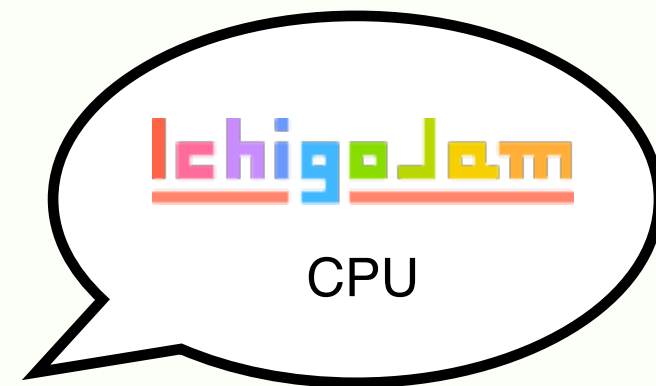
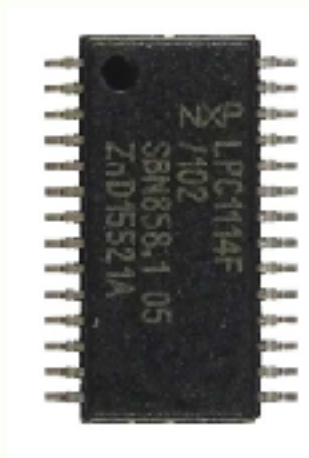


IchigoJamは、本当に 1秒に5000万回計算できるのか!?

～IchigoJam マシン語入門～



jig.jp 会長 / IchigoJam 開発者
@taisukef <https://fukuno.jig.jp/>

福野泰介





IchigoJam

CPU

1 秒に5000万回！

本当に？

は か る う !



じかんのはかりかた

?TICK()

?TICK()

CLT

?TICK()

CLTでリセット

足し算1000回の速さ

NEW

10 N=0:CLT

20 N=N+1:IF N<1000 CONT

30 ?TICK()/6

RUN

何秒？

18 = 1.8秒

おや？おそくない？



足し算1000回の速さ

NEW

5 VIDEO0

10 N=0:CLT

20 N=N+1:IF N<1000 CONT

30 ?TICK()/6

40 VIEDO1

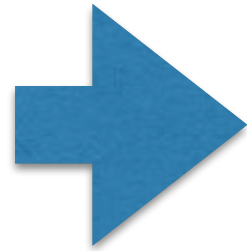
表示を消して
本気出す

1秒1000回くらい
なぜ"おそい"？

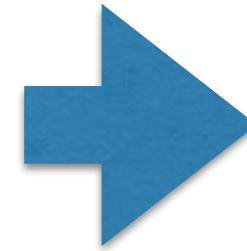


インタプリタ (ほんやくしゃ)

LED1



```
while (program) {  
  cmd = ...;  
  param = ...;  
  if (cmd == CMD_LED) {  
    if (param) {  
      GPIO_LED |= LED;  
    } else {  
      GPIO_LED &= ~LED;  
    }  
  } else if (cmd == CMD_WAIT) {  
    ...  
  } ...  
  ...  
}
```



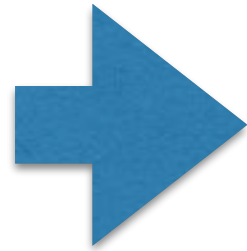
BASIC
人にやさしい

BASICのコマンドを
マシン語に都度翻訳

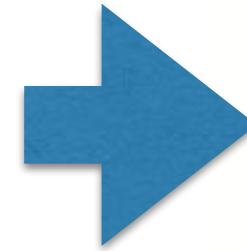
マシン語
わかる

コンパイラ (ほんやくき)

led(1)



```
`00100 011 01010000  
`00000 01000 011 011  
`00110 011 00000001  
`00000 10000 011 011  
`00110 011 00111100  
`01100 00000 011 000  
`0100011101110000
```



C 言語

人にやさしい

事前にマシン語に
翻訳してしまう

マシン語


わかる

マシン語ではなそう



LPC1114 のとりせつ

参考資料



UM10398


LPC111x/LPC11C1x ユーザーマニュアル

Rev. 00.15 — 2010 / 9 / 28

ユーザーマニュアル

Document information	
Info	Content
Keywords	ARM Cortex-M0, LPC1111, LPC1112, LPC1113, LPC1114, LPC11C12, LPC11C14
Abstract	LPC111x/LPC11C1x User manual

本マニュアルは、参考資料です。英文マニュアルは頻繁に更新されます。
最新情報は英文マニュアルをご参照ください。



参考資料

UM10398

第 1 章：LPC111x/LPC11C1x の概要

Rev. 00.15 — 2010 / 9 / 28

ユーザーマニュアル

1.1 はじめに

LPC111x/LPC11C1x は ARM Cortex-M0 ベースの高性能 32 ビット MCU ファミリーであり、8/16 ビットのマイクロコントローラアプリケーション用に設計され、高性能、低消費電力、簡便な命令セットとメモリアドレッシングを提供するとともに、既存の 8/16 ビットノートデバイスに比べてコードサイズが小さいことが特長です。

LPC111x/LPC11C1x は、最大 50 MHz の CPU 周波数で動作します。

LPC111x/LPC11C1x のペリフェラルコンプリメントには、最大 32 kB のフラッシュメモリ、最大 8 kB のデータメモリ、1 個の C_Can コントローラ (LPC11C12/14)、1 個の Fast モードプラス I/O バスインターフェース、1 個の RS-485/EIA-485 UART、最大 2 個の SSP 機能付き SPI インターフェース、4 個の多目的タイマ、1 個の 10 ビット ADC、最大 40 個の汎用 I/O ピンがあります。

LPC11C12/14 には、オンチップ C_Can ドライバと、C_Can を介したフラッシュ・インシステムプログラミング (ISP) ツールが内蔵されています。

1.2 特長

- システム：
 - ARM Cortex-M0 プロセッサ、最大周波数 50 MHz で動作。
 - ARM Cortex-M0 内蔵ネストベクタ割り込みコントローラ (NVIC)。
 - シリアルワイヤデバッグ (SWD)。
 - システム tick タイマ。
- メモリ：
 - 32 kB (LPC1114/LPC11C14)、24 kB (LPC1113)、16 kB (LPC1112/LPC11C12)、または 8 kB (LPC1111) のオンチップ・フラッシュプログラミングメモリ。
 - 8 kB、4 kB、または 2 kB の SRAM。
 - オンチップのブートローダソフトウェアによるインシステムプログラミング (ISP) とインアプリケーションプログラミング (IAP)。
- デジタルペリフェラル：
 - 安定可能なプルアップ/プルダウン抵抗の付いた、最大 40 個の汎用 I/O (GPIO) ピン。
 - GPIO ピンは、エッジセンシティブおよびレベルセンシティブな割り込みソースとして使用可能。
 - 1 個のピンに高電流出力ドライバ (20 mA)。
 - Fast モードプラスの I/O バスピン 2 個に高電流シンクドライバ (20 mA)。
 - 4 個の多目的タイマ/カウンタと、合計 4 個のキャプチャ入力および 13 個のマッピング出力。
 - プログラマブルなウォッチドッグタイマ (WDT)。

UM10398
ユーザーマニュアル

Rev. 00.15 — 2010 / 9 / 28

62979 Rev. 00.15 npxen-um10398 3

http://www.nxp-lpc.com/images/LPC111x_UM_Rev.00.15_Japanese.pdf

CPU: Arm Cortex-M0



Arm Cortex-M0 のとりせつ

ARM アーキテクチャ リファレンスマニュアル

ARM®

Copyright © 1996-1998, 2000, 2004, 2005 ARM Limited. All rights reserved.
ARM DDI 0100HJ-00

ARM アーキテクチャリファレンスマニュアル

Copyright © 1996-1998, 2000, 2004, 2005 ARM Limited. All rights reserved.

リリース情報

このドキュメントには、以下の変更が加えられています。

改訂履歴		
日付	変更箇所	変更内容
1996 年 2 月	A	初版
1997 年 7 月	B	更新と索引の追加
1998 年 4 月	C	更新
2000 年 2 月	D	ARM アーキテクチャ v5 に対応した更新
2000 年 6 月	E	ARM アーキテクチャ v5TE に対応した更新とパート B の修正
2004 年 7 月	F	ARM アーキテクチャ v6 に対応した更新 (非公開)
2004 年 12 月	G	誤記の修正
2005 年 3 月	H	誤記の修正

著作権表記

ARM, ARM Powered ロゴ, Thumb, StrongARM は ARM 社の登録商標です。

ARM ロゴ, AMBA, Angel, ARMulator, EmbeddedICE, ModelGen, Multi-ICE, PrimeCell, ARM7TDMI, ARM7TDMI-S, ARM9TDMI, ARM9E-S, ETM7, ETM9, TDMI, STRONG は ARM 社の商標です。

このドキュメントに表記されている他の製品やサービスは、対応する所有者の商標の場合があります。

このドキュメントに説明されている製品は、継続的に開発と改良が行われています。このドキュメントにある製品とその使用方法に関する記載事項について、ARM は保証しません。

1. 下記の条件に従い、ARM はこの ARM アーキテクチャリファレンスマニュアルを次の目的に利用する永続的、非排他的、移転不可、無料、国際的なライセンスを許可します。使用目的は、(1) ARM からのライセンスにより配布されるマイクロプロセッサコアで実行することを目的としたソフトウェアアプリケーションとオペレーティングシステム(2) ARM からのライセンスにより配布されるマイクロプロセッサコアで実行することを目的としたソフトウェアプログラムの開発用に設計されたツール(3) ARM からのライセンスにより製造されるマイクロプロセッサコアを搭載する集積回路のいずれかの開発に限られます。

2. 条項 1 で明示的に与えられているものを除き、ARM アーキテクチャリファレンスマニュアル、またはそれに含まれるいかなる知的著作物についても、いかなる権利、資格、利益も与えるものではありません。条項 1 に示されている許諾は、いかなる場合でも明示的、暗黙的、禁反言、その他の形で ARM アーキテクチャリファレンスマニュアル以外のいかなる ARM テクノロジーに関するライセンスも与えるものではありません。条項 1 で与えられるライセンスには、ARM パテントを使用する、または使用に含める権利は明示的に除外されます。条項 1 の条件では、次に示す権利は与えられません。(1) ARM アーキテクチャリファレンスマニュアルを、この ARM アーキテクチャリファレンスマニュアルで説明されている命令、プログラマモデル、

<http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.subset.architecture.reference/index.html>

まとめ

Cortex-M0 Armマシン語表 (asm15、抜粋)

代入	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	cycles
Rd = u8	0	0	1	0	0	Rd			u8								1
Rd = Rm	0	1	0	0	0	1	1	0	Rd3	Rm			Rd2-0				1,3

※Rd3とRd2-0の4bitでRdを指定する、RdがPCの時3cycles

演算	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	cycles
Rd += u8	0	0	1	1	0	Rd			u8								1
Rd -= u8	0	0	1	1	1	Rd			u8								1
Rd = PC + u8	1	0	1	0	0	Rd			u8								1
Rd += Rm	0	1	0	0	0	1	0	0	Rd3	Rm			Rd2-0				1,3
Rd = Rn + u3	0	0	0	1	1	1	0	u3			Rn			Rd			1
Rd = Rn - u3	0	0	0	1	1	1	1	u3			Rn			Rd			1
Rd = Rn + Rm	0	0	0	1	1	0	0	Rm			Rn			Rd			1
Rd = Rn - Rm	0	0	0	1	1	0	1	Rm			Rn			Rd			1

Rd	32bit レジスタ x 16 (CPU内にあるメモリ)
R0~R7	汎用的に使えるレジスタ (R4~R7は元に戻す)
R0	パラメータの受け渡しに使う
R8~R12	使えるコマンドが限られるレジスタ
R13~R15	使う用途が特殊なレジスタ

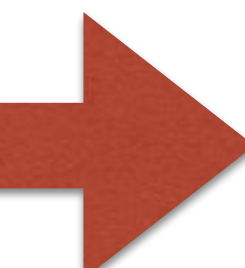
<https://ichigojam.github.io/asm15/armasm.html>

足し算させよう

Cortex-M0 Armマシン語表 (asm15、抜粋)

代入	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	cycles
Rd = u8	0	0	1	0	0	Rd			u8								1
Rd = Rm	0	1	0	0	0	1	1	0	Rd3	Rm			Rd2-0				1,3

※Rd3とRd2-0の4bitでRdを指定する、RdがPCの時3cycles



演算	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	cycles
Rd += u8	0	0	1	1	0	Rd			u8								1
Rd -= u8	0	0	1	1	1	Rd			u8								1
Rd = PC + u8	1	0	1	0	0	Rd			u8								1
Rd += Rm	0	1	0	0	0	1	0	0	Rd3	Rm			Rd2-0				1,3
Rd = Rn + u3	0	0	0	1	1	1	0	u3			Rn			Rd			1
Rd = Rn - u3	0	0	0	1	1	1	1	u3			Rn			Rd			1
Rd = Rn + Rm	0	0	0	1	1	0	0	Rm			Rn			Rd			1
Rd = Rn - Rm	0	0	0	1	1	0	1	Rm			Rn			Rd			1

足し算する時は「Rd += u8」

(レジスタd に、符号なし8bitの数を足し込む) を使う

例えばレジスタ0(R0) に 1 足す時は

、00110 000 000000001 (2進法) となる

RETURN

GOTO n11	1	1	1	0	0	n11										3	
GOTO Rm	0	1	0	0	0	1	1	1	0	Rm			0	0	0	3	
GOSUB Rm	0	1	0	0	0	1	1	1	1	Rm			0	0	0	3	
GOSUB n22	1	1	1	1	0	n22(21-11)										1	
-	1	1	1	1	1	n22(10-0)										3	
RET (= #4770)	0	1	0	0	0	1	1	1	0	1	1	1	0	0	0	0	3

呼び出しから返ってくる RETURN コマンドは
`0100 0111 0111 0000 (2進法) となる

#4770 (16進法) とも書いても
18288 (10進法) とも書いてもOK

10進法、2進法、16進法

	10進法へ	10進法から
2進法	?`111 7	?BIN\$(7) 111
16進法	?#F 15	?HEX\$(15) F

BASIC から マシン語 よび だし USR

[0]=`00110 000 000000001

[1]=`0100 0111 0111 0000

?USR(#800,0)

1

?USR(#800,255)

256

マシン語で
足し算できた！

足し算1000回の速さ

R1=0

R1+=1

R1-R0

IF !0 GOTO -2

RET



R1-R0が
0じゃない時
2つ前へ

*USRの二番目のパラメーターは、R0 になる

足し算1000回の速さ

Cortex-M0 Armマシン語表 (asm15、抜粋)

代入	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	cycles
Rd = u8	0	0	1	0	0	Rd			u8								1
Rd = Rm	0	1	0	0	0	1	1	0	Rd3	Rm			Rd2-0				1,3

*Rd3とRd2-0の4bitでRdを指定する、RdがPCの時3cycles

演算	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	cycles
Rd += u8	0	0	1	1	0	Rd			u8								1
Rd -= u8	0	0	1	1	1	Rd			u8								1
Rd = PC + u8	1	0	1	0	0	Rd			u8								1
Rd += Rm	0	1	0	0	0	1	0	0	Rd3	Rm			Rd2-0				1,3
Rd = Rn + u3	0	0	0	1	1	1	0	u3			Rn			Rd			1
Rd = Rn - u3	0	0	0	1	1	1	1	u3			Rn			Rd			1
Rd = Rn + Rm	0	0	0	1	1	0	0	Rm			Rn			Rd			1
Rd = Rn - Rm	0	0	0	1	1	0	1	Rm			Rn			Rd			1

[0]=`00100001 00000000 R1=0

[1]=`00110001 00000001 R1+=1

[2]=`01000010 10000001 R1-R0

[3]=`11010001 11111100 IF!0GOTO-2

[4]=`01000111 01110000 RET

*GOTOで使うパラメーターは2を引いて、2の補数表現 <https://fukuno.jig.jp/1188>

足し算1000回の速さ

NEW

10 POKE#800,0,33,1,49,129,66,252,
209,112,71

20 CLT:?USR(#800,1000):?TICK()

SAVE2

RUN

足し算1万回の速さ

LIST

10 POKE#800,0,33,1,49,129,66,252,
209,112,71

20 CLT:?USR(#800,10000):?TICK()

RUN

足し算100万回の速さ

LIST

10 POKE#800,100,34,80,67,0,33,1,

49,129,66,252,209,112,71

20 CLT:?USR(#800,10000):?TICK()

SAVE2

RUN

*追加 (パラメータを100倍する)

R2=100 `00100 010 1100100

R0*=R2 `0100001101 010 000

計算してみよう



足し算100万回の速さ

13/60秒 やく0.2秒で"100万回

1秒で"500万回？

まだ 10倍おそい・・・

足し算100万回の速さ

LIST

5 VIDEO0

10 POKE#800,100,34,80,67,0,33,1,
49,129,66,252,209,112,71

20 CLT:?USR(#800,10000):?TICK()

30 VIEDO1

RUN

表示を消して
本気出す

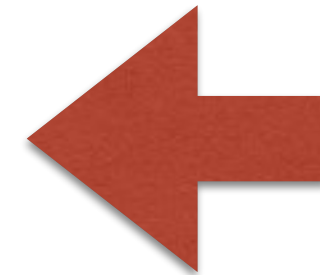
1秒1000万回！？

Cortex-M0 Armマシン語表 (asm15、抜粋)

代入	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	cycles
Rd = u8	0	0	1	0	0	Rd			u8								1
Rd = Rm	0	1	0	0	0	1	1	0	Rd3	Rm			Rd2-0				1,3

※Rd3とRd2-0の4bitでRdを指定する、RdがPCの時3cycles

演算	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	cycles
Rd += u8	0	0	1	1	0	Rd			u8								1
Rd -= u8	0	0	1	1	1	Rd			u8								1
Rd = PC + u8	1	0	1	0	0	Rd			u8								1
Rd = Rm	0	1	0	0	0	1	1	0	Rd3	Rm			Rd2-0				1,3



何サイクルで
処理するか？

R1=0

R1+=1

R1-R0

IF !0 GOTO -2

RET



← 1 cycle

← 1 cycle

← 3 cycle

=合計 5サイクル

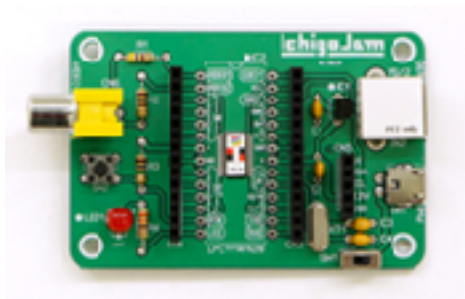
計算は1秒5000万回ペースでやっていた！



IchigoJam

CPU

1 秒に**5000万回**！



(C)IchigoJam

IchigoJam

5000万回

IchigoJam
何台分？→

1500円



(C)Apple

iPhone 11
GPU

1兆回

2万台分

8万円



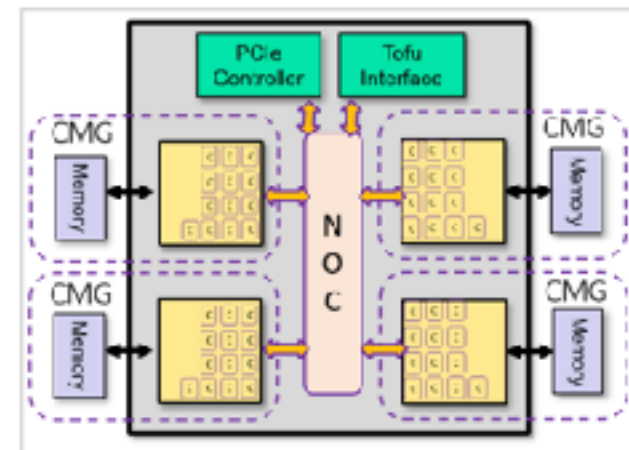
(C)TSUKUMO

パソコン
GPU

10兆回

20万台分

10万円



(C)RIKEN

スパコン富岳

100京回

200億台分

1100億円

コンピューターを
つかいこなそう



A background image showing a group of children in a classroom setting, some sitting at desks and others standing, all appearing to be engaged in an activity. The image is slightly blurred and has a dark overlay.

Hanaわらび

～すべての世代に可能性を～

Hanaわらびへどうぞ！

<https://hanawarabikoza.page/>

Hana道場（鯖江）、寺子屋Hana（会津）もあるよ