

オフラインコンパイラと CMSIS-DAPデバッグ環境の紹介

アーム株式会社

ARMサービス部門 アプリケーションエンジニア

渡會 豊政



自己紹介

渡會豊政 （わたらい とよまさ）

ソフトウェア開発ツール（主にコンパイラ）の技術サポート
LPC1114 mbedのポーティングとかやりました



Twitter : @toyowata

アーム株式会社の公式ツイッター @arm_link もよろしく申し上げます

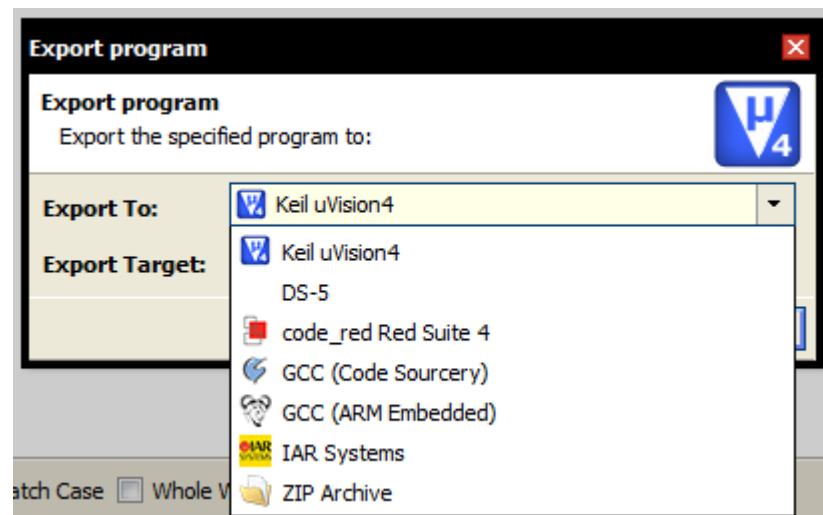


Agenda

- オフラインコンパイラについて
- KEIL™ MDK-ARM
- CMSIS-DAPを使ったデバッグ環境
- デモ

オフラインコンパイラって？

- mbedのプロジェクトファイルをエクスポートして、オフライン（ローカル）環境でビルドする
- 様々な利点
 - ターゲットのソースコードレベルデバッグ
 - 最新バージョンの最適化コンパイラの使用
- mbedでサポートされているオフライン環境
 - **Keil uVision4**
 - DS-5
 - Code_red Red Suite 4
 - GCC (Code Sourcery)
 - GCC (ARM Embedded)
 - IAR Systems
 - **ZIP Archive**



Keil™ MDK-ARMについて

- Keil と書いて「かいる」と読む
 - ドイツの開発ツールベンダー（2005年にARMによって買収）
 - 8051, C166等の8/16-bit MCU用のツールも提供中
- MDK-ARMは、IDE、コンパイラ、RTOS、ミドルウェア含めた統合開発ツールの製品名
- uVision®4は、IDEの名称
- C/C++コンパイラは、ARM純正コンパイラ (RealView®)
- ホストOSは、Windows XP SP2, Vista, 7 (32/64)

MDK-ARMエディションの違い

	MDK-Pro	MDK-Standard	MDK-Cortex-M	MDK-Lite
μVision				
IDE(統合開発環境)	✓	✓	✓	✓
デバッガ	✓	✓	✓	32KB
シミュレータ	✓	✓	✓	32KB
ARM コンパイラ				
C/C++ コンパイラツール	✓	✓	✓	32KB
RTOS, ミドルウェア				
RTX RTOS (ソースコード)	✓	✓	✓	✓
TCP Networking Suite	✓			
Flash File system	✓			
USB Driver	✓			
CAN (Source Code)	✓			
ライセンス				
購入の必要性	✓	✓	✓	無料

※ Cortex-M エディションは、Cortex-R4, ARM7, ARM9, SecureCore には未対応

ARM Compiler 5 の新機能

- Thumb-2インラインアセンブラ対応
 - C/C++ソースから、アセンブラコードの記述が可能
- 新しいコンパイラフロントエンド
- Cortex-M0+対応
 - オンラインコンパイラは、ARMv6-Mアーキテクチャ対応
- コード最適化の向上 (`--loop_optimization_level=n`)
 - インライン展開
 - ループ展開
 - <http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.dui0491i/BABDECFA.html>

ベンチマーク（参考値） など...

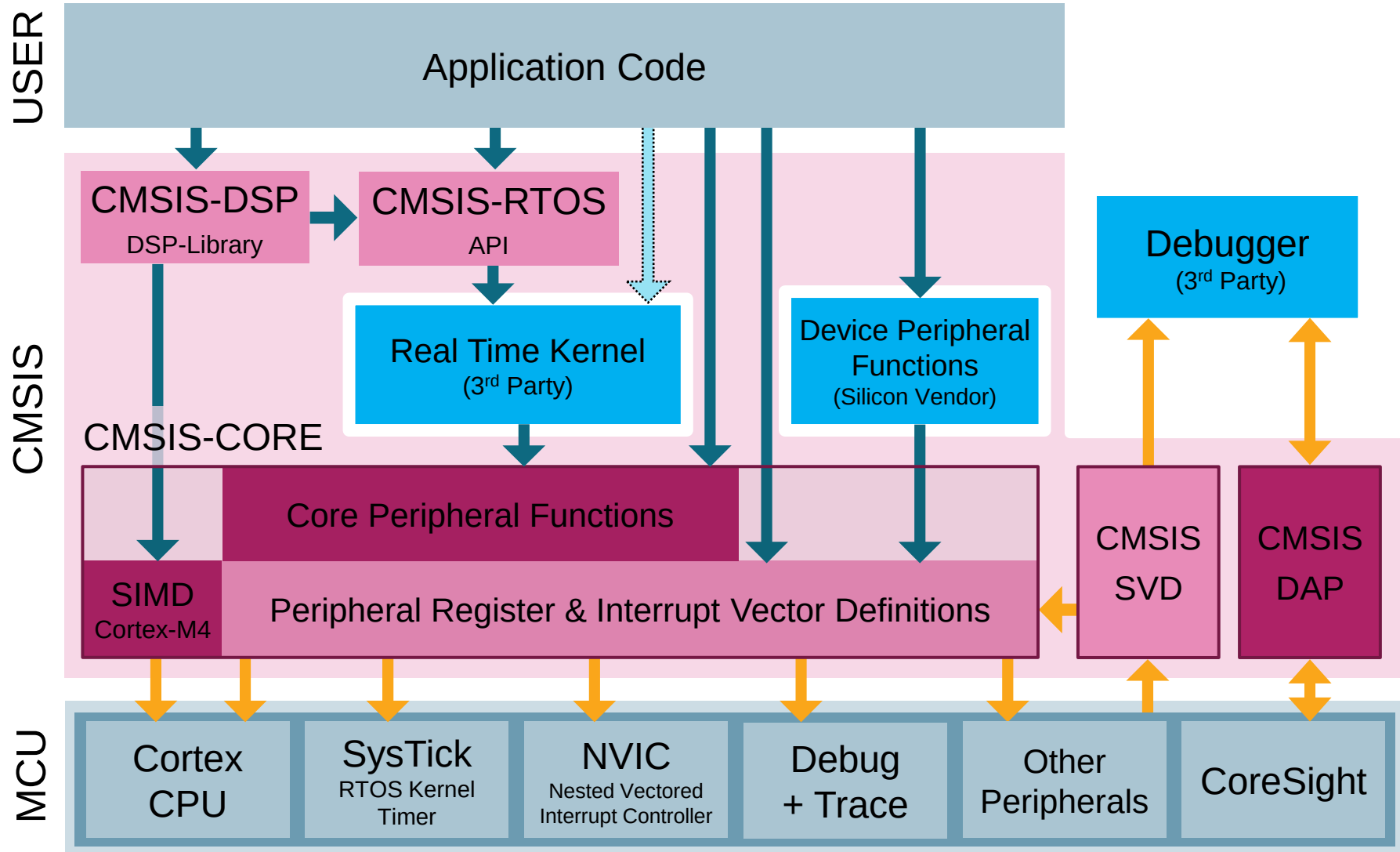
- mbed Clock Control / Benchmarks
 - <http://mbed.org/users/no2chem/notebook/mbed-clock-control--benchmarks/>
 - CoreMark[®]のmbedへの移植版

	最適化用の コンパイルオプション	CoreMark @ 96MHz	Code size (bytes)
オンライン版 armcc v4.1	-O2 -Otime	155.86	31,428
オフライン版 armcc 5.03	-O2 -Otime	161.70	31,384
オフライン版 armcc 5.03	-O3 -Otime --loop_optimization_level=2	232.88	34,272

CMSISって？

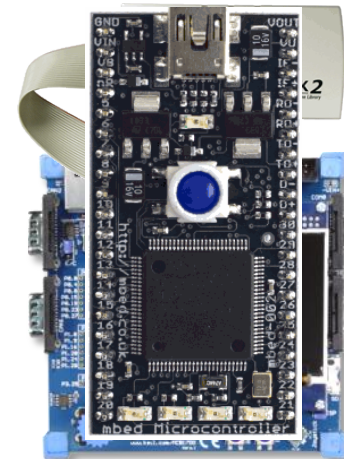
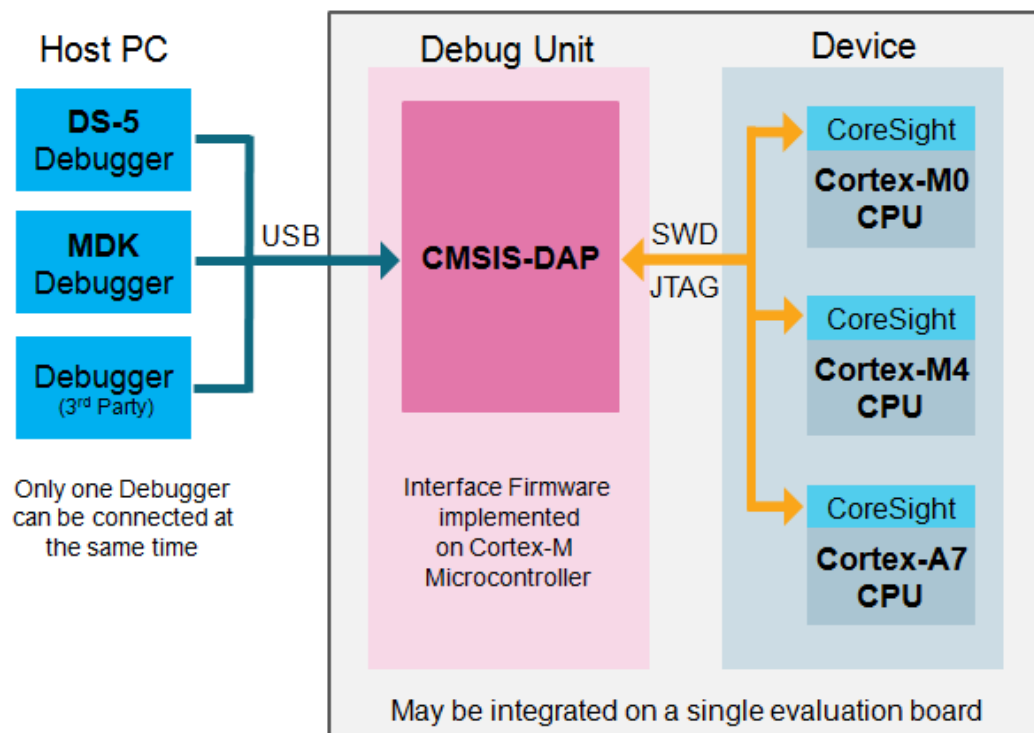
- Cortex Microcontroller Software Interface Standard
- 「シーえむしす」または、「シーえむえすあいえす」と読んで下さい ☺
- Cortex-Mプロセッサ シリーズ向けのベンダに依存しないハードウェア抽象化レイヤ
- CMSIS Coreの例
 - 特殊レジスタアクセス関数
 - `__get_xpsr()`, `__set_PRIMASK()`, `__set_MSP()` 等
 - NVICアクセス関数
 - `NVIC_EnableIRQ()`, `NVIC_GetPriority()` 等
 - SysTickアクセス関数
 - デバッグ関数
 - `ITM_SendChar()`関数を使った `printf` ライクなデバッグ出力

CMSIS v3 最新バージョン



CMSIS-DAPって？

- Firmware for the CoreSight Debug Access Port
- USB HID接続を通じて、CoreSight-DAPへのアクセスを標準化
- 異なるベンダーツールを使用したデバッグを提供
 - 例：Keil uVision4とNXP LPC-Link2の接続

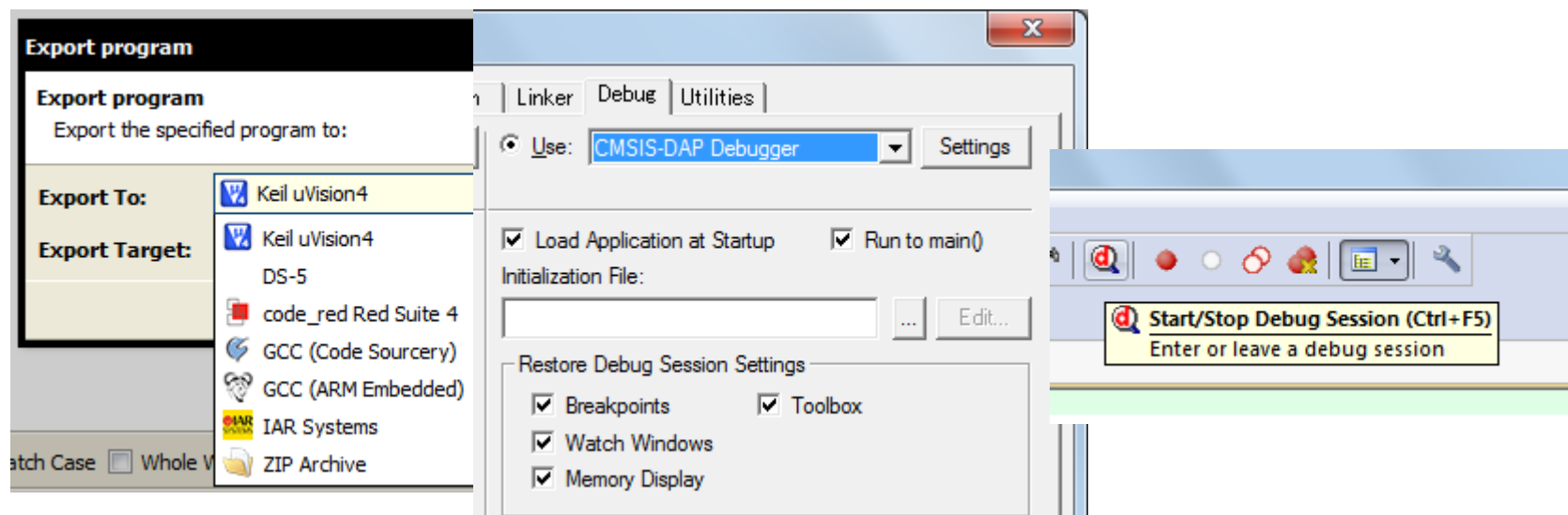


mbedをデバッグする（準備）

- ファームウェアを最新版にアップデートする
 - <http://mbed.org/handbook/CMSIS-DAP-MDK>
 - <http://mbed.org/handbook/Firmware>
- Keil™ MDK-ARM無償評価版をインストールする
 - <https://www.keil.com/arm/demo/eval/arm.htm>
- シリアルドライバのインストール（必須！）
 - http://mbed.org/media/downloads/drivers/mbedWinSerial_16466.exe

mbedをデバッグする

- オンラインコンパイラで作成したプロジェクトをエクスポートする
 - Keil uVision4を選択
- zipファイルを展開し、<project>.uvprojをダブルクリック
- Project -> Options... -> Debug
 - CMSIS-Debuggerを選択
- Debug -> Start/Stop Debug Session



mbedで使用可能なデバッグ機能

- C/C++ソースレベルデバッグ、逆アセンブラ
 - 実行・停止、シングルステップ
- フラッシュメモリへの書き込み（ROM上のコードをデバッグ）
 - USBストレージにはダウンロードしない（する事も可能）
- ハードウェアブレークポイント（4または8箇所）
 - Flashのコード領域は、ハードウェアブレークポイントを使用
- ウォッチポイント（2または4カ所）
 - Read/Writeデータアクセスブレーク
- CPUレジスタ、ペリフェラルレジスタへのアクセス

デバッガの便利な機能を使う

- パフォーマンスアナライザとコードカバレッジ
- MTBでインストラクショントレース
- ロジックアナライザ機能でバーチャルLチカ

デバッガの便利な機能を使う

- パフォーマンスアナライザ
 - ボトルネックの検出
- コードカバレッジ
 - デッドコードの検出

Performance Analyzer

Reset Show: Modules

Module/Function	Calls	Time(Sec)
[-] ClockControl		43.005 ms
[-] CoreMark/core_list...		16.414 ms
core_list_init	1	16.771 us
core_bench_list	19	13.246 ms
core_list_merg...	29	1.829 ms
cmp_idx	1974	472.938 us
cmp_complex	1009	262.615 us
calc_func	2017	586.115 us
[+] CoreMark/core_st...		11.115 ms
[-] CoreMark/core_ma...		10.689 ms
core_init_matrix	1	17.229 us
core_bench_ma...	37	11.083 us
matrix_test	37	10.661 ms
[+] CoreMark/core_uti...		4.780 ms
[+] CoreMark/core_ma...		5.469 us
[+] ClockControl/Cloc...		0.490 us
[-] CoreMark/core_po...		0.219 us
portable_fini	0	0us
portable_init	1	0.063 us
time_in_secs	0	0us
get_time	0	0us
stop_time	0	0us

Disassembly Performance Analyzer

Code Coverage

Update Clear Module: <All Modules>

Modules/Functions	Execution percentage
cmp_complex	100% of 11 instructions
calc_func	100% of 54 instructions
[-] CoreMark/core_state.c	
core_init_state	88% of 88 instructions, 14 condjump(s) not fully executed
core_bench_state	100% of 96 instructions, 8 condjump(s) not fully executed
core_state_transition	97% of 194 instructions, 24 condjump(s) not fully executed
[-] CoreMark/core_util.c	
check_data_types	0% of 2 instructions
crcu32	100% of 9 instructions
crc16	100% of 26 instructions
crcu16	100% of 26 instructions
get_seed_32	88% of 17 instructions, 1 condjump(s) not fully executed

Disassembly Code Coverage

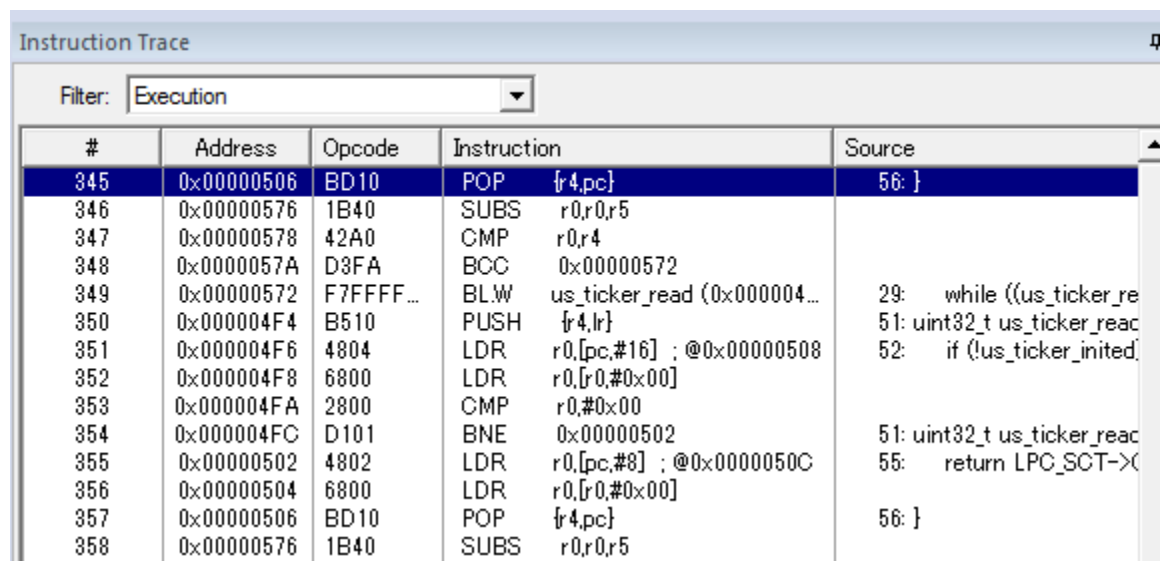
core_state.c core_util.c main.cpp core_list_join.c FileHandle.h

```
142 ee_u16 crcu8(ee_u8 data, ee_u16 crc)
143 {
144     ee_u8 i=0,x16=0,carry=0;
145
146     for (i = 0; i < 8; i++)
147     {
148         x16 = (ee_u8)((data & 1) ^ ((ee_u8)crc & 1));
149         data >>= 1;
150
151         if (x16 == 1)
152         {
153             crc ^= 0x4002;
154             carry = 1;
155         }
156     }
157 }
```

0% 0%

デバッガの便利な機能を使う

- MTB(Micro Trace Buffer)でインストラクショントレースを取得する (LPC800-Miniを使う場合)
 - C:¥Keil¥ARM¥Boards¥Embedded Artists¥LPCXpresso LPC800¥Blinky_MTB¥Dbg_MTB.ini をローカルプロジェクトにコピー
 - [Debug] – [Initialization file:] に追加
 - Buffer Size : 256B
 - Buffer Position : 0x10000200 (.bssとスタックに重ならないように)

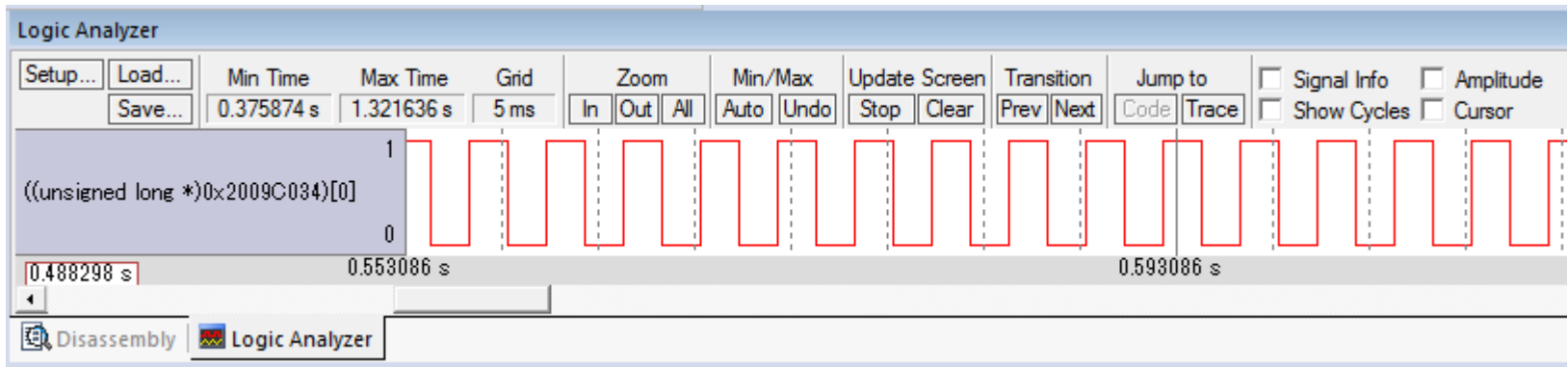


The screenshot shows the 'Instruction Trace' window in Keil. The 'Filter' is set to 'Execution'. The table below lists the instructions being traced, including their addresses, opcodes, and the instructions themselves. The source code is also visible on the right side of the table.

#	Address	Opcode	Instruction	Source
345	0x00000506	BD10	POP {r4,pc}	56: }
346	0x00000576	1B40	SUBS r0,r0,r5	
347	0x00000578	42A0	CMP r0,r4	
348	0x0000057A	D3FA	BCC 0x00000572	
349	0x00000572	F7FFF...	BLW us_ticker_read (0x000004...	29: while ((us_ticker_re
350	0x000004F4	B510	PUSH {r4,r}	51: uint32_t us_ticker_reac
351	0x000004F6	4804	LDR r0,[pc,#16] ; @0x00000508	52: if (!us_ticker_initd
352	0x000004F8	6800	LDR r0,[r0,#0x00]	
353	0x000004FA	2800	CMP r0,#0x00	
354	0x000004FC	D101	BNE 0x00000502	51: uint32_t us_ticker_reac
355	0x00000502	4802	LDR r0,[pc,#8] ; @0x0000050C	55: return LPC_SCT->X
356	0x00000504	6800	LDR r0,[r0,#0x00]	
357	0x00000506	BD10	POP {r4,pc}	56: }
358	0x00000576	1B40	SUBS r0,r0,r5	

デバッガの便利な機能を使う

- ロジックアナライザ機能でバーチャルLチカ（LPC1768でLED1:P1_18を使う場合）
 - ターゲットは、シミュレータ
 - [Logic Analyzer] - [Setup]から、[New]を選択し、
'`((unsigned long *)0x2009C034)[0]`'を追加
 - [Display Type] -> Bit
 - [Add Mask] -> `0x00040000`
 - [Shift Right] -> 18
 - **LA `((unsigned long *)0x2009C034)[0] & 0x40000 >> 18`**



まとめ

- オフラインコンパイラもmbed開発の選択肢のひとつ
- CMSIS-DAP対応ファームウェアとMDK-ARM無償評価版で、お手軽デバッグ
 - 本格的なデバッグが出来る
 - MTBでトレースも取れる