果



#### TeraTermの設定

| Tera Term: Terminal setup  |                                  | ×        |  |  |  |
|--|----------------------------------|----------|--|--|--|
| Terminal size  | New-line                         | ОК       |  |  |  |
| 80 X 24  | Receive: LF 🗸                    |          |  |  |  |
| ⊠Term size = win size  | Transmit: CR+LF ~                | Cancel   |  |  |  |
| □ Auto window resize   |                                  |          |  |  |  |
| Terminal ID: VT100 ~   |                                  | Help     |  |  |  |
|  |                                  |          |  |  |  |
| Answerback: Answer |                                  |          |  |  |  |
| Kanji (receive) Kanji (transmit)   |                                  |          |  |  |  |
| UTF-8 V UTF-8  | I 🗸 Kanji-in:                    | ^[\$B ~  |  |  |  |
| 🗆 7bit katakana 🛛 🗆 7bit   | <mark>katakana</mark> Kanji-out: | ^[(B ~~~ |  |  |  |
| locale: japanese CodePage: 932   |                                  |          |  |  |  |

課題

### ファイル名は「kadai02\_xx.c」とすること。

- 1. printf()関数を用いて、自己紹介をするプログラムを作成してくだ さい
  - •[出力イメージ]

Name:Seigyo Taro Age:5 From:Kumamoto

- シリアル通信でPCより'g'が送られてきたら緑のLEDを点灯するプログラムを作成せよ
- 3. 'o'が送られてきたらオレンジのLEDを点灯する
- 4. 'q'が送られてきたらLEDを消灯する

## printfデバッグの注意点

- シリアル通信を活用したprintf関数は、他の制御(GPIOやADなど)と比べ非常に遅い。
- シリアル通信を多用すると、システムの動作に大きな影響が出る可能性がある。
  - デバッグ時には使用するがリリース時には、printfの処理を無効化する などの工夫が必要

```
#ifdef DEBUG
#define debug_printf(fmt, ...) printf(fmt, __VA_ARGS__)
#else
#define debug_printf(...)
#endif
```

### 03:GPIOによるスイッチ制御





- 前章ではGPIOの出力機能について学びました。この章では、GPIO による入力機能について学びます。
- GPIOの入力機能も出力機能と同様に汎用性の高い機能になります。
- 今回は、ボタンスイッチとスライドスイッチ、2つのスイッチを扱います。
  - ボタンスイッチについて
    - ボタンを押下中のみONとなり、導通するモーメンタリ方式と、ボタンを押した後に 指を離してもON状態が持続するオルタネイト方式の2つ種類があります。
    - モーメンタリ…リモコンやゲームコントローラ、インターフォンなど
    - オルタネイト…生産設備の非常停止ボタンやブラウン管TVなど
      - モーメンタリ方式のスイッチとマイコンを活用して、ソフトウェア的にオルタネイトにする事もあります
  - スライドスイッチについて
    - つまみ部分をずらす事でON/OFFを切り替えます。切替後はON/OFFの状態が常時続きますので、製品のモード設定などによく使われます。

## 回路図:VS-WRC-103LV



- GPIOによるスイッチ回路
- スイッチのON/OFFとPIO0\_1への信号
  - SW1 ON :(
  - SW1 OFF :(
- チャタリング防止回路は無し
  - 必要があればソフトウェア的に対処する事





拡大図



- ・ボタンスイッチ\*2
  ・黄色・白色
- スライドスイッチ\*1
- •10kΩ抵抗\*6



fritzing



fritzing

### 回路の簡単な説明

- VS-WRC10LV基板には、ボタンスイッチが1つ実装されており、IOへの割付は以下の通りです。
  - ボタンスイッチ(黒) : PIO0\_1
- ブレッドボード上には、ボタンスイッチとスライドスイッチが実装されており以下の通りの 割付です。
  - ボタンスイッチ(左・白) : PIO2\_10
  - ボタンスイッチ(右・黄) : PIO2\_11
  - スライドスイッチ1 : PIO2\_4
  - スライドスイッチ2 : PIO2\_5
  - スライドスイッチ3 : PIO2 6
  - スライドスイッチ4 : PIO2\_7
- <u>スイッチがオフ</u>の状態では、ポートはプルアップされた状態になる為、'1'が入力されます。 LPC1343
- スイッチがオンの状態では、ポートはGNDに接続されますから、'0'が入力されます。

例題:rei03 01.c

# #ifdef \_\_USE\_CMSIS #include "LPC13xx.h" #endif

```
// 過去に作成した関数などは省略
```

```
int main(void) {
    init_led();
    LPC GPI00->DIR &= ~0x0002;
```

```
while(1) {
    if((LPC_GPI00->DATA & 0x0002) == 0){
        set_led_orange(LED_ON);
    }else{
        set_led_orange(LED_OFF);
    }
}
return 0 ;
```

- スイッチ(SW1)を押下している
   間、橙LEDが点灯するプログラム
- DIRレジスタの設定
  - LPC\_GPIO0->DIR &= ~0x0002;
  - GPIO0の2bit目を入力設定に
- DATAレジスタからの読み出し
  - if((LPC\_GPI00->DATA & 0x0002) == 0)
  - GPIO0の2bit目の入力状態のみを 抽出
    - SW1が押下であるかを判断

課題

### ファイル名は「kadai03\_xx.c」とすること。

- 1. ボタンスイッチ(黒)を押下している間、緑LEDと橙LEDが点 灯するプログラムを作成せよ
- 2. ボタンスイッチ(黒)を押下している間、緑LEDと橙LEDが同時に500msec周期で点滅するプログラムを作成せよ

課題

ファイル名は「kadai03\_03.c」とし、順次機能追加して実装すること。

- 1. スライドスイッチ1~4のON/OFF状態を、随時16進数でTeraterm画面に表示する
- 2. スライドスイッチ1のON/OFFによりフルカラーLEDを制御する
  - 1. ON :赤→緑→青の順で1秒周期で順次点灯する
  - 2. OFF :青→緑→赤の逆順で1秒周期で順次点灯する
- 3. スライドスイッチ4のON/OFFにより、順次点灯の開始・停止をする電源ボタンの機能を実装する
- 4. 各スライドスイッチの入力により、順次点灯の動作が変化するプログラムを作成する

|     | SSW1           | SSW2 | SSW3             | SSW4    |
|-----|----------------|------|------------------|---------|
| ON  | 赤→緑→青の<br>順次点灯 |      | 点灯間隔を<br>500msec | 順次点灯の開始 |
| OFF | 青→緑→赤の<br>順次点灯 |      | 点灯間隔を<br>1sec    | 順次点灯の停止 |

- 5. [発展]スライドスイッチ2をONにしたら順次点灯のパターンを以下に変更
  - 1. 赤→黄→青→消灯→紫→水の順で順次点灯する
  - 2. 5.1の逆順で順次点灯する

# 事例紹介:ボタン押下後に動作したLTC

// main関数
int main(void) {
 // 初期化
 init\_uart\_io(9600);
 init\_led();
 LPC\_GPI00->DIR &= ~0x0002;

// 開始時のボタン押下待ち set\_led\_orange(LED\_ON); set\_led\_green(LED\_ON); while((LPC\_GPIOO->DATA & 0x0002) != 0);// 押下待ち while((LPC\_GPIOO->DATA & 0x0002) == 0);// 指を離す待ち set\_led\_orange(LED\_OFF); set\_led\_green(LED\_OFF);

#### // <u>printf表示</u>

printf("Name:Kumikomi Taro¥n");
printf("Age:5¥n");
printf("From:Kumamoto¥n");

return 0 ;

- ビュートビルダー2でビジュア ルプログラミングしていた際に は、以下のフローで動作をして いた
  - 1. 電源投入
  - 2. 橙・緑LED点灯
  - 3. ボタンスイッチ(黒)押下
  - 4. 橙・緑LED消灯
  - 5. 書き込んだプログラムが実行
- この動作は左のコードのように 実装できる
  - kadai02\_01.cに機能追加した事例



## 関数が増えてcファイルが長い問題

#### // 課題03 03:発展課題も実装

#ifdef USE CMSIS #include "LPC13xx.h" #endif

#include <stdlib.h> #include <stdio.h> #include <string.h> #include <cr section macros.h> #include "uart.h" #include "uart io.h"

// 定数・グローバル変数定義

// 基板上のTED制御向けの定数 #define LED OFF0// 消灯 #define LED ON1// 点灯

// ブレッドボード上のフルカラーLED制御向けの定数

#define LED BLANK0x00// 消灯 #define LED RED0x01// 赤色 #define LED GREEN0x02// 緑色 #define LED BLUE0x04// 青色 #defineLED PURPLE(LED RED | LED BLUE)// 紫色 #defineLED\_YELLOW (LED\_RED | LED\_GREEN) // 黄色 #defineLED LIGHT BLUE (LED BLUE | LED GREEN) // 水色 #defineLED WHITE (LED RED | LED BLUE | LED GREEN) // 白色

#### // プロトタイプ宣言

// LED制御関数群 void init led(void); void set led orange(unsigned char value); void set led green(unsigned char value); void init ex led full color(void); void set ex led full color(unsigned char value);

#### // Wait.関数群

extern inline void wait 1ms(void); extern inline void wait ms(unsigned short ms);

#### // スイッチ制御関数群

void init button(void); unsigned char get button black (void); void init ex slide switch(void); unsigned char get ex slide switch(void); unsigned char is ex slide switch1 on (void); unsigned char is ex slide switch2 on (void); unsigned char is ex slide switch3 on (void); unsigned char is ex slide switch4 on(void);

#### // メイン関数

int main(void) { unsigned char i = 1unsigned int msec = unsigned char power const unsigned char LED RED, LED YELLOW };// 赤・黄・青・消灯 unsigned char flash

init uart io(9600); init button(); init ex slide s

> set led orange ( while(1) {

// 課題03 0

set led green LED ON ,

printf("sw1:%02X, ", get button black()); printf("ssw:%02X¥n ", get ex slide switch()); printf(" ssw1:%x, ", is ex slide switch1 on()); printf(" ssw2:%x, ", is ex slide switch2 on()); printf(" ssw3:%x, ", is ex slide switch3 on()); printf(" ssw4:%x¥n", is ex slide switch4 on()); printf("-----¥n");

set led green (LED OFF);

### • kadai03 03 ex.cで240行ほどの コードの長さがある

- まだ、LED/printf/Switchのみで あり、今後まだまだ増える…!!
- <sup>init\_Ied();</sup> <sup>init\_ex\_led\_ful</sup> 既に可読性が落ちている

ファイル分割



- 1ファイルのコード量が増えると可読性が落ち、メンテナンスし辛い。追加で開発し辛い。といった側面が現れる
- 対応策としては、コードを複数のファイルに分割し、分散管理する手法
  - チーム開発時にも有効であり、
     1つのファイルを複数名で同時
     に編集する場面を避けられる
  - アプリ・システム全体が見通しやすくなる

## C言語におけるファイル分割



- C言語の場合、ヘッダファイル (hoge.h)とソースファイル (hoge.c)をワンセットに分割する スタイルが慣習的にある
   例:
- ヘッダファイル
  - 定数やグローバル変数、プロトタイ プ宣言を記述する
  - includeガードを記述する
- ・ソースファイル
  - 関数の実体などを記述

# C言語におけるファイル分割②



# 今回の分割方針



- ファイル分割の際の切り分け方方は、設計者や対象とするシステムによって異なる
- 今回は次のように分割する
  - pk\_ltc
    - -• ライントレースカーに関する関数や定数 などを管理
    - LEDやスイッチ、モータなど
  - ex\_dock
    - 学習&デバッグ用のドック上で構築する
       回路などの制御に関する関数や定数などを管理
    - フルカラーLEDやスライドスイッチ、温 度センサなど
  - wait
    - wait関数を管理
  - Stdcall
    - システムコールを管理



- Kadai03\_03.cをベースにファイル分割を行い、ソースコードを 整理します
  - test02\_kadai.zipをプロジェクトに追加して下さい
    - Quickstart Panel>import project(s)>Project archive(zip)より追加可能
  - 前のページの分割方針に合わせて、既に雛形が出来上がっています
  - pk\_ltc.cおよびex\_dock.cが未完成となっていますので、こちらを埋め て完成させてください
- 注意事項
  - Redlibの設定は再度行う必要があります
  - 今後、書き込むbinファイルはtest02.binという名前に変わります

## 05:ADコンバータ入門



### はじめに

- ADコンバータとは、アナログ量をデジタル量に変換する機能です。多くのマイコンに内蔵されています。
- 物理世界のほとんどの物理量はアナログ量であり、温度・湿度・光量・音量・加速度など全て連続的なアナログ値になります。
- 組込み製品ではこれらのアナログ量を扱うことが多々ありますが、そのままでは扱えない問題がある。そこでアナログ量をデジタル量に変換し、マイコンでも扱えるようにする。

## アナログ値がデジタル値に 変換されるまでの流れ



#### ・センサ

- アナログ量を電圧に変換するモノ
- 変換できるアナログ量、電圧はセンサによって様々
- 変換出力は電圧だけではなく、電流や抵抗値のことも
  - この場合、ADCとの間にI→VやR→Vの変換回路を入れる
- ADコンバータ(ADC)
  - 電圧をデジタル値に変換するモノ
  - 分解能が高ければ高い程、より細かくデジタル値で表現できる

図の事例の場合、マイコンが… ADCから0を取得したら0℃ ADCから1023を取得したら100℃ ということになる

## 分解能について

- ADコンバータの性能を指す。分解能が高い程、より細かくアナログ 値をデジタル値に変換できる。
- 通常、ビット数で表され、8bit, 10bit, 12bit, 16bit…というように表 現される。
- •このビット数は、デジタル量で表現できる大きさを表す。
  - 仮に2bitのADコンバータであれば、4段階に分けてアナログ量を表現できる
  - 温度センサであれば、「冷たい・ぬるい・暖かい・熱い」の四段階程度
- 分解能が高ければより細かい正確な測定ができるのか?
  - 高度な測定に必要な要素の1つであるが、他にも必要とされる要素がある為、 分解能1つでは高度な測定は実現できない




- ボリュームと温度センサについて説明
- 回路図



# 回路の簡単な説明

- ・ボリューム
  - ボリューム抵抗、可変抵抗などと呼ばれるツマミを回すことで抵抗値を変化させることが出来る抵抗器です
  - 総和(Ra+Rb)は常に一定で、ツマミの位置でRa とRbの値が変化します

| ツマミ        | 1-2間 | 2-3間 |
|------------|------|------|
| 反時計回りに回し切る | 1kΩ  | 0Ω   |
| 真ん中に合わせる   | 500Ω | 500Ω |
| 時計回りに回し切る  | 0Ω   | 1kΩ  |

- 総和はボリュームの型番によって変わります
   100Ω, 1kΩ, 10kΩ, 100kΩなどがある
- ボリュームはマイコンのAD5へ接続





# 回路の簡単な説明

- •温度センサ(LM61)
  - •-30~100°Cの温度測定が可能なセンサ
  - -30~100℃の温度を+300mV~1600mVの 電圧に変換
    - 0°Cの時に600mVの電圧が出力される
- ・温度センサはマイコンのAD6へ接続



 $V_{O} = (+10 \text{ mV} / ^{\circ}\text{C} \times \text{T} ^{\circ}\text{C}) + 600 \text{ mV}$ 

| Temperature (T) | Typical V <sub>O</sub> |
|-----------------|------------------------|
| +100°C          | +1600 mV               |
| +85°C           | +1450 mV               |
| +25°C           | +850 mV                |
| 0°C             | +600 mV                |
| –25°C           | +350 mV                |
| –30°C           | +300 mV                |

#### 回路の簡単な説明

- VS-WRC10LV基板には、ボタンスイッチが1つ実装されており、IOへの割付は以下の通りです。
  - ボタンスイッチ(黒) : PIO0\_1
- ブレッドボード上には、ボタンスイッチとスライドスイッチが実装されており以下の通りの 割付です。
  - ボタンスイッチ(左・白) : PIO2\_10
  - ボタンスイッチ(右・黄) : PIO2\_11
  - スライドスイッチ1 : PIO2\_4
  - スライドスイッチ2 : PIO2\_5
  - スライドスイッチ3 : PIO2 6
  - スライドスイッチ4 : PIO2\_7
- <u>スイッチがオフ</u>の状態では、ポートはプルアップされた状態になる為、'1'が入力されます。 LPC1343
- スイッチがオンの状態では、ポートはGNDに接続されますから、'0'が入力されます。

例題:rei05 01.c

#### #ifdef \_\_USE\_CMSIS #include "LPC13xx.h" #endif

#### // 省略

```
int main(void) {
    int vol, ls_l, ls_c, ls_r;
    double temp, dist;
```

```
init_syscall();
init_adc();
```

#### while(1) {

```
vol = get_volume();
ls_l = get_left_line_sensor();
ls_c = get_center_line_sensor();
ls r = get right line sensor();
```

```
temp = get_temperature();
dist = get_distance();
```

```
printf("vol:%4d, l:%4d, c:%4d, r:%4d,", vol, ls_l, ls_c, ls_r);
printf("temp:%4.21f, dist:%4.21f¥n", temp, dist);
```

wait\_ms(500);

return 0 ;

- 各種センサの値を取得し、
   シリアル通信で値を確認
   できるプログラム
  - ・ボリューム(可変抵抗)
  - ラインセンサ左
  - ラインセンサ右
  - ラインセンサ中央
  - ・ 温度センサ
  - •距離センサ

# 例題:rei05\_01.c(動作確認)

- 各種センサの値を変化させ、シリアル通信で出力されるAD値の 値が変化する事を確認しましょう
  - ラインセンサ

|        | 左 | 中央 | 右 |
|--------|---|----|---|
| 何もしない時 |   |    |   |
| 指で覆った時 |   |    |   |

• 温度センサ

| 何もしない時        |  |
|---------------|--|
| ドライヤーで5秒間あてた時 |  |

•距離センサ

| 何もしない時         |  |
|----------------|--|
| LTCの前方に手をかざした時 |  |

# ADCのレジスタ制御 (初期化)

- IOCONの設定(UM Chap7参照)
  - IO Pinの機能割当をADに変換する
  - プルアップモードを無効化
- SYSCONの設定(UM Chap3.5参照)
  - 電源管理機能よりADCへ電源供給する
  - システム及び周辺ブロックへのクロック管理機能より、ADCへクロック供給する
- AD Control Register(UM Chap20.6.1参照)
  - ADCの管理を行うレジスタの初期設定を行う

# ADCのレジスタ制御(値取得)

- AD Control Register
  - AD変換の開始・停止制御
- AD Data Register 0 ${\sim}7$ 
  - 各ADチャンネルに対して変換中の確認
  - 変換できたデジタル値の取得
  - オーバーラン発生の有無確認
- AD General Data Register
  - 最新のAD変換の結果が格納される
  - 実行中やオーバーラン発生情報なども含まれる



# センサからの温度算出方法



(…温度からの電圧算出)

- 今回使用するLM61と呼ばれるセンサは-30°C~+100°Cまでの温度範囲を検出できる温度センサIC
- 出力電圧は摂氏温度にリニアに比例(+10mV/°C)
- 内部に600mVのDCオフセットを持つ
  - つまり、出力が600mVの時に0°Cとなる
- 温度と電圧の関係は以下の算出式から求められる
  - $Vo = (+10mV/^{\circ}C * T^{\circ}C) + 600mV$
  - T°C = (Vo 600mV) / +10mV (…電圧からの温度算出)

 Temperature (T)
 Typical Vo

 +100°C
 +1600 mV

 +85°C
 +1450 mV

 +25°C
 +850 mV

 0°C
 +600 mV

 -25°C
 +350 mV

 -30°C
 +300 mV

# アナログ値がデジタル値に 変換されるまでの流れ



今回の場合、リニアリティの取れたセンサである為、温度算出は非常に容易

- 一方で、出力電圧が0.3v~1.6vと差が1.3v程度しかない.
   ADCは0~3.3vの領域を0~1023に変換する為、ADCのスペックの半分程度しか出せていない
  - ADCのスペックを最大限に出したい場合、温度センサとADC間にゲインやオフセット調整の為の回路が必要になる

課題

#### ファイル名は「kadai05\_xx.c」とすること。

1. ボリュームの値でLチカ。ボリュームで取得した値が... 500未満:フルカラーLEDを消灯 500以上:フルカラーLEDを点灯(赤)

2. ボリュームで取得した値が…
~299 : フルカラーLEDを消灯
300~599 : フルカラーLEDを点灯(青)
600~899 : フルカラーLEDを点灯(緑)
900~ : フルカラーLEDを点灯(赤)

## 06:ADCによるラインセンサ制御



#### ラインセンサの仕組み



- 赤外線LEDより赤外線光を照射
   する
- 白い床、黒いラインに光が当たって反射した光を受光素子
   (フォトダイオードorフォトトランジスタ)で受け取る
- ・受光素子は、その光量に合わせた電圧を出力する







• CN6,7,8の先にはそれぞれラ インセンサが接続されている

| ラインセンサ左  | CN | (AD | ) |
|----------|----|-----|---|
| ラインセンサ中央 | CN | (AD | ) |
| ラインセンサ右  | CN | (AD | ) |

- 回路図に起こすと左図のよう な形となる
- ・受光素子は光量が多ければ、
   多いだけ電流を流す
  - CN9には測距センサが接続さ れている

#### デジタル値と物理量

- ラインセンサやボリュームなどは正確な物理量の算出は出来ない
  - ラインセンサ:光量からライン検知
  - ボリューム :回転角度の検出
  - 変化量から何かを判断したい場合、この手のセンサは扱われる
- ・温度センサや測距センサは、デジタル値→電圧→物理量まで算 出が可能
- •2種類あるので、扱うセンサがどちらか

## 07:ADCによる測距センサ制御



### 測距センサの仕組み



[仕組みについて]

- 赤外線LEDから照射された光の反射で測量する
- ポイントは受光素子であり、反射した光の角度
   によりスポット位置が異なる
- この位置から距離を計算によって求める。という仕組み



# 距離の算出方法



\*1 : Using reflective object : White paper (reflective ratio : 90%)

- グラフから以下の事が分かる
  - 2Vの時、物体まで10cm
  - 1.3Vの時、物体まで30cm
  - 10.6Vの時、物体まで50cm
- このグラフの切片と傾きを求める y=ax+b b=2.35 2=10a+b y=-0.035x+2.35 0.6=50a+b a=-0.035 y:電圧, x:距離
- YとXを入れ替えると…
  - x=(-200y+470)/7
  - •本式により電圧から距離が算出可



課題

ファイル名は「kadai06\_xx.c」とすること。

- 1. 温度計
  - 温度センサの値が…
    - 20°C未満 20°C以上28°C未満 28°C以上35°C未満 : フルカラーLEDを点灯(緑) 28°C以上35°C未満 : フルカラーLEDを点灯(赤)
- 2. 高温アラーム機能の追加
  - 1の機能に以下を追加
    - 35℃以上40℃未満 :フルカラーLEDを点灯(赤)を500ms周期で点滅 40℃以上 :フルカラーLEDを点灯(赤)を100ms周期で点滅

#### 課題

- ファイル名は「kadai07\_xx.c」とすること。
- 1. ラインセンサの組み合わせ
  - 左・中央・右のセンサが覆われていない
  - 左のセンサが覆われた
  - 右のセンサが覆われた
  - 左・右のセンサが覆われた
  - 左・中央・右のセンサが覆われた

(点滅周期は全て200ms周期)

- 2. 測距センサの関数を完成させよう!
  - -現在はADコンバータで取得したデジタル値を得られるだけ デジタル値を距離に変換してください
- 3. [発展] 侵入検知システム
  - -20cm以内に物体を検知した時,緑・橙LEDを交互点灯するプログラムを作成

- :緑・橙LEDを消灯
- :緑LEDを点滅
- : 橙LEDを点滅
- :緑・橙LEDを点滅
- :緑・橙LEDを点灯

## 08:GPIOによるDCモータ制御



#### はじめに

- •いくつか種類のあるモータの中でも、この章ではDCモータの 制御について学びます。
- ここでは、DCモータ制御の中でも基本的な正転・逆転の方法と速度調整の方法について触れます。
  - ・電圧を印加することで回転速度を容易に制御でき、また印加する電圧の方向で回転方法の制御ができる、容易に制御が可能なモータです。
     ・一方で、DCモータ単体での位置決め制御は向きません。
- 今回使うDCモータはブラシモータとも呼ばれ、電動歯ブラシ USBファンやミニ四駆などにも利用される安価なモータです。





- DCモータを制御する典型的な Hブリッジ回路
  - 正転/逆転/停止を実現する
  - •1方向に対して回転させる/停止 させるならばFETは1個で実現
- PIO2\_0~2\_3でモータの回転 方向の制御
- PIO0\_8,PI1\_9で速度制御

Hブリッジ回路



|          | 左モータ(CN3) |        | 右モータ(CN4)     |        |        |         |
|----------|-----------|--------|---------------|--------|--------|---------|
| Port     | PIO2_0    | PIO2_1 | PIO0_8        | PIO2_2 | PIO2_3 | PIO1_9  |
| 正転       | Н         | L      | H or_Ţ_       | Н      | L      | H or_Ţ_ |
| 逆転       | L         | Н      | H or <b>_</b> | L      | Н      | H or    |
| ブレー<br>キ | Н         | Н      | Н             | Н      | Н      | Н       |
| フリー      | -         | -      | L             | -      | -      | L       |

# 例題①:コマンドでモータ制御

#### int main(void) { char cmd;

```
init_syscall();
init_motor();
init_ex_led_full_color();
```

```
set_ex_led_full_color(LED_LIGHT_BLUE);
```

```
printf("-----¥n");
printf("*** Motor Control TEST ***¥n");
printf("-----¥n");
while(1) {
    printf("input command:");
    scanf("%c", &cmd);
    printf("¥n");
```

```
switch(cmd){
  case 'a':
    rotate_motor(MOTOR_LEFT, CW);
    printf("> LEFT CW¥n");
    break;
  case 'b':
    rotate_motor(MOTOR_LEFT, CCW);
    printf("> LEFT CCW¥n");
    break;
  case 'c':
    rotate_motor(MOTOR_LEFT, BRAKE);
    printf("> LEFT BRAKE¥n");
  }
}
```

```
break;
```

```
case 'd':
           rotate motor(MOTOR LEFT, FREE);
          printf("> LEFT FREE¥n");
           break;
 case 'e':
          rotate motor (MOTOR RIGHT, CW);
          printf("> RIGHT CW¥n");
          break:
 case 'f':
           rotate motor (MOTOR RIGHT, CCW);
          printf("> RIGHT CCW¥n");
          break:
 case 'q':
           rotate motor(MOTOR RIGHT, BRAKE);
          printf("> RIGHT BRAKE¥n");
          break:
 case 'h':
          rotate motor(MOTOR RIGHT, FREE);
          printf("> RIGHT FREE¥n");
          break:
return 0 ;
```

必ずLTCを浮かして動かすこと

```
TeraTermからコマンドを
打ち込んで動作確認
```

```
コマンド一覧は以下の通り
 'a'
         :左モータ 正転
  'h'
         :左モータ 逆転
         :左モータ ブレーキ
  'C'
 •
         :左モータ フリー
  'ď'
         :右モータ 正転
  'e'
 •
  'f'
         :右モータ 逆転
         :右モータ ブレーキ
  'g'
 •
         ・右モータ フリー
  'h'
```

# 課題①:コマンドによる前進・後進ほか

- kadai08\_01.cの名前で作成
- 前進・後進・右旋回・左旋回・停止・フリーをする関数を作成

| void _f | forward(void);   |
|---------|--|
| void k  | back(void);  |
| void _t | <pre>turn_left(void);</pre>                                    |
| void _t | turn_right(void);  |
| void s  | stop(void);  |
| void _  | free(void);  |
|         | void _i<br>void _i<br>void _i<br>void _i<br>void _i<br>void _i |

•コマンドは以下の通り

| 'i'  | :前進  | 'm' | :停止  |
|------|------|-----|------|
| (;;' | :後進  | 'n' | :フリー |
| 'k'  | :左旋回 |     |      |
| " "  | :右旋回 |     |      |

## 課題②:前進&後進

- ・ここより2人1組でやること
  - ・1人しかいない席は、LTCを2台用いて行う
  - 片方の人がドックからライントレースカーを取り外す
    - 1台は机上でのデバッグ・検証用途
    - もう1台は実際に動作させてのデバッグ・検証用途
- ・ボタン押下後、1秒の前進をしたあと、1秒の後進し、最後に停止をするプログラムを作成せよ



# 速度制御とその仕組み



- PWM制御による速度制御 を実現
- モータに流す電流を細かく ON/OFFさせてTotalで流れ る電流量を調整することで 速度変化が生じる



# 例題②: GPIO+Waitによる速度制御

```
while(1) {
   mode = get ex slide switch();
   vol = get volume();
   if(vol <= 0)
   vol = 1;
   //printf("%x, %d¥n", mode, vol);
    switch(mode)
    case 0x0f:
        stop();
       break;
    case 0x0e:
        forward();
       break;
    case 0x0c:
        forward();
        wait ms(10);
        stop();
        wait ms(10 * vol / 1000);
       break;
```

- ドック上に置いた実機で動作確認してください
- 機能は以下の通りです
  - SSW全てOFF
    - 停止状態
  - SSW1:ON
    - 通常の前進
  - SSW1+SSW2:ON
    - ボリュームで速度調整
- プログラム書き込み後、速度調整機能が確認出来たらオシロスコープで、モータの波形を確認してください
  - ボリュームの捻り具合でデューティ比が変わる事を確認





#### rei08\_02.cの一部処理をまとめ、下記の関数を作成ください Whileループに組み込めば、周期20msec、デューティ比50%でモー タ制御を行う関数となる

```
void _forward_20ms(void);
void _back_20ms(void);
void _turn_left_20ms(void);
void _turn_right_20ms(void);
```

### 課題④:LEDの輝度調整

- 1. rei08\_02.cを改変し、フルカラーLEDの赤色の輝度調整がで きるように修正してみて下さい
  - LEDはモータに比べて即応性が高い為、waitの時間調整が必要な可能 性があります
- 2. フルカラーLEDの緑色を点灯させた状態で、1を確認してみ て下さい
- 3. その他の色についても試してみて下さい

# 09:C言語によるライントーレス制御



# ライントレース制御①

- ラインセンサを用いて線の上を走る
- 今回は中央のラインセンサは使わない
- ライントレースの基本アルゴリズム
  - 左のラインセンサがラインに反応した場合
    - 舵を右に切りながら走行する
  - 右のラインセンサがラインに反応した場合
     舵を左に切りながら走行する
  - どちらのセンサを反応しない場合
    - 直進する
  - 以上をひたすら繰り返す


# ライントレース制御2)

- ライン(白・黒)を識別しよう
- ラインが識別出来たら…
  - ラインセンサの値に合わせてLTCの動作(振舞い)を定めよう!
- Let's Programing!!
  - ファイル名:kadai09\_01.c

| 場所      | ラインセンサ(左) | ラインセンサ(右) |
|---------|-----------|-----------|
| ライン上(黒) |           |           |
| それ以外(白) |           |           |

| ラインセンサ(左) | ラインセンサ(右) | LTCの動作 |
|-----------|-----------|--------|
| 白         | 白         |        |
| 黒         | 白         |        |
| 白         | 黒         |        |



### ヒント:ドック無しでシリアル通信



CN5からもシリアル通信に必要なTxD, RxDピンは取得可能です PK-LTC用の拡張ブレッドボードと併用する事でドックと接続していなくても シリアル通信が実現可能です fritzing

# 課題①:センサ1つでライントレース

- ファイル名: kadai09\_02.c
- ラインセンサ(中)だけを用いてライン
  トレースするプログラムを実現しよう。

| 場所      | ラインセンサ(中) |  |
|---------|-----------|--|
| ライン上(黒) |           |  |
| それ以外(白) |           |  |

| ラインセンサ(中) | LTCの動作 |  |
|-----------|--------|--|
| 山         |        |  |
| 黒         |        |  |



# 課題②:灰色を検知したら橙LED点灯

• ラインセンサ(中)で灰色のラインを検出したら、橙LEDを点灯させるプログラムを作成してください。ファイル名:kadai09\_03.c

| 場所            | ラインセンサ(左)     | ラインセンサ(中      | ) ラインセンサ(右) |
|---------------|---------------|---------------|-------------|
| ライン上(黒)       |               |               |             |
| ライン上(灰)       |               |               |             |
| それ以外(白)       |               |               |             |
| ラインセンサ<br>(左) | ラインセンサ<br>(中) | ラインセンサ<br>(右) | LTCの動作      |
| 白             | _             | 白             |             |
| 黒             | _             | 白             |             |
| 白             | _             | 黒             |             |
| _             | 白             | _             |             |
| _             | 灰             | _             |             |
| _             |               | _             |             |



#### 新たな課題の発見

- •ファイル名:kadai09\_04.c
- •作成したライントレースのプログラムに、200ms周期で緑LED を点滅させるコードを追加してみて下さい。

挙動はどのように変わるでしょうか?









拡張コネクタのピン配置図

CN5 CN13 CT32B1\_MAT2 CT32B0\_MAT1 CT32B0\_MAT1 CT16B0\_MAT2 CT16B1\_MAT1 CT32B0\_CAP0 PI00\_4 AD4 TxD AD6 TXD PI01\_10 PI00\_10 PI01\_3 PI02\_11 PI01\_7 PI01\_7 PI02\_5 PI02\_7 ഹ +3.3V +3.3V +5V PI01\_ RES RES +5ν SCL 13 25 23 21 19 17 15 11 9 5 3 9 7 5 3 1 1 22 10 26 24 20 18 16 14 12 10 8 6 4 2 8 6 2 4 PI02\_10 PI01\_6 P100\_9 PI01\_6 PI02\_6 P100\_2 VBATT PI02\_4 VBATT 4 GND GND SDA PI01 PIO0\_5 RxD AD5 RxD CT32B0\_MAT0 CT32B0\_MAT0 CT16B0\_MAT1 CT32B1\_MAT3 CT16B0\_CAP0 9,10pinはCN5の1,2pinと被ってる 5,6pinは3.3KΩでプルアップ済み \* \* 3,7,8pinはCN12の2,3,4pinと被っている 1,2pinはCN5の9,10pinと被ってる \* \*

P103\_3 PI03\_1 +3.3V +5ν 9 5 3 7 1 10 2 8 6 4 VBATT PI03\_0 2 GND PIO3\_

CN14

\* 3pinは33kΩでプルアップ済み

CN5,13,14への再接続時のチートシート



### 関連リンクほか

- ビュートローバーARM(ヴイストン社)
  - <u>https://www.vstone.co.jp/products/beauto\_rover/index.html</u>
  - <u>本テキストに関する問い合わせについて、ヴイストン社は受け付けておりま</u> <u>せんのでご注意ください。</u>
- LPCXPresso(NXP社)
  - <u>http://www.nxp-lpc.com/lpc\_boards/lpcxpresso/</u>
- ARMマイコンによる組込みプログラミング入門(オーム社)
  - <u>http://shop.ohmsha.co.jp/shop/shopdetail.html?brandcode=0000000146</u> <u>1&search=ARM%A5%DE%A5%A4%A5%B3%A5%F3</u>
- Special Thanks
  - SSEST5(Summer School on Embedded System Technologies)