STM32F373VC 開発ボード

マニュアル

株式会社日昇テクノロジー

http://www.csun.co.jp

info@csun.co.jp

作成・更新日 2014/02/08

• 修正履歴

NO	バージョン	修正内容	修正日
1	Ver1.0	新規作成	2014/02/08

※ この文書の情報は、文書を改善するため、事前の通知なく変更されることがあります。最新版は弊社ホームページからご参照ください。「http://www.csun.co.jp」

目 录

1. 概要 5

1.1. チップ機能 5
1.2. ボード搭載機能 6

2. HARDWARE LAYOUT AND CONFIGURATION 7

2.1. POWER SUPPLY 7
2.2. BOOT OPTION 7
2.3. CLOCK SOURCE 7
2.4. RESET SOURCE 7
2.5. SD CARD 8
2.6. DEBUG PORT 8
2.7. BUTTON 8
2.8. LED 8
2.9. IIC EEPROM 8
2.10. IIC TEMPERATURE SENSOR 8
2.11. IIC PORT 8
2.12. SPI LCD 9
2.13. RS-232 9
2.14. USB 9
2.15. AUDIO OUT 10
2.16. AUDIO IN 10
2.17. POTENTIOMETER 10
2.18. LDR 10
2.19. EXPANSION PORT 10

3. ソフトウェア説明 10

3.1. KEIL コンパイル環境 10
3.1.1. コンパイラ環境構築 10
3.1.2. コンパイル環境設定 10

4. サンプルソースの説明 15

4.1. \CODE\STM32373C-EVAL_FW_V1.0.0\PROJECT フォルダのプログラムの説明 15
4.2. \CODE\STM32F37X_DSP_STDPERIPH_LIB_V1.0.0\PROJECT フォルダのプログラムの説明 31
4.2.1. \IOToggle\MDK-ARM 34
4.2.2. \IOToggle\MDK-ARM 34
4.2.3. \USART_Printf\MDK-ARM 34
4.2.4. \USART_HyperTerminal_Interrupt\MDK-ARM 35
4.2.5. \CAN_LoopBack\MDK-ARM 36
4.2.6. \CAN_Networking\MDK-ARM 36
4.2.7. \CAN_DualFIFO\MDK-ARM 37
4.2.8. \CAN_FIFOExtension\MDK-ARM 38

4.2.9. \ADC_Basic_Example\MDK-ARM 40
4.2.10. \ADC_DMA\MDK-ARM 41
4.2.11. \DAC_ADC\MDK-ARM 44
4.2.12. \DAC_SignalsGeneration\MDK-ARM 44
4.2.13. \COMP_LDR\MDK-ARM 46
4.2.14. \DMA_ADCToTIM3Transfer\MDK-ARM 48
4.2.15. \DMA_RAMToDACTransfer\MDK-ARM 48
4.2.16. \I2C_EEPROM\MDK-ARM 49
4.2.17. \I2C_TSENSOR\MDK-ARM 50
4.2.18. \SPI_MSD\MDK-ARM 51
4.2.19. \WWDG_Example\MDK-ARM 51
4.2.20. \IWDG_Example\MDK-ARM 51
4.2.21. \NVIC_IRQMask\MDK-ARM 52
4.2.22. \NVIC_WFIMode\MDK-ARM 52
4.2.23. \PWR_Standby\MDK-ARM 52
4.2.24. \PWR_Stop\MDK-ARM 52
4.2.25. \RTC_Calendar\MDK-ARM 53
4.2.26. \RTC_LSI \MDK-ARM 54
4.2.27. \RTC_Tamper\MDK-ARM 54
4.2.28. \RTC_Timer\MDK-ARM 55
4.2.29. \TIM_TimeBase\MDK-ARM 55
4.2.30. \CRC_32BitsCRCMessage\MDK-ARM 56

1. 概要

1.1. チップ機能

Core
- ARM 32-bit Cortex-M4 CPU
- 72 MHz max. frequency、 1.25 DMIPS/MHz
- FPU (floating-point unit)
- MPU (memory protection unit)

Memories
- 256 Kbytes Flash memory
- 32 Kbytes SRAM with HW parity check

Clock management
- 4 to 32 MHz crystal oscillator
- 32 kHz oscillator for RTC with calibration
- Internal 8 MHz RC with x 16 PLL option
- Internal 40 kHz oscillator

Calendar RTC
- Alarm、 periodic wakeup from Stop/Standby

Reset and supply management
- 2.0 to 3.6 V
- POR、 PDR and PVD

Low power
- Sleep、 Stop、 and Standby modes
- VBAT supply for RTC and backup registers

Debug mode
- serial wire debug (SWD)、 JTAG
- Cortex-M4 ETM

DMA
- 12-channel DMA controller
- Peripherals supported: timers、 ADCs、 SPIs、 I2Cs、 USARTs and DACs

Up to 3 x 16-bit Sigma Delta ADC、 up to 19 single/ 10 diff channels
1 x 12-bit、 1 us ADC with separate analog supply from 2.4 V to 3.6 V
Up to 2 fast rail-to-rail analog comparators
Temperature sensor
Up to 3 x 12-bit DACs
Support for up to 18 capacitive sensing keys
Up to 84 fast I/O ports
- mappable on external interrupt vectors
- several 5 V-tolerant

14 timers
- 2 x 32-bit timer and 3 x 16-bit timers
- 2 x 16-bit timers with up to 2 IC/OC/PWM or pulse counter
- 4 x 16-bit timers with up to 1 IC/OC/PWM or pulse counter
- 2 x watchdog timers (independent、 window)
- SysTick timer: 24-bit downcounter
- 3 x 16-bit basic timers to drive the DAC

Communication interfaces
- CAN interface (2.0B Active)
- USB 2.0 full speed interface
- 2 x I2C with 20 mA current sink to support Fast mode plus
- Up to 3 USARTs (ISO 7816 interface、 LIN、 IrDA capability、 modem control).
- Up to 3 SPIs、 with muxed I2S
- CRC calculation unit、 96-bit unique ID
- CEC bus interface

1.2. ボード搭載機能

- 20 ピン 2.54 ピッチ JTAG インタフェース
- USB2.0 full-speed slave インタフェース
- RS232 インタフェース
- CAN2.0B インタフェース
- SD メモリーカードインターフェース (SPI)
- オーディオ出力インタフェース (C43L22、音質が良い)
- MIC インターフェース
- 4 つ制御可能な LED
- 2 つのユーザーボタン
- 1 つの 5 方向のナビゲーションボタン
- IIC インタフェースの EEPROM
- IIC インタフェースの温度センサー
- 四ピン IIC インターフェース
- 6 ピン UART インターフェース (TTL)
- １つの液晶インターフェース(SPI)
- １つの光センサー
- １つのボタン電池ホルダー（ボタン電池なし）
- １つの PT100 温度センサー取付ポート
- １つの圧力センサー取付ポート
- １つの ECG モジュールの取り付けポート

利用していない I / O は 2.54mm ピッチのピンコネクタで引出

2. Hardware layout and configuration

2.1. Power supply

給電方法：

1. DC5V 電源給電。
2. MiniUSB で給電する。

2.2. Boot option

開発ボードの起動モードは下記の 3 つの方法がある：

- ユーザプログラムスペースから起動
- ISP ブートプログラム（Boot loader）スペースから□起動
- 内部 SRAM スペースから起動

上記の 3 つの起動モードはボードの SW1 とユーザーレジスタの Bit12 で設定する：

表 1、ブート関連のジャンパー：

BOOT 0(SW1)	接続	Boot1(Bit12)	起動モード
0	1 2 3 ● ● ●	無視	ユーザプログラムスペースから起動
1	1 2 3 ● ● ●	1	内部 SRAM スペースから起動
1	1 2 3 ● ● ●	0	ISP ブートローダ（Boot loader）スペースから起動し

2.3. Clock source

開発ボードに 2 つのクロックソースがある：

- Y1、32.768K　STM32F373VC チップの RTC 水晶発振器
- Y2、8MHz　　STM32F373VC チップのメイン頻度水晶発振器

2.4. Reset source

開発ボードのリセット信号は、ローレベルリセットで、下記の方法で実現する：

- RESET キー
- JTAG インタフェースの JP1-15Pin
- JP3-26Pin

2.5. SD card

開発ボードは SPI インタフェースで SD カードと接続。通常 I/ O PE3 で SD カードスロットの状態を検出する（SD カード存在するか）

2.6. Debug Port

開発ボードに 20 ピン 2.54mm ピッチのコネクタを搭載、市販の J-link と U-link2 インタフェースをサポート、通信モードは SW モード或いは JTAG モード。

2.7. Button

開発ボード、3 つの独立キーと 1 つの 5 方向のナビゲーションボタンがある。

表 2. キーファンクション

キー	ファンクション
RESET	リセット
USER	押す：ローレベル、通常：ハイレベル、外部割り込みモードに設定可能
TAMPER	押す：ローレベル、通常：ハイレベル、外部割り込みモードに設定可能
S4	電池ホルダー向きのは UP キー、SW1 向きのは LEFT キー

2.8. LED

開発ボードに 5 つの LED がある。緑色の CAN コネクタの隣には電源状態 LED、5 方向のナビゲーションボタンの隣には 4 つの通常 I/O 制御 LED。

2.9. IIC EEPROM

開発ボードに 1 つの I2C バス EEPROM を搭載し、I2C2 を介し接続する、デバイスアドレスは 0xA6。

2.10. IIC Temperature Sensor

開発ボードに 1 つの I2C バス温度センサーを搭載し、I2C2 を介して接続する、デバイスアドレスは 0x90。

2.11. IIC Port

J7 は IIC 拡張ポート、外部 IIC モジュール接続或いは 2 つの開発ボードの IIC 通信実験に使用する。

表 3 J7 ピン定義

pin 番号	ファンクション
Pin1	GND
Pin2	+3.3V
Pin3	I2C2_SCL
Pin4	I2C2_SDA

2.12. SPI LCD

JP2 は SPI 通信モードの LCD インタフェース、2.8 インチの LCD 拡張ボードを接続する。

表 4 JP2 ピン定義

pin 番号	ファンクション
Pin1-4	使用しない
Pin5	PC11、SPI3_MISO
Pin6	LCD バックライト制御、プルアップ抵抗と直結
Pin7、8	+3.3V
Pin9	PD2、LCD_CS
Pin10	LCD リセットピン
Pin12	PC12、SPI3_MOSI
Pin13、14	GND

2.13. RS-232

開発ボードの J4 DB9 は RS-232 インタフェース。USART2 ピンとリンクする。J15 は USART2 TTL レベルを引き出したインタフェース、使用する時は、R11-R14 の 0 オーム抵抗を取り除く必要がある。

表 5 J15 ピン定義

pin 番号	ファンクション
Pin1	PD6、USART2_RX
Pin2	PD5、USART2_TX
Pin3	PD4、USART2_RTS
Pin4	PD3、USART2_CTS
Pin5	+3.3V
Pin6	GND

2.14. USB

J1 は full speed USB2.0 slave インタフェース、MiniUSB ケーブルを介してホストコンピュータと接続する。PC5 は、D+データラインのプルアップを制御する。D+はハイレベルの時、ホストコンピュータは、新しいデバイスが接続されていると認識する。U2 は ESD デバイスがホットスワップする時の静電保護機能。

2.15. Audio Out

開発ボードのオーディオチップは CS43L22。J3 はヘッドフォン出力、3.5 インチのヘッドホンプラグをサポートする。SP1 はスピーカーインターフェイス、普通の 8 オームスピーカーを使用できる、音質は普通。

2.16. Audio In

開発ボードは、マイクインターフェース回路があり、録音記録実験ができる。

2.17. Potentiometer

開発ボードに１つの 10K の高精度な可変抵抗があり、PB1 の ADC9 と接続、対応するコンポーネントは R39。

2.18. LDR

開発ボードに１つの感光抵抗を搭載して、CPU はコンパレータで外部 CPU の光強度を判断する。

2.19. Expansion Port

開発ボードに一部の I/O を JP3、JP4 に引き出す。

3. ソフトウェア説明

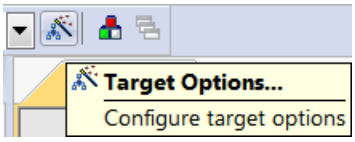
3.1. Keilコンパイル環境

3.1.1. コンパイラ環境構築

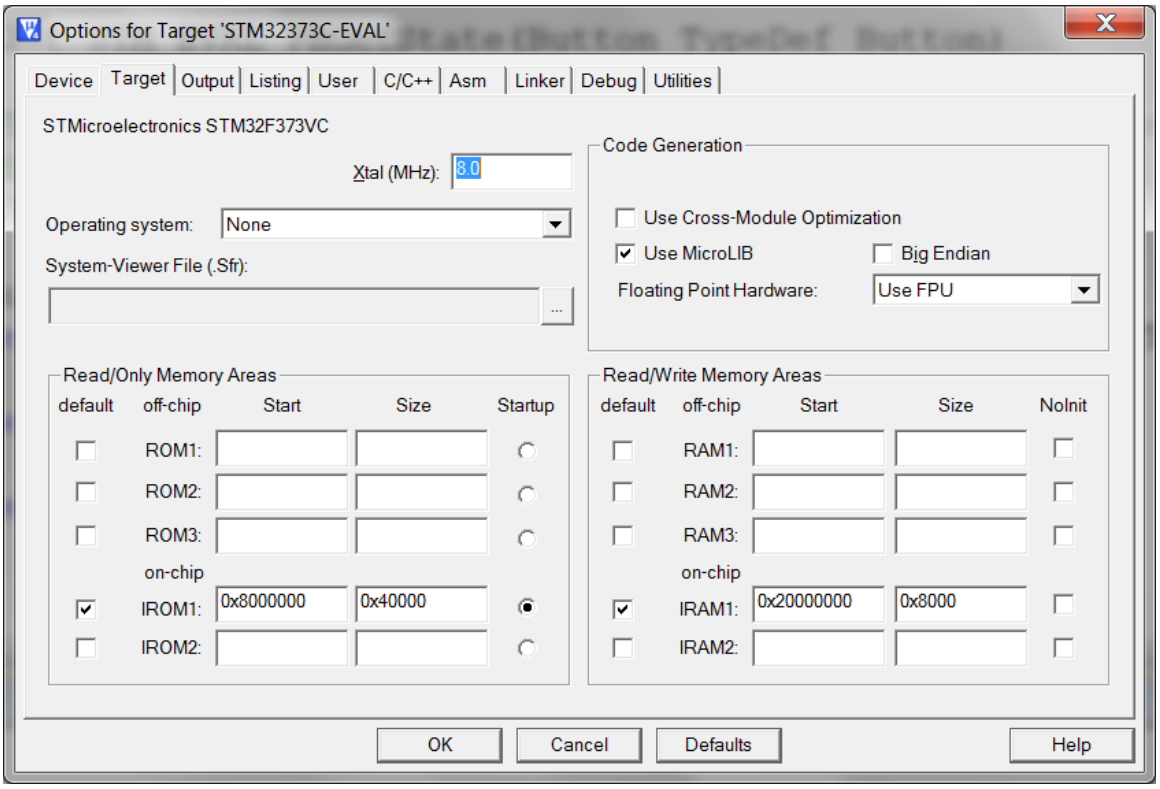
- libを使用する場合バージョン関連エラーがある可能性がありますので、MDK453.exeバージョンをお勧め。
- コンパイラは C ドライブのルートディレクトリ (C:¥Keil¥ARM) 下にインストールする事をお勧め、でなければコンパイルエラーの可能性があります。

3.1.2. コンパイル環境設定

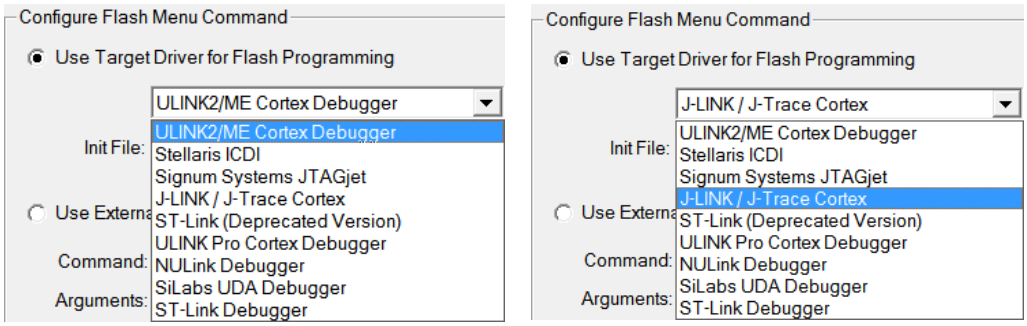
- 一例をオープンし、` Options for Target` をクリック



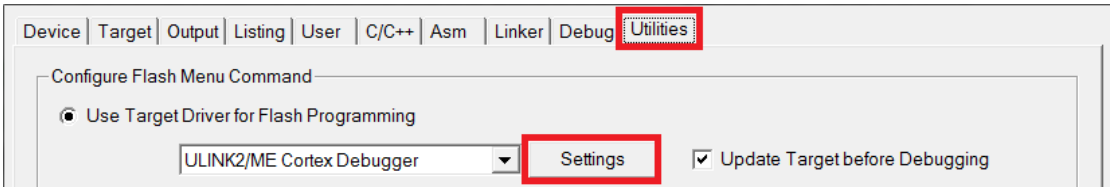
◆　下記のウィンドウ通り：

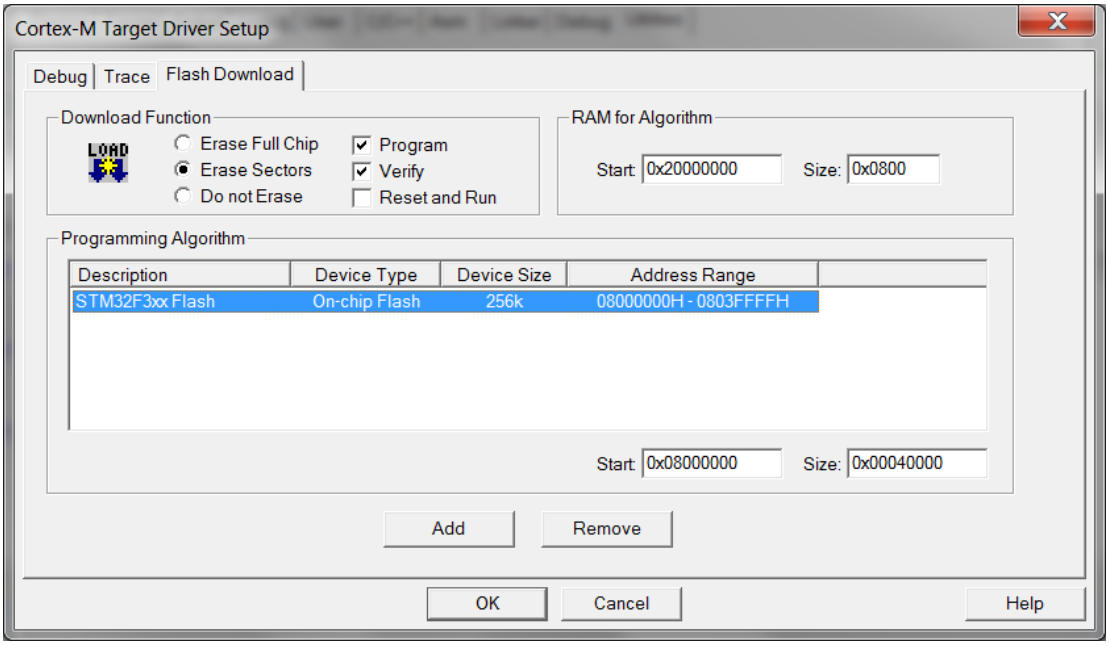


◆　〝Utilities〟を選択、下記図のように〝Use Target Driver for Flash Programming〟でエミュレータの種類を選択、ULINK2エミュレータを使用する場合、〝ULINK2/ME Cortex Debugger〟を選択する；JLINK V8エミュレータを使用する場合、〝J-LINK/J-Trace Cortex〟を選択する。。

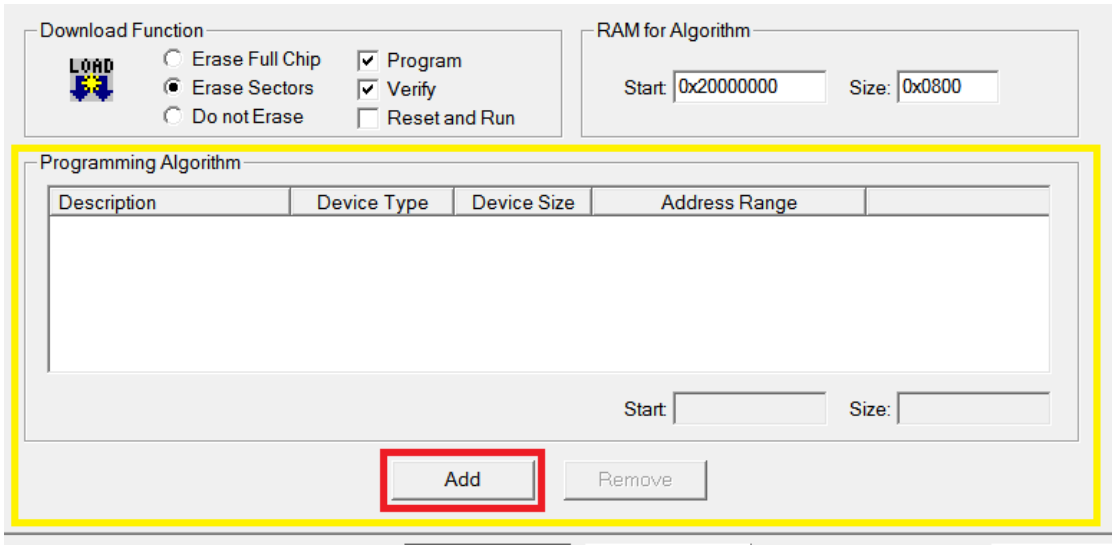


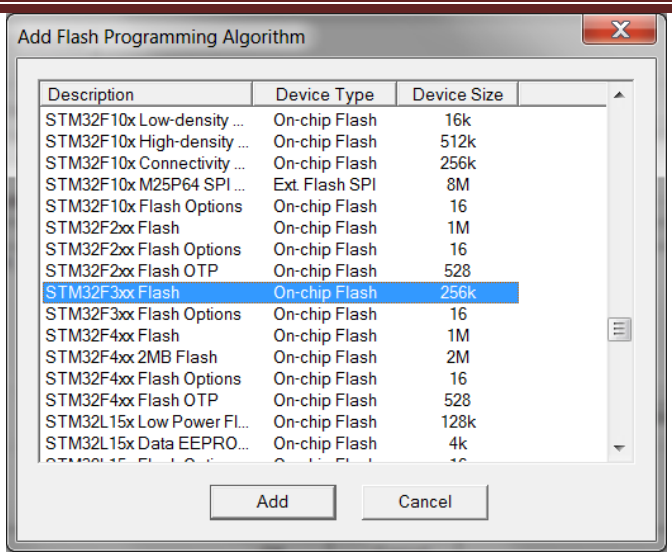
◆　〝Settings〟をクリック、エミュレータパラメータ設定に入る。エミュレータによって、設定インターフェースの異なるが、原理は同じである。ULINK2バージョンエミュレータの設定について説明する。



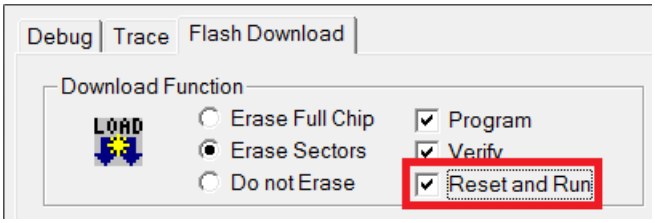


◆ ｀Flash Download｀オプション下の｀Programming Algorithm｀リストに開発ボード必要なデバイスがない場合、｀Add｀で追加する。

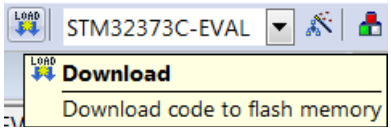




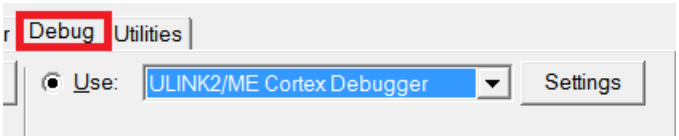
◆ ｀Reset and Run｀有効する場合、ULINK2バージョンエミュレータはプログラムをダウンロード完了後、直接リセット・実行する。

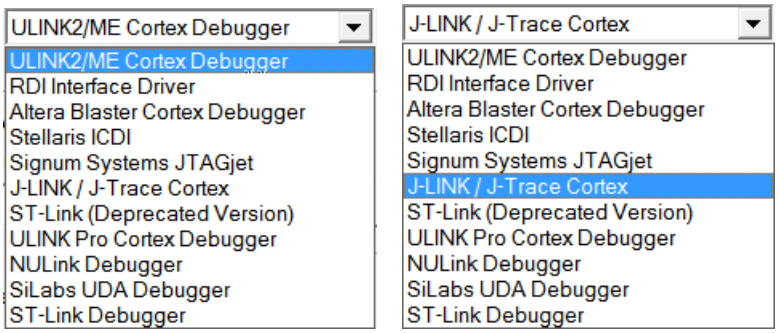


◆ 設定完了後、｀Download to Flash Memory｀をクリック、エミュレータでプログラムをダウンロードできる。

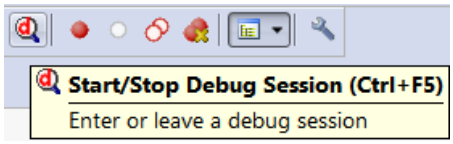


◆ 次はエミュレータの設定、｀Options for Target｀を選択、ULINK2エミュレータは｀ULINK2/ME Cortex Debugger｀；JLINK V8エミュレータは｀J-LINK/J-Trace Cortex｀を選択する。

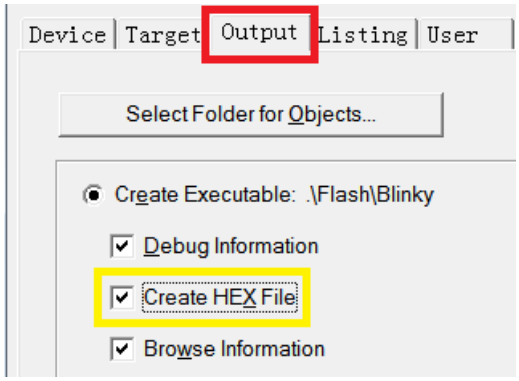




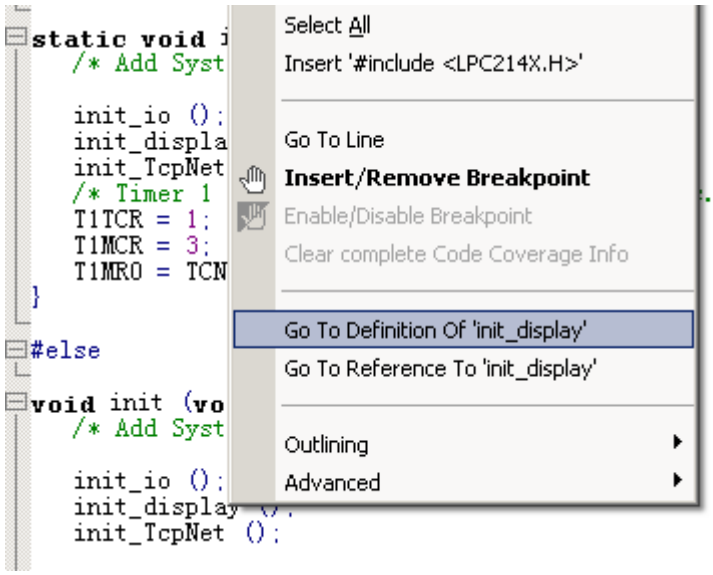
◆ ここまで、エミュレータでシミュレーションプログラムをダウンロードできる。



◆ hexフォーマットファイルが必要な 場合、`Create HEX File`を有効にする。



◆ コードをデバッグするため、`Browse Information`も有効にすると勧める。例えばプログラムにinit_display関数を呼び出す時、関数のいずれかのフィールドでマウスを右クリックし、`Go To Definition Of init_display`を選択、ソフトウェアは自動的に当関数の実体にジャンプする。



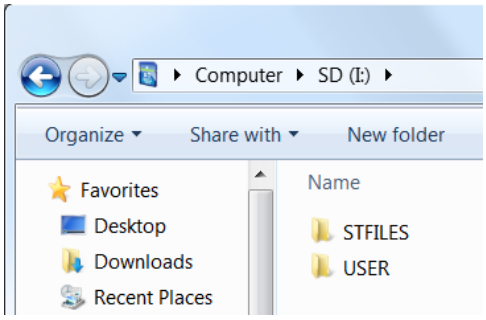

```
/*--------------------------- init_display ------------
static void init_display () {
  /* LCD Module.2x16 init*/

#ifdef USE_4BIT_LCD
  LCD_init ();
  LCD_cur_off ();
  upd_display ();
#endif
}
```

4. サンプルソースの説明

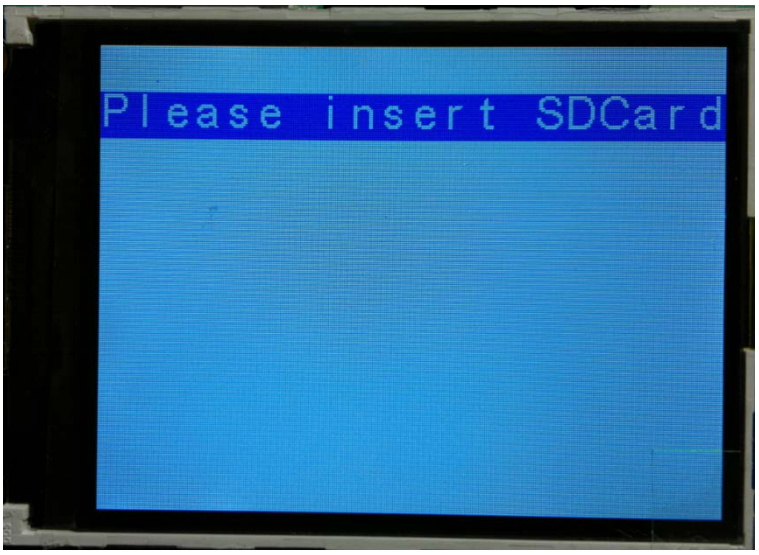
4.1. ¥Code¥STM32373C-EVAL_FW_V1.0.0¥Projectフォルダのプログラムの説明

開発ボードに STM32373C-EVAL¥MDK-ARM ディレクトリ下のプログラムは、プロセッサの大部分の機能を示す。

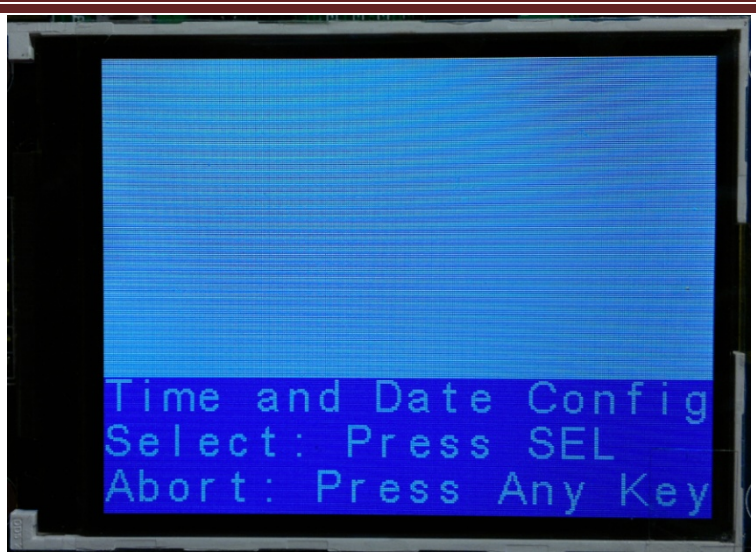
- 液晶スクリーンで表示するアイコンは SD カード内に保存する。テストのため、¥Code¥Media ディレクトリ下の 2 つのフォルダ（STFILES、USER）を SD カードのルートディレクトリにコピーする必要：



- SD カードを検出されない場合、下記の情報を表示する：



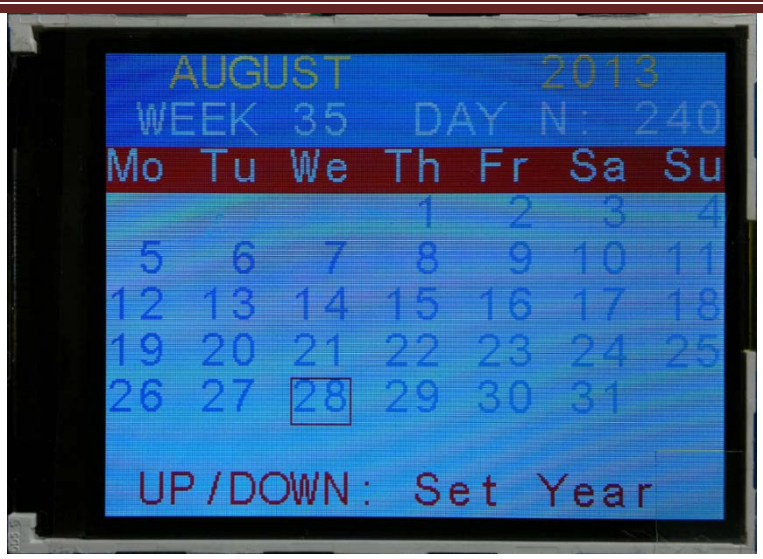
- SD カード（(STFILES、USER）コピー済み）を SD カードスロットに入れ、5 方向のナビゲーションボタンの上方向キーを押しプログラムをリセットする。次はスクリーンの提示に従い、5 方向のナビゲーションボタンの真ん中のキーを押し、時間設定に入る、他キーで設定を終了する。



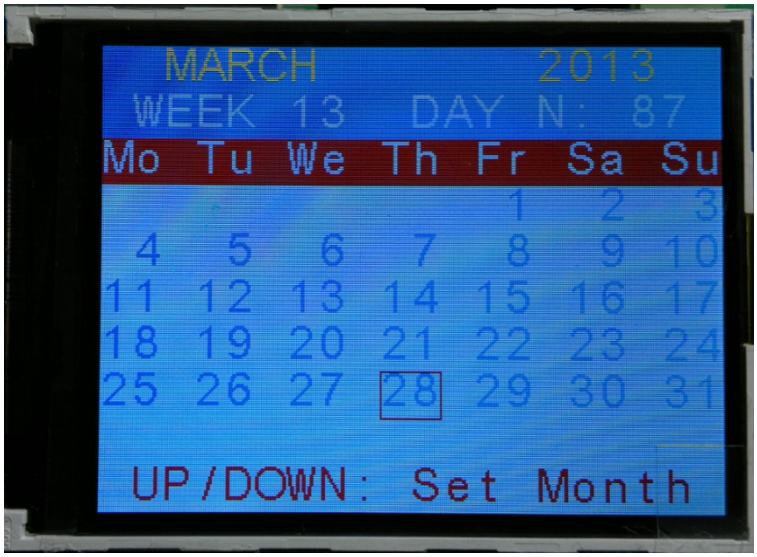
➢ 真ん中のキーで時間設定に入る(ボタン電池を装填しない場合、設定は保存できない)：

➢ 上方向キーで数字をインクリメント と下方向キーで数字をデクリメント、中間（確定）キーで確認：

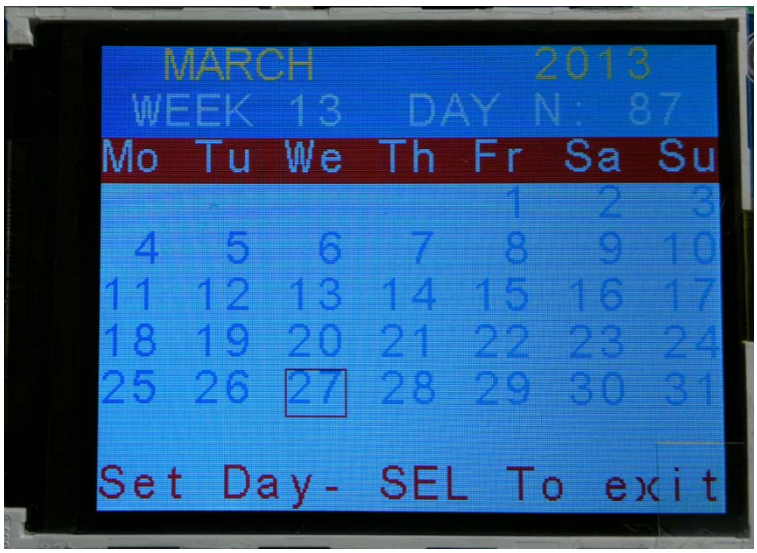
➢ 確定キーで年設定に入り、上/下方向キーで調節し、確認…下記図を 2013 年と設定する：



➢ 月設定に入り、操作は上記と同じ。下記図は MARCH と設定する：



➢ 日時も同じ、上、下、左、右方向キーで調節、中間キー確認。下記図は 27 日 We（水曜日）と設定する：



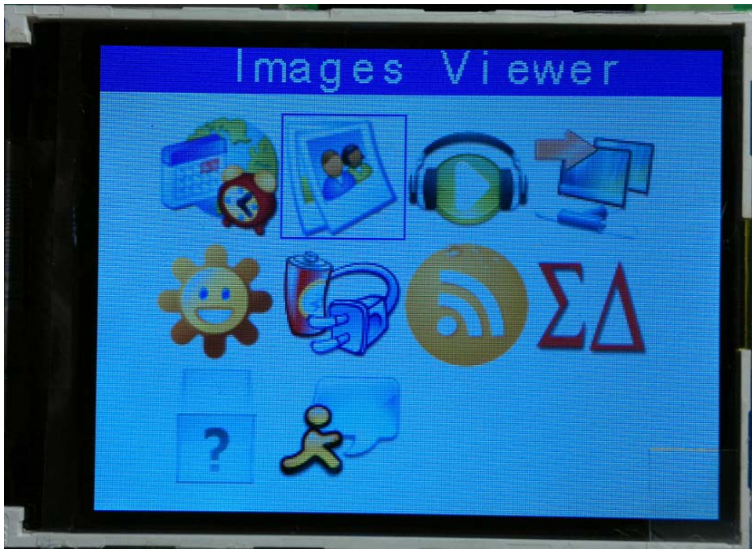
➢ 全ての時間設定完了後、スクリーンの表示は下記の通り：

➢ 方向キーで Organizer を選択、中間キーでオプションに入る：

➢ 上、下方向キーで Time、Date、StopWatch、LapTimer、AlarmA を切り替え、Adjust は現在時間設定、Show は現在時間表示、Return は戻る：

- Show は現在時間表示、開発ボードの User キーで終了する。

- メインメニューで Images Viewer を選択、サブメニューに入る：

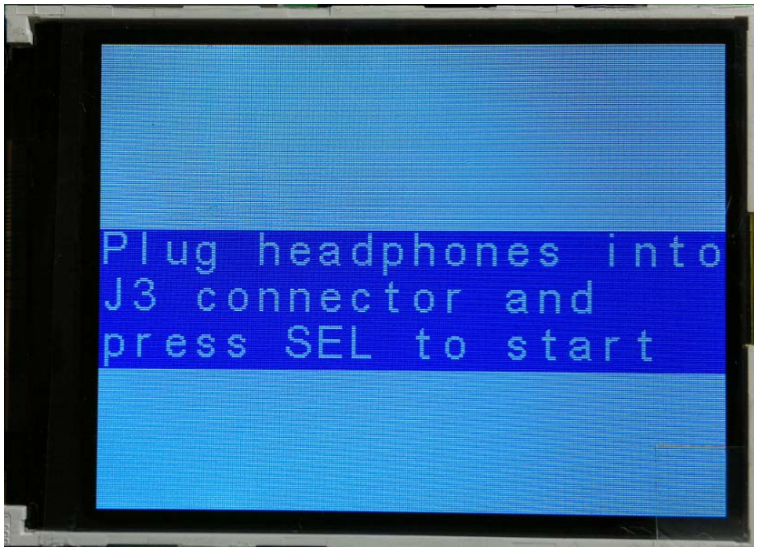


- サブメニューに Images Viewer を選択、画像表示に入る、Return は戻る。

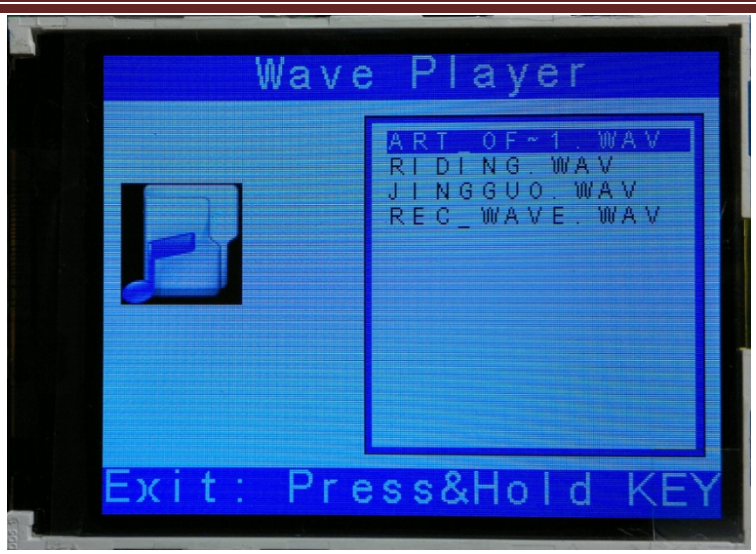
➢上/下方向キーで画像を選択、表示する。画像は SD カードの USER ディレクトリ下に保存する。ピクセル 320x240、USER キーで終了する。

➢メインメニューで Audio を選択、サブメニューに入る：WavePlayer は wav ファイルを再生；WaveRecorder は録音して SD カードに保存。

- WavePlayer を選択、wav オーディオファイルを再生する(J3 に 3.5 インチヘッドホンプラグを挿入し、中間キーを押す)



- 上/下方向キーで WAV ファイルを選択し、中間キー（確定）でオーディオファイルを再生する。USER キーで終了する。

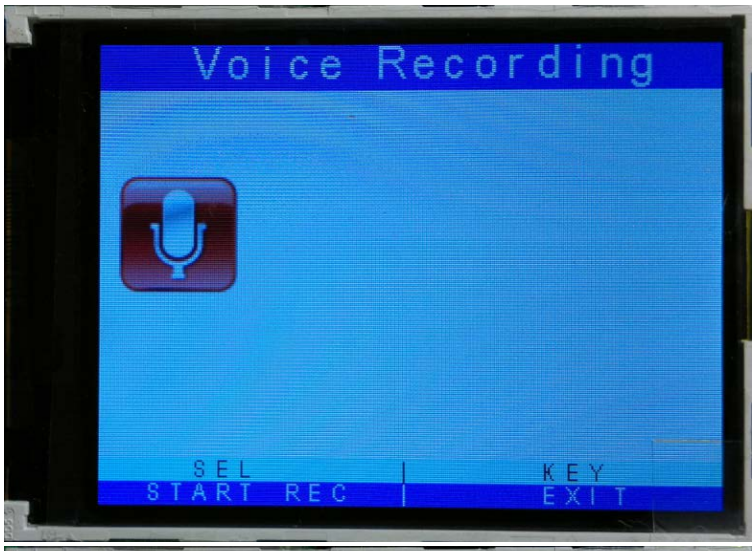


- 左/右方向キーで音量調節、中間キーで再生/一時停止する。UESR キーで終了する。WAV ファイルは SD カードの USER ディレクトリ下に保存する。

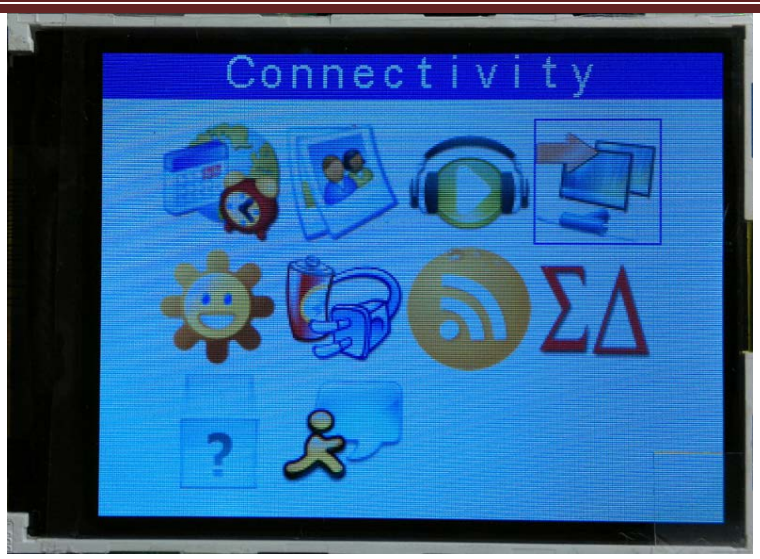
注：
本プログラムは WAV ファイルしか識別できない。

- WaveRecorder で録音する。中間キーで入る。

- 中間キーで録音開始、オーディオファイルは SD カードの USER ディレクトリ下に保存する。ファイル名は REC_WAVE.wav。

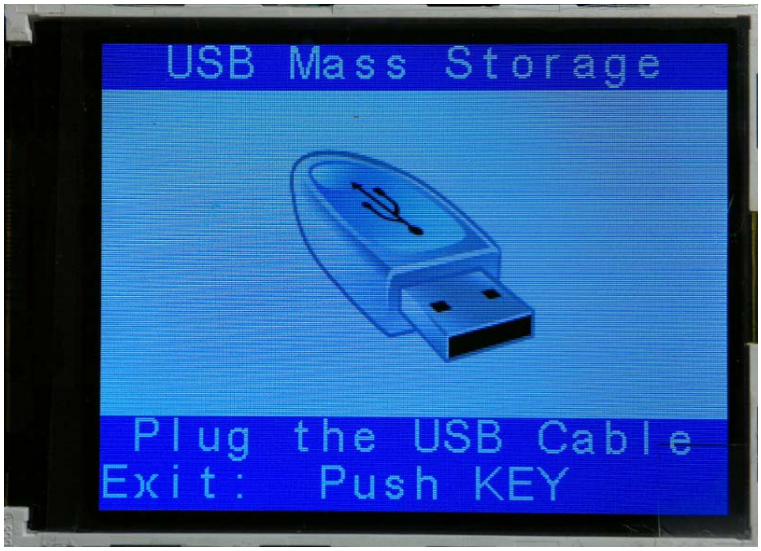


- メインメニューで Connectivity を選択、サブメニューに入る。



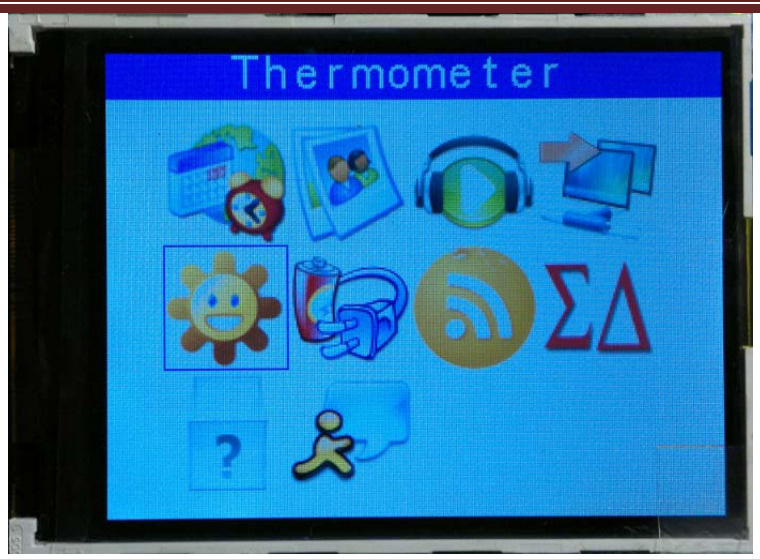
- 下記図のように、Connectivity メニューで HDMI CEC、IR Transmitter、IR Receiver、Mass Storage のオプションがある。前の 3 つのオプションは外部デバイスが必要とする。ここでは Mass Storage の機能をテストする。Mass Storage を選択、USB 大容量デバイステストに入る。

- 下記図の通り、Start でテストを開始する。

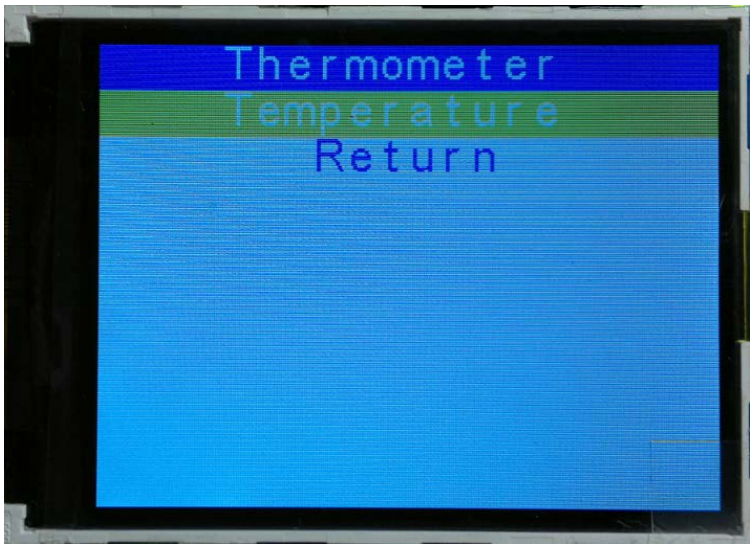


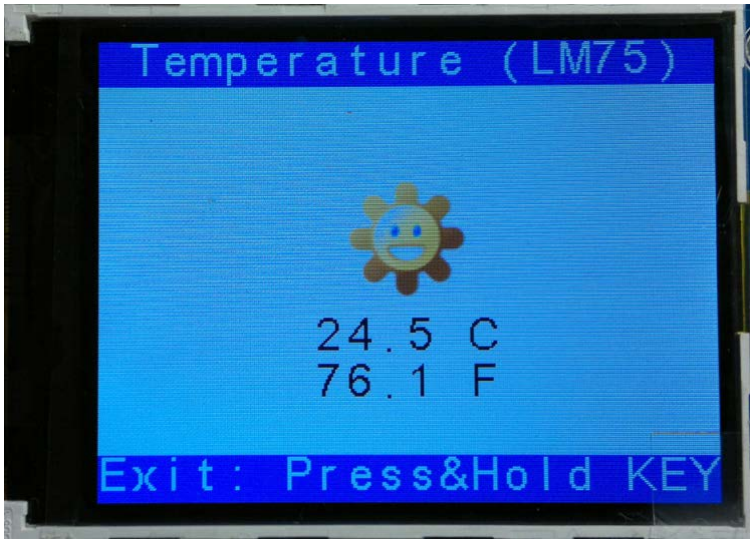
- Mini USB ケーブルを J1 に接続、ホストコンピュータは、新しいデバイスが接続されたと表示し、自動的ドライバをインストールする。完了後、新しいリムーバブルデバイスは確認できる。

- メインメニューに戻って Thermometer（温度計）を選択、サブメニューに入る：

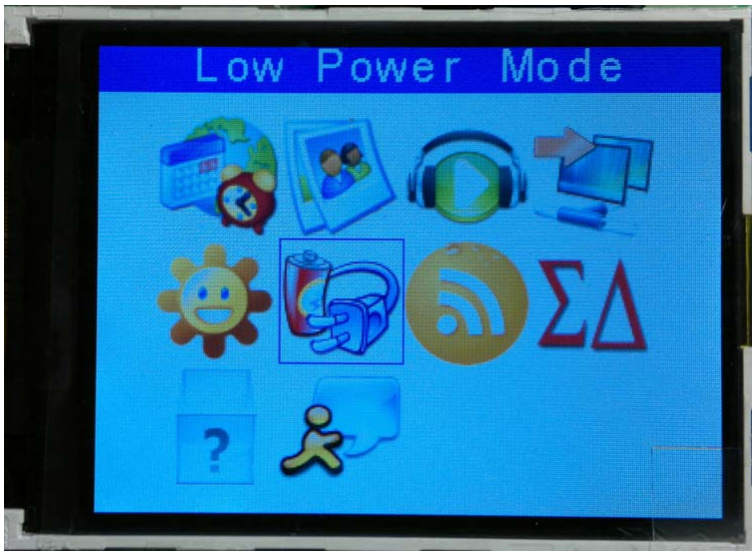


➢ Temperature を選択し、現在温度表示、指先で開発ボードの U8 チップを押さえるとスクリーンの表示温度も変化する。USER キーで終了する。

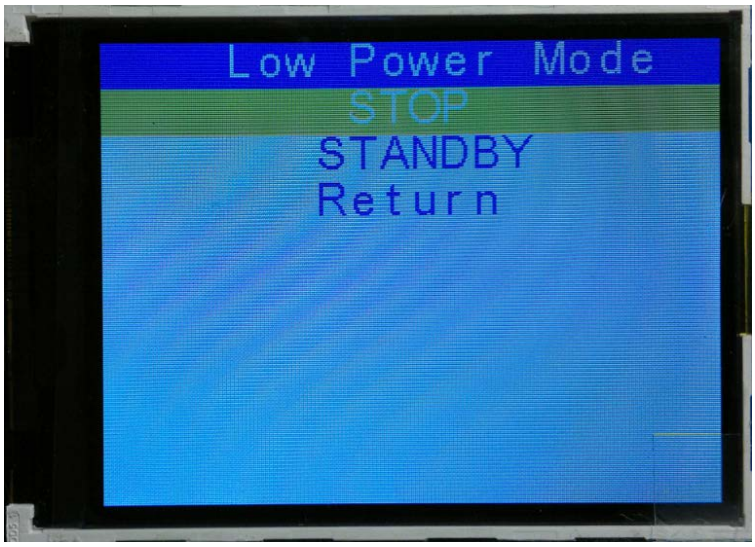




➢ メインメニューで Low Power Mode（省電力モード）を選択、サブメニューに入る：



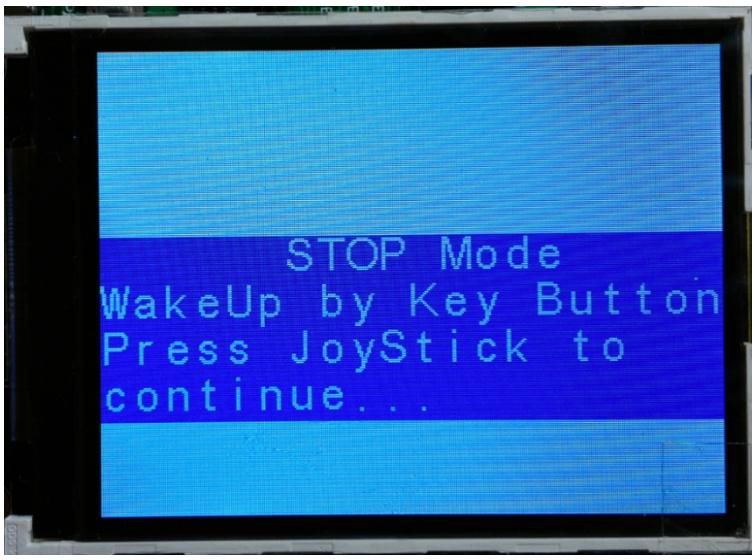
- STOP は停止モード（ストップ）、STANDBY は待機モード（スタンバイ）、Return で戻る：



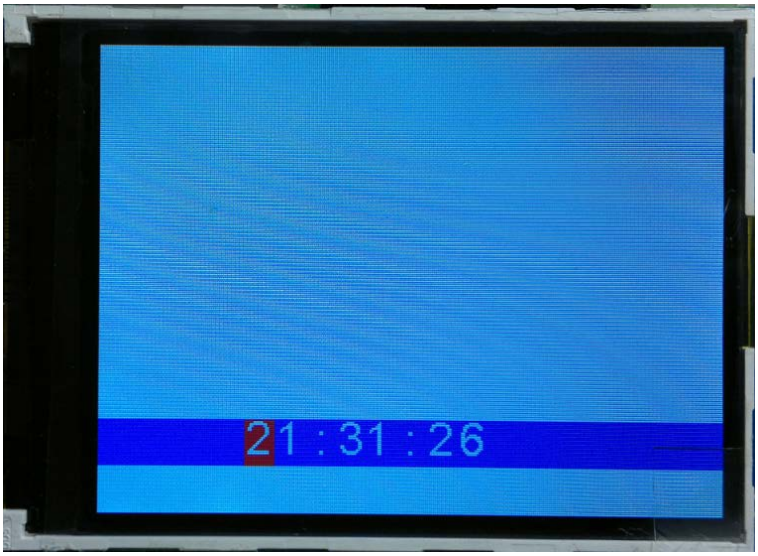
- モードを選択し、そのサブメニューに入る。例えば STOP モード、スクリーンに下記のオプションが表示する。EXIT は外部キーで STOP モードを終了；RTC Alarm は RTC 割り込みで STOP モードを終了する：

- EXTI を選択、確認すると、プロセッサは STOP モードに入り、開発ボードの 4 つの LED が消灯し、5 方向ナビゲーションボタンも反応しなくなる。

- 開発ボードの USER キーを押し、STOP モードを終了する。4 つ LED がフラッシュし、スクリーンの提示に従い､5 方向ナビゲーションボタンの任意キーでSTOPモードのサブメニューから上級メニューに戻る。

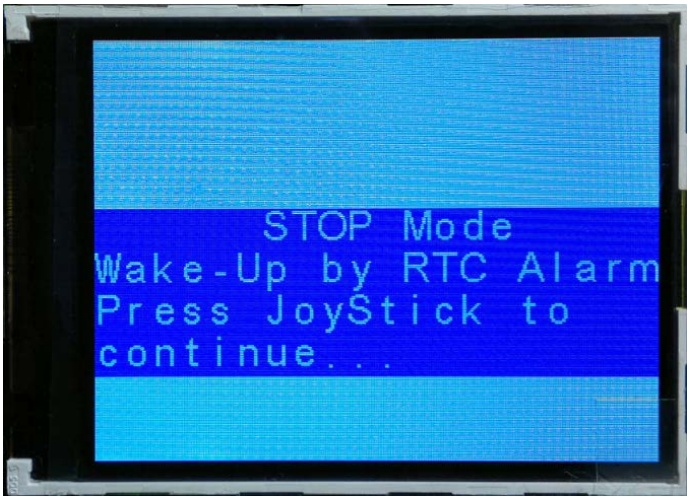


- STOP モードのサブメニューで RTC Alarm を選択、RTC ウェイクアップ時間設定に入る：

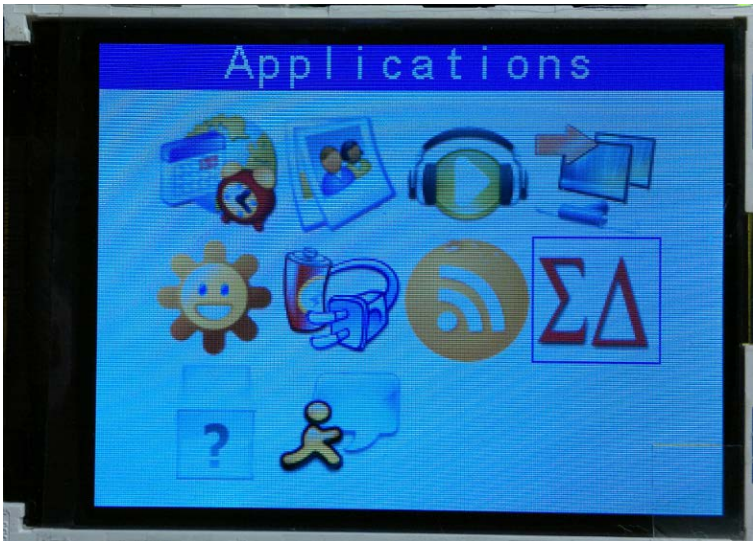


- 上、下、中間キーでウェイクアップ時間を設定する。実験のため、1 分後に設定する：
- RTC ウェイクアップ時間設定後、中間キーを押し STOP モードに入り、全てのキーは反応がなくなる。

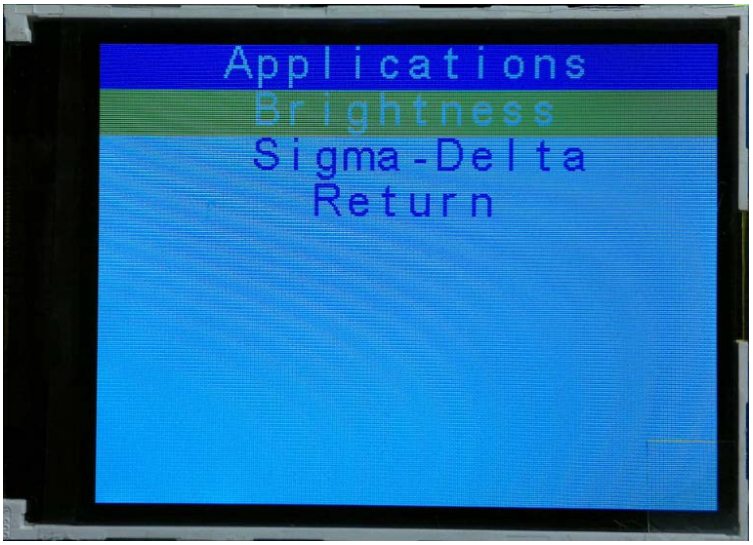
- RTC の現在時間が設定時間と一致すると、LED はフラッシュし、スクリーンの提示に従い、5 方向ナビゲーションボタンの任意キーで STOP モードのサブメニューから上級メニューに戻る。



➢ メインメニューで、方向キーApplications（アプリケーション）を選択



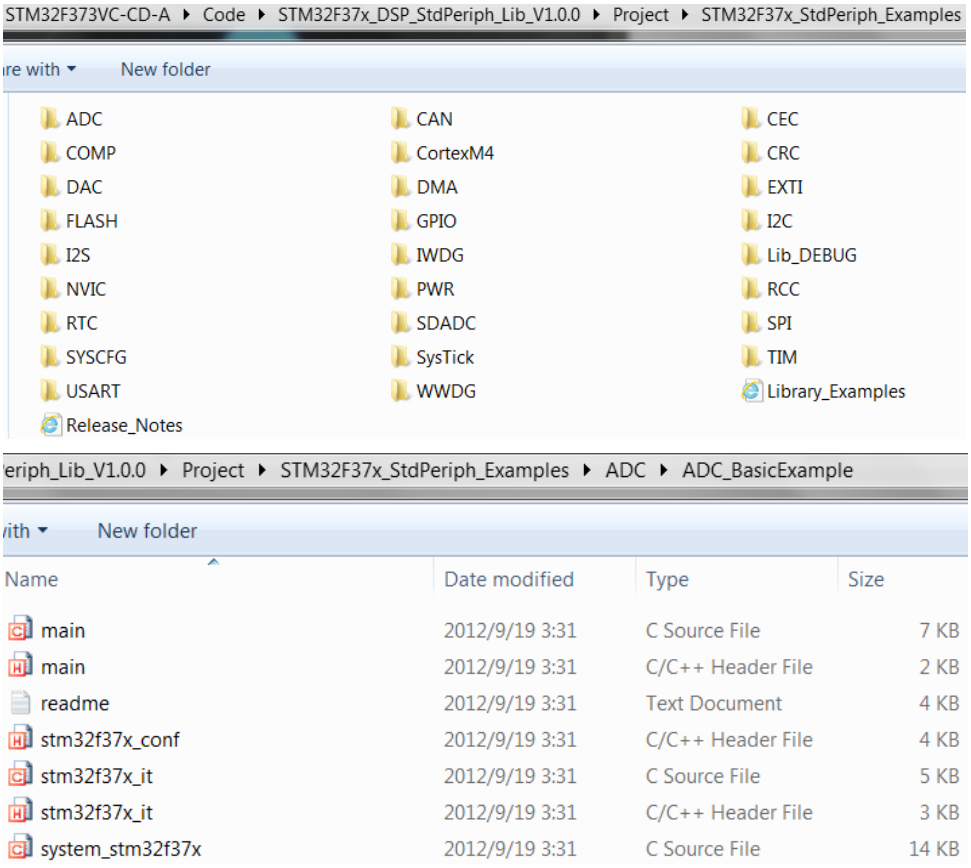
➢ サブメニューに入る。Brightness は光センサーテストプログラム、Sigma-Delta は ECG、PT100 は温度センサー、圧力センサーのテストプログラム（外部デバイスが必要とする）。ここでは光センサーテストをテストする。



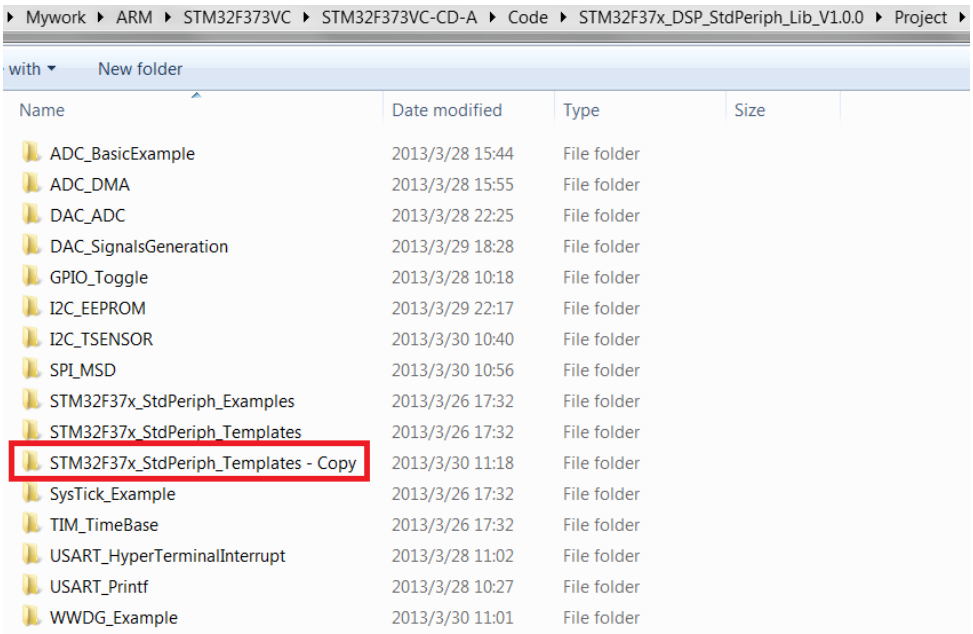
➢Applications 名メニューで Brightness を選択、光センサーのテストプログラムに入る。プロセッサは光センサーR2 上の光度の強弱でスクリーンの表示状態を変更する。USER キーで終了する：

4.2. ¥Code¥STM32F37x_DSP_StdPeriph_Lib_V1.0.0¥Projectフォルダのプログラムの説明

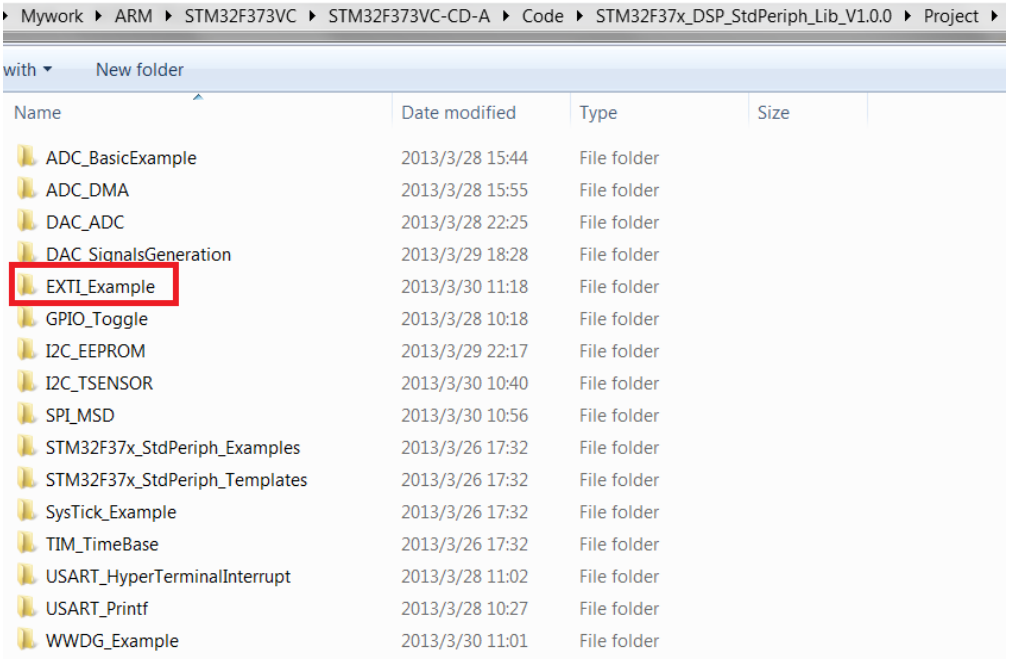
➢ 本フォルダのプログラムは STM32F37x_StdPeriph_Examples フォルダに保存し、スペース節約のため、全ての底層ドライブはソースコードで保存する：



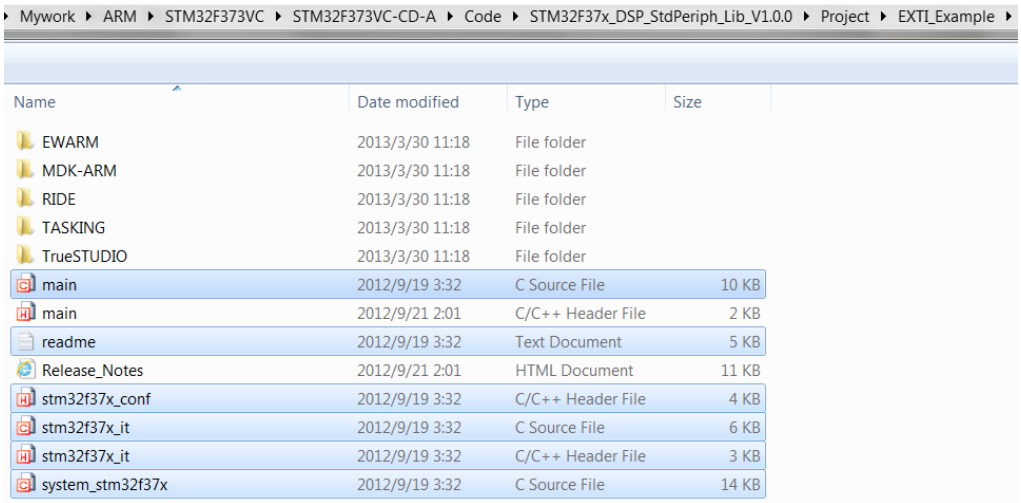
- STM32F37x_StdPeriph_Templates をテンプレートとして、底層レベルのドライバ使用する時に、STM32F37x_StdPeriph_Templates テンプレートをコピーし、ディレクトリ¥Code¥STM32F37x_DSP_StdPeriph_Lib_V1.0.0¥Project 下に保存する。



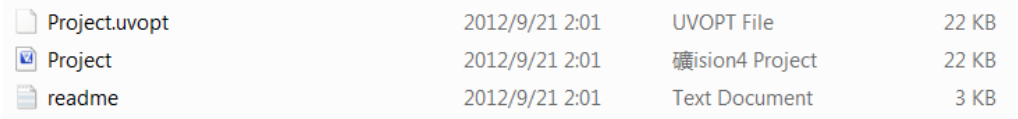
- 例えば今回¥Code¥STM32F37x_DSP_StdPeriph_Lib_V1.0.0¥Project¥STM32F37x_StdPeriph_Examples¥EXTI¥EXTI_Example ディレクトリ下のプログラムをテストするには、テンプレートフォルダを EXTI_Example にリネームする。



- 次に¥Code¥STM32F37x_DSP_StdPeriph_Lib_V1.0.0¥Project¥STM32F37x_StdPeriph_Examples¥EXTI¥EXTI_Example ディレクトリ下の全てのファイルを¥Code¥ STM32F37x_DSP_StdPeriph_Lib_V1.0.0¥Project¥EXTI_Example ディレクトリ下にコピーする。



- 本ディレクトリ下の MDK-ARM フォルダにテストプログラムのプロジェクトファイルが保存される：



- プロジェクトファイルをコンパイルする時、関数が不足な場合は¥Code¥STM32F37x_DSP_StdPeriph_Lib_V1.0.0¥Utilities¥STM32_EVAL¥STM32373C_EVAL ディレクトリ下の内容とコンパイラのエラー情報によって、対応ファイルを追加する。

4.2.1.　¥IOToggle¥MDK-ARM

開発ボード上の LED1、LED2 をフラッシュ制御するプログラムである。プログラムは交替的に BSRR 及び BRR レジスタに値 0x0003（0000　0000　0000　0011）を与え、PC0、PC1 ピン状態を 0、1 の交替変化する、I/O＝0 の時 LED 点灯。

4.2.2.　¥IOToggle¥MDK-ARM

本ディレクトリ下のプログラムは外部割り込み機能を利用。プログラムは外部割り込みピンを立ち上がりエッジ或いは立ち下がりエッジをトリガーにして、対応するピン割り込みサービス･ルーチンは開発ボードの LED を反転動作させる。

表 6. 割り込みピン定義

ピン	対応キー	割り込みソース	割り込み型	LED 各桁反転
PA0	WK_UP	EXTI0	立ち上がりエッジ	LED1
PA2	USER	EXTI2	立ち下がりエッジ	LED4
PE6	S4-Center	EXTI6	立ち上がりエッジ	LED2
PE10	S4-UP	EXTI10	立ち上がりエッジ	LED3

4.2.3.　¥USART_Printf¥MDK-ARM

本ディレクトリ下のプログラムは stdio.h の printf 関数を USART にリダイレクトし、ハイパーターミナルで｀USART Printf Example: retarget the C library printf function to the USART｀をプリントアウトする。シリアルポートボーレートは 115200、ハードウェアフロー制御なし。

➢ リダイレクトは下記の通り：

```
#define PUTCHAR_PROTOTYPE int fputc(int ch、 FILE *f)

PUTCHAR_PROTOTYPE
{
  //PUTCHAR_PROTOTYPE 関数にシリアルポート送信関数をパッケージする
USART_SendData(EVAL_COM1、 (uint8_t) ch);
  // データ送信完了待ち
  while (USART_GetFlagStatus(EVAL_COM1、 USART_FLAG_TXE) == RESET)
  {}
  return ch;
}
```

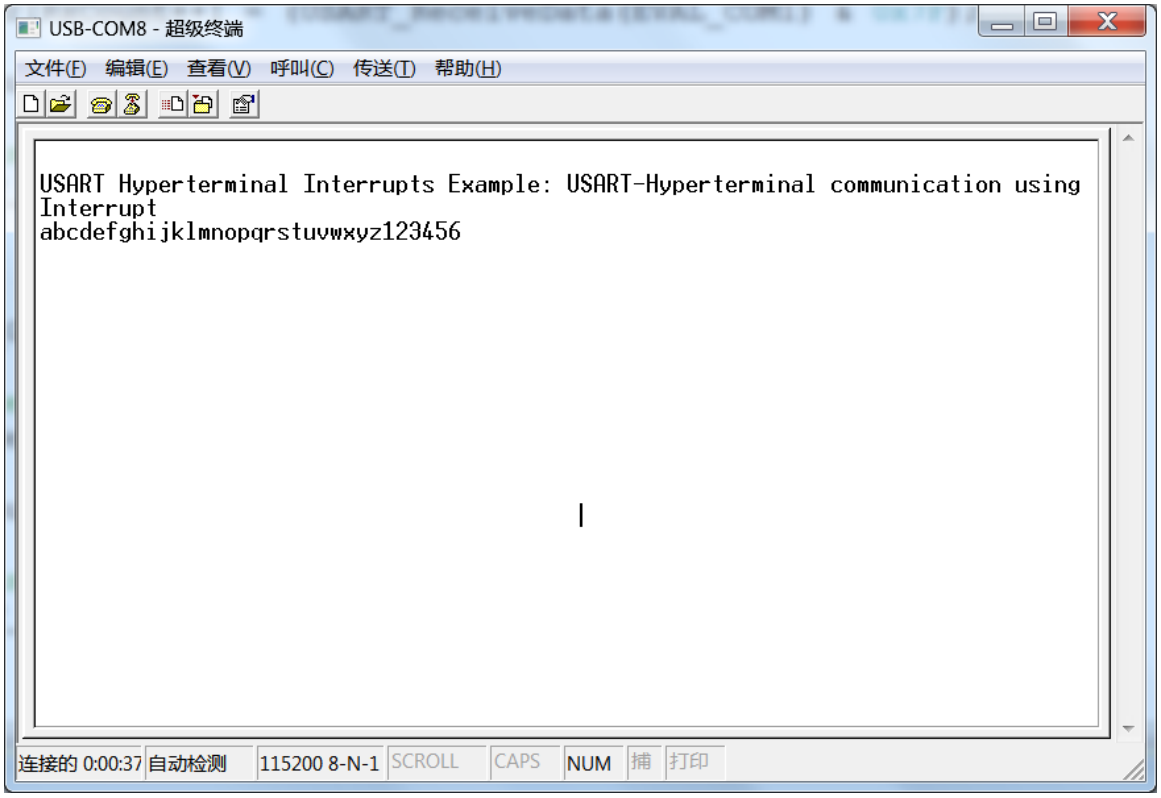
4.2.4. ¥USART_HyperTerminal_Interrupt¥MDK-ARM

本ディレクトリ下のシリアルポートテストプログラムは割り込み方式でデータを受送信する。プログラムはハイパーターミナルから送信した文字をハイパーターミナルに返送する。受信文字数は 32 個を超えると、LED1、LED2 消灯、プログラムは無限ループに入る。

➢ 割り込みプログラムは stm32f37x_it.c の USART2_IRQHandler：

```
void USART2_IRQHandler(void)
{
  if(USART_GetITStatus(EVAL_COM1、 USART_IT_RXNE) != RESET)
  {
    /* シリアルポートからデータを受信、レジスタは１バイトを取り出す */
RxBuffer[RxCounter++] = USART_ReceiveData(EVAL_COM1);
/*受信データを返信*/
USART_SendData(EVAL_COM1、 RxBuffer[RxCounter-1]);

    if(RxCounter == NbrOfDataToRead)// 受信データ長は、受信バッファの長さに等しい場合
    {
      /*受信割り込みオフ*/
      USART_ITConfig(EVAL_COM1、 USART_IT_RXNE、 DISABLE);
    }
  }

  if(USART_GetITStatus(EVAL_COM1、 USART_IT_TXE) != RESET)
  {
    /* １バイトのデータを送信レジスタに書き込む */
    USART_SendData(EVAL_COM1、 TxBuffer[TxCounter++]);

    if(TxCounter == NbrOfDataToTransfer)// 送信データ長は、送信バッファの長さに等しい場合
    {
      /*送信割り込みオフ*/
```

```
        USART_ITConfig(EVAL_COM1、 USART_IT_TXE、 DISABLE);
      }
    }
  }
```



4.2.5. ¥CAN_LoopBack¥MDK-ARM

本ディレクトリ下のプログラムはCANのループバック通信モードを実現する。プログラムは先ずCANのボーレートを125 Kbpsに設定し、クエリ方式でデータを受送信する。そして受信データを送信データと比較し、正確な場合LED1点灯、エラーな場合LED3点灯。次にCANのボーレートを500 Kbpsに設定し、割り込み方式でデータ(CAN1_RX0_IRQHandler)、受送信データを比較し、正確な場合LED4点灯、エラーな場合LED2点灯。

4.2.6. ¥CAN_Networking¥MDK-ARM

本ディレクトリ下のプログラムはCANネット通信を実現する（2つまた以上の開発ボードが必要とする)。

➢テストする前に、2つの開発ボードのCAN_HとCAN_H、CAN_LとCAN_Lを接続し、プログラム実行後、開発ボードのUSERキーを押し、もう一つの開発ボードのLED 1は点灯、もう一回USERキー押すと、別のLEDが点灯。プログラムは割り込み方式でデータを受信する。割り込み関数CAN1_RX0_IRQHandlerにLEDの制御を確認できる。

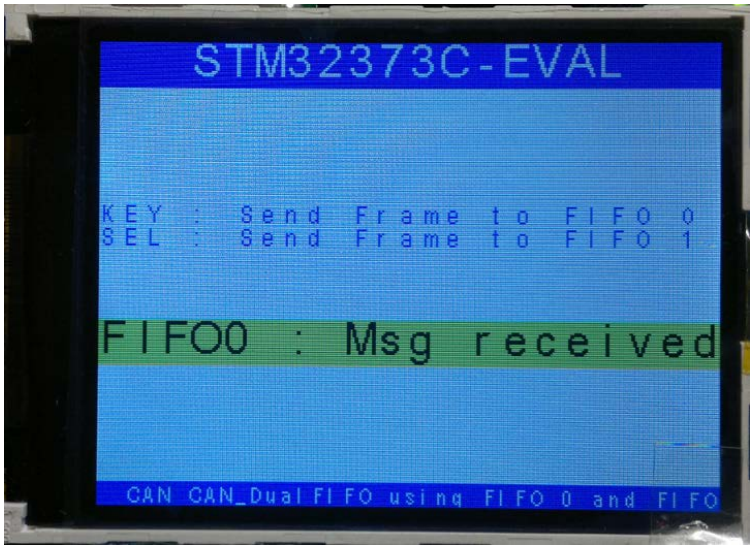
4.2.7. ¥CAN_DualFIFO¥MDK-ARM

本ディレクトリ下のプログラムは CAN のダブル FIFO 機能を実現する (2 つまた以上の開発ボードが必要とする)。

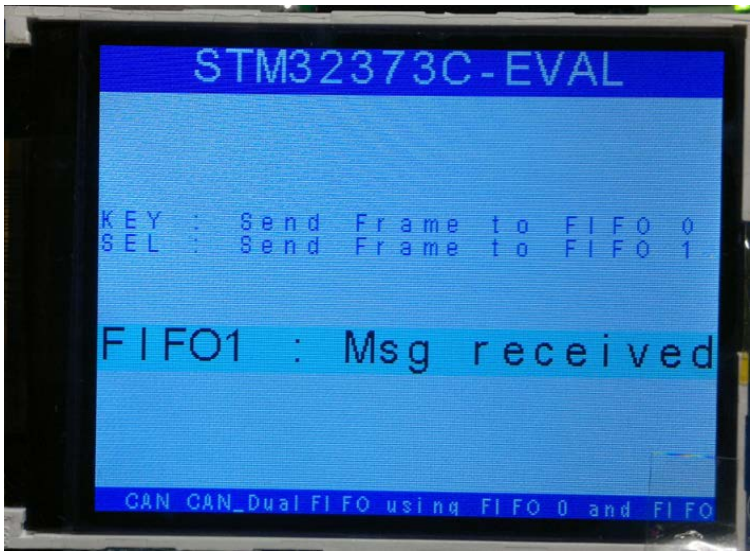
➢ テストする前に、2 つの開発ボードの CAN_H と CAN_H、CAN_L と CAN_L を接続し、シリアルポートで開発ボードの DB9 インタフェースを接続、ハイパーターミナルボーレート 115200、ハードウェアフロー制御なし。

➢ プログラム実行後、開発ボードのスクリーンに下記の内容を表示する。USER キーでデータを FIFO0 に送信する、ナビゲーションの中間キーで FIFO1 に送信する:

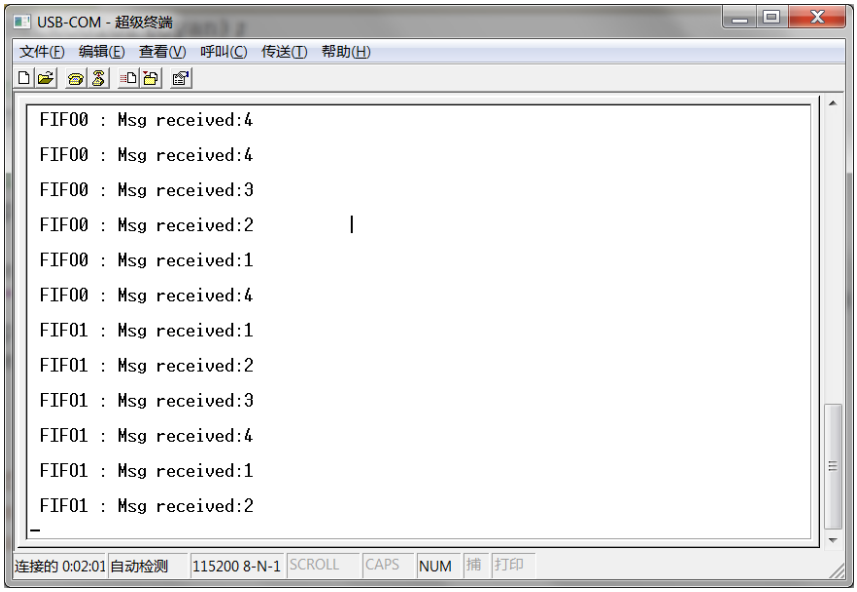
➢ USER キーを押し、開発ボードの LED4 点灯、もう一回押すと、LED4 滅 LED3 点灯。毎回キーを押すたびに、LED は変化する。



➢ 5 方向のナビゲーションボタンの中間キーを押し、スクリーンに FIFO1 がデータ受信と表示する。毎回キーを押すたびに、LED は変化する。



➢ ハイパーターミナルで受信データも確認できる。



➢ CAN_Config 関数で CAN 伝送モード、ボーレート、FIFO 初期化を設定する。CAN1_RX0_IRQHandler 及び CAN1_RX1_IRQHandler は CAN 割り込みサービス･ルーチン、FIFO データ取得と対応処理する。

4.2.8. ¥CAN_FIFOExtension¥MDK-ARM

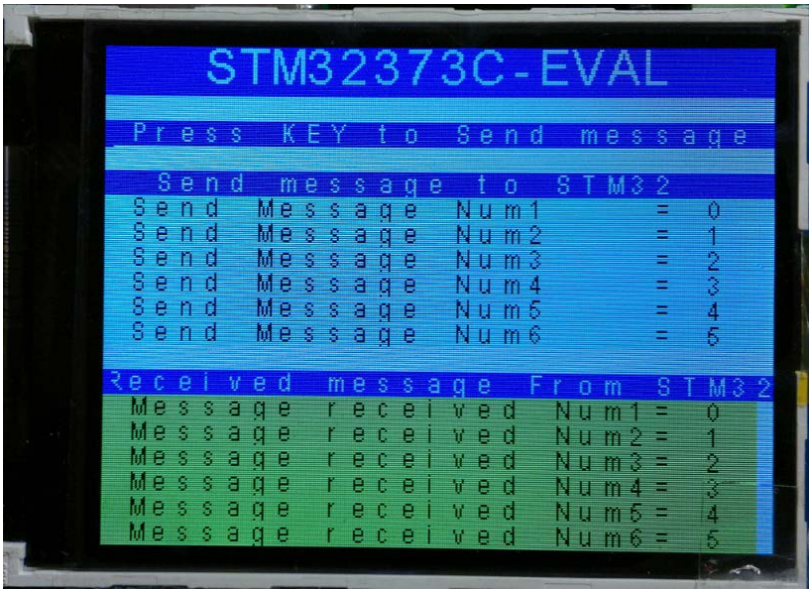
本ディレクトリ下のプログラムは 2 つの CAN の FIFO を 1 つの大き FIFO に合併し、6 つのメールボックスでデータ送信を実現する（2 つまた以上の開発ボードが必要とする）。

- テストする前に、2 つの開発ボードの CAN_H と CAN_H、CAN_L と CAN_L を接続し、シリアルポートで開発ボードの DB9 インタフェースを接続、ハイパーターミナルボーレート 115200、ハードウェアフロー制御なし。
- プログラム実行後、スクリーンに下記の内容が表示する。USER キーでデータ送信開始：

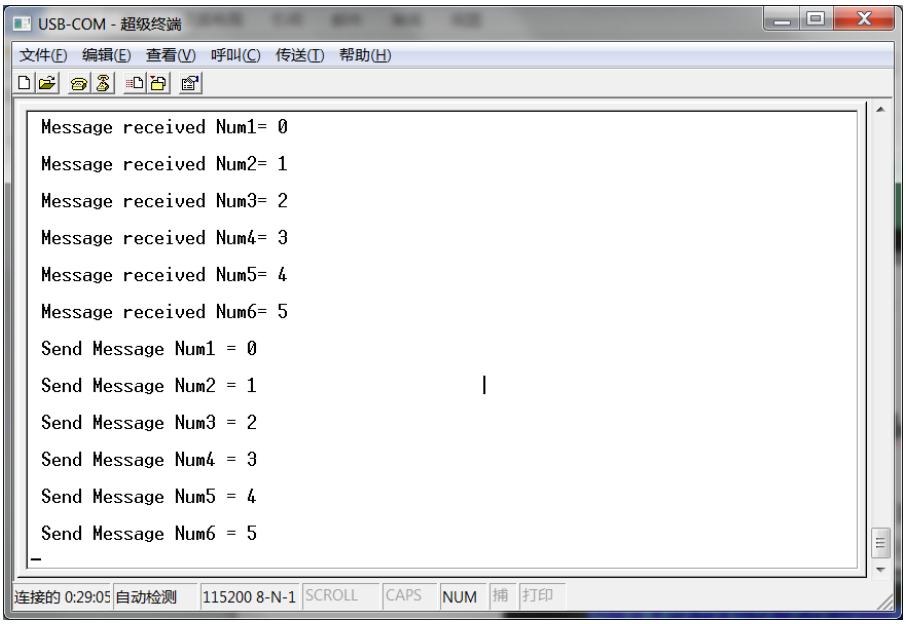
➢USER キーを押して、送信開発ボードのスクリーンに下記の内容を表示する：

➢受信側の開発ボードのスクリーンに下記の内容を表示する：

➢再度受信開発ボードの USER キーを押し、送信開発ボード側のスクリーンにも下記の内容を表示する（6 組データ受信）：

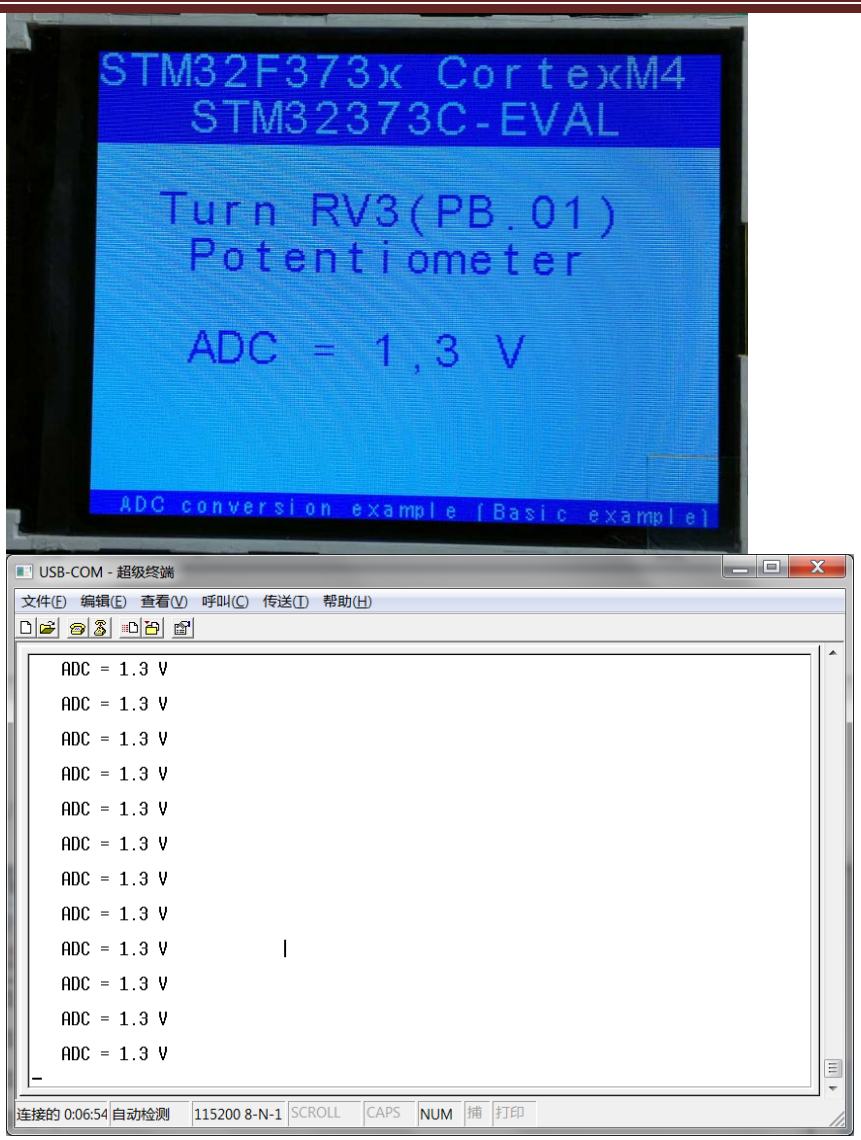


➢ ハイパーターミナルでデータ受送信プロセスを確認できる：



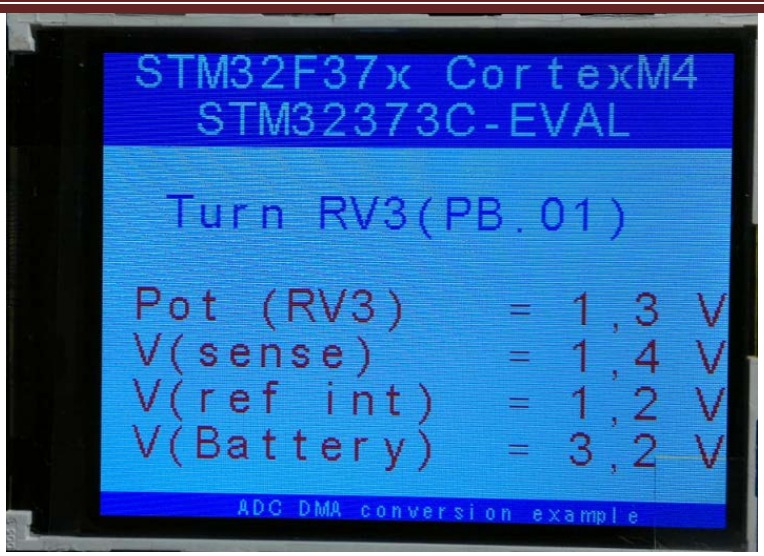
4.2.9. ¥ADC_Basic_Example¥MDK-ARM

本ディレクトリ下のプログラムは ADC チャンネル 9 を使用し、PB1 ピン接続の可変抵抗の電圧をスクリーンに表示する。ハイパーターミナルのボーレートは 115200、ハードウェアフロー制御なし、ハイパーターミナルにも電圧値をプリントアウトする。

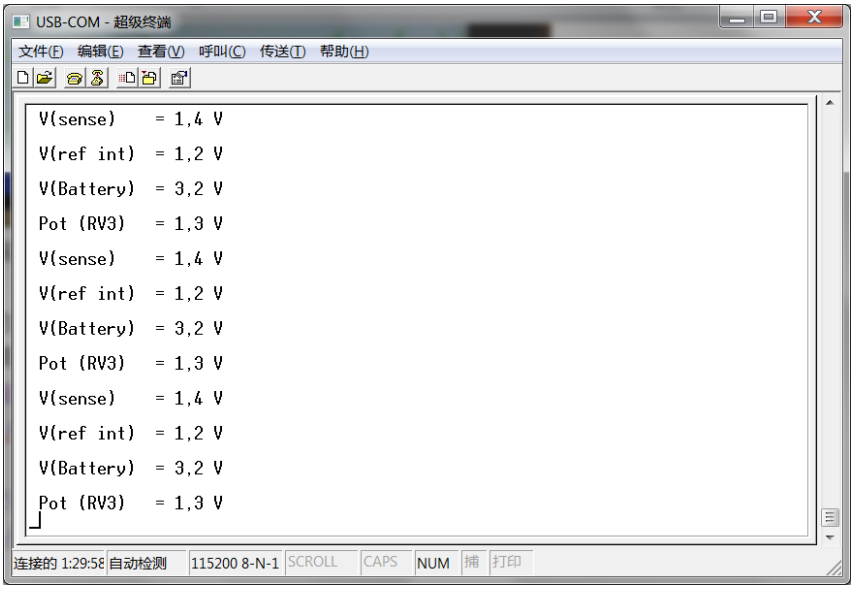


4.2.10. ¥ADC_DMA¥MDK-ARM

本ディレクトリ下のプログラムは外部アナログ入力チャンネル 9、内部温度センサ、内部基準電圧、外部 VBAT 電源ピンなど、上記のセンサのピンの電圧値を DMA メモリを介し、スクリーンにプリントアウトする。



➢ ハイパーターミナルボーレート 115200、ハードウェアフロー制御なし。:



➢ ADC1 の設定は main.c 中の ADC_Config 関数で実行する。

① 部アナログ入力チャネル 9 対応する PB1 ピンを初期化：

```
/* ADC サンプル･クロック周波数を設定、ADCCLK = PCLK2/4 */
RCC_ADCCLKConfig(RCC_PCLK2_Div4);
/* GPIOB の クロックをイネーブルする */
  RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIOB、 ENABLE);
  /* ADC チャンネル 9 をアナログ入力に設定する */
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_1;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AN;
  GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
  GPIO_Init(GPIOB、 &GPIO_InitStructure);
```

② DMA チャンネル 11 を設定する：

```
  /* DMA1 の クロックを有効する */
  RCC_AHBPeriphClockCmd(RCC_AHBPeriph_DMA1 、 ENABLE);
```

```
/* DMA1 チャンネル 11 の設定をリセット  */
DMA_DeInit(DMA1_Channel1);
/* ADC1 のデータレジスタアドレスを DMA チャンネル 11 に割り当てる */
DMA_InitStructure.DMA_PeripheralBaseAddr = (uint32_t)ADC1_DR_Address;
/* DMA チャンネル 11 のメモリベースアドレスを割り当て  */
DMA_InitStructure.DMA_MemoryBaseAddr = (uint32_t)RegularConvData_Tab;
/*周辺はソース/ターゲットを指定、ここではソース  */
DMA_InitStructure.DMA_DIR = DMA_DIR_PeripheralSRC;
/* 指定されたチャネルデータユニットのバッファサイズに割り当て */
DMA_InitStructure.DMA_BufferSize = 4;
/* ペリフェラルアドレスレジスインクリメンタルタ判断*/
DMA_InitStructure.DMA_PeripheralInc = DMA_PeripheralInc_Disable;
/* メモリアドレスレジスタインクリメンタル判断*/
DMA_InitStructure.DMA_MemoryInc = DMA_MemoryInc_Enable;
/*周辺データの幅設定、ハーフバイトの幅に設定する */
DMA_InitStructure.DMA_PeripheralDataSize = DMA_PeripheralDataSize_HalfWord;
/* メモリデータの幅設定、ハーフバイトの幅に設定する */
DMA_InitStructure.DMA_MemoryDataSize = DMA_MemoryDataSize_HalfWord;
/* DMA の動作モード、ループモードに設定する */
DMA_InitStructure.DMA_Mode = DMA_Mode_Circular;
/*優先レベルを設定する、 */
DMA_InitStructure.DMA_Priority = DMA_Priority_High;
/* メモリ － メモリ転送モードに設定する */
DMA_InitStructure.DMA_M2M = DMA_M2M_Disable;
/* DMA_InitStructure データ構成により DMA を初期化*/
DMA_Init(DMA1_Channel1、 &DMA_InitStructure);
/* DMA1 有効する */
DMA_Cmd(DMA1_Channel1、 ENABLE);
```

③ ADC 初期化、データ構造の中にコンフィギュレーションデータを設定する。

```
/* ADC1 の クロックを有効する */
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1、 ENABLE);

  /* 原来 ADC1 の設定をリセット */
  ADC_DeInit(ADC1);

  /* ADC_DMA を有効する*/
  ADC_DMACmd(ADC1、 ENABLE);

  /* ADC  データ構造を初期化*/
  ADC_StructInit(&ADC_InitStructure);
  /* ADC1 をマルチチャンネルスキャンモードに設定する */
  ADC_InitStructure.ADC_ScanConvMode = ENABLE;
```

```
/*ADC1 を連続モードに設定する */
ADC_InitStructure.ADC_ContinuousConvMode = ENABLE;
/* トリガモード設定、ここではトリガーなし*/
ADC_InitStructure.ADC_ExternalTrigConv = ADC_ExternalTrigConv_None;
/* データアライメント設定、ここでは右揃えを選択*/
ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;
/* 4 つ変換チャンネル */
ADC_InitStructure.ADC_NbrOfChannel = 4;
/* ADC1 初期化 */
ADC_Init(ADC1、 &ADC_InitStructure);
```

④ADC1 各チャンネルのサンプリング周波数設定する：

```
/* ADC1 外部アナログ入力チャンネル 9 のサンプリング周波数を 55.5 の ADC クロックサイクルに設定する
*/
ADC_RegularChannelConfig(ADC1、 ADC_Channel_9、 1、 ADC_SampleTime_55Cycles5);
/* ADC1 内部温度センサチャンネルのサンプリング周波数を 55.5 の ADC クロックサイクルに設定する */
ADC_RegularChannelConfig(ADC1、ADC_Channel_TempSensor、2、ADC_SampleTime_55Cycles5);
/*内部基準電圧チャンネルのサンプリング周波数を 55.5 の ADC クロックサイクルに設定する*/
ADC_RegularChannelConfig(ADC1、ADC_Channel_Vrefint、3、ADC_SampleTime_55Cycles5);
 /*内部基準電圧 VBAT 電源ピンチャンネルのサンプリング周波数を 55.5 の ADC クロックサイクルに設定
する */
ADC_RegularChannelConfig(ADC1、ADC_Channel_Vbat、4、ADC_SampleTime_239Cycles5);
```

4.2.11. ¥DAC_ADC¥MDK-ARM

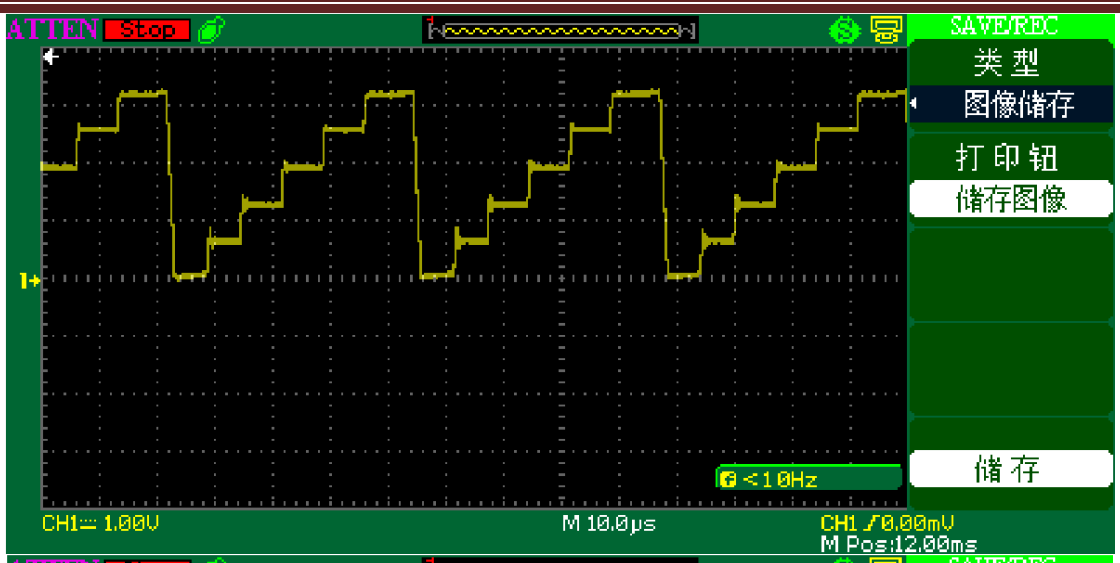
ディレクトリ下のプログラムは DAC と ADC を使用する、プログラムは割り込みモードで、ADC1 チャンネル 9(PB1)の対応ピンの電圧値を DAC 方式で DAC_OUT1 (PA4)ピンで表現できる。

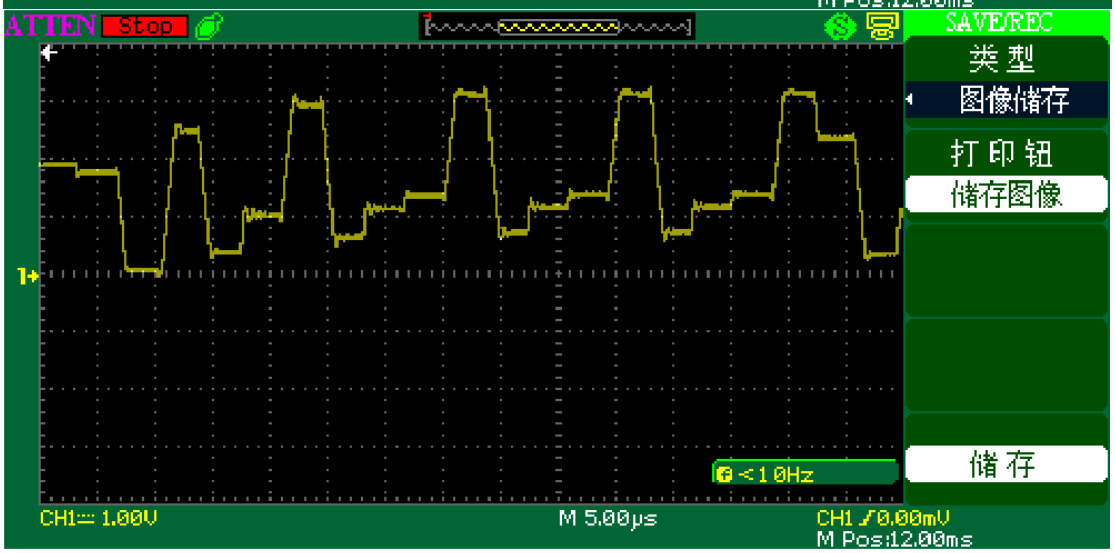
- プログラムは ADC_Config、DAC_Config２つの設定関数と１つの割り込み関数で組み合わせる。ADC 変換設定は変換完成後割り込みを生成し(EOC)、ADC 変換したデジタル量をアナログ量に変換し、出力する。可変抵抗 R39 の抵抗値を調節し、マルチメータで PA4 ピンの対応変化を測定する。(電圧は似ているが同じではない)

4.2.12. ¥DAC_SignalsGeneration¥MDK-ARM

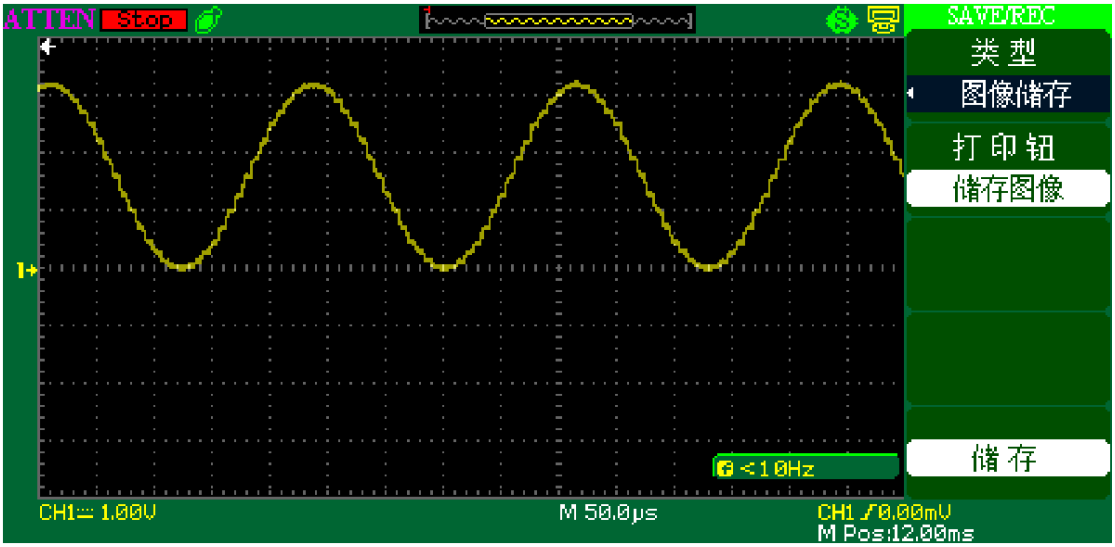
本ディレクトリ下のプログラムは DAC を使用し、DMA 方式で正弦波またはエスカレーターの波形を生成する。USER キーで波形を切り替える。オシロスコープで PA4、PA6 ピンの出力波形を確認できる。

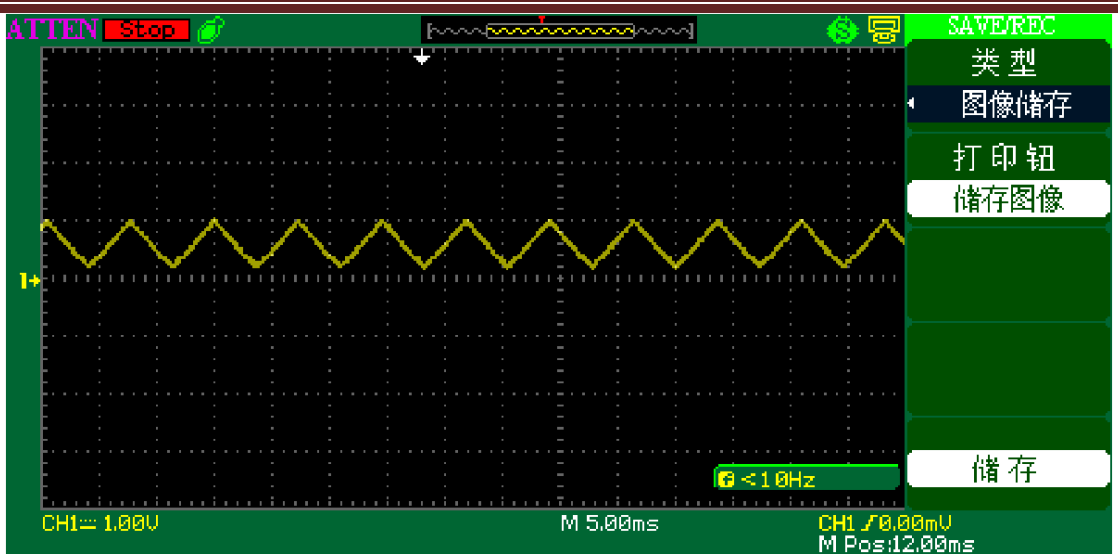
- PA4 ピンは USER キーでエスカレーター波形またはクラッター出力を切り替える。





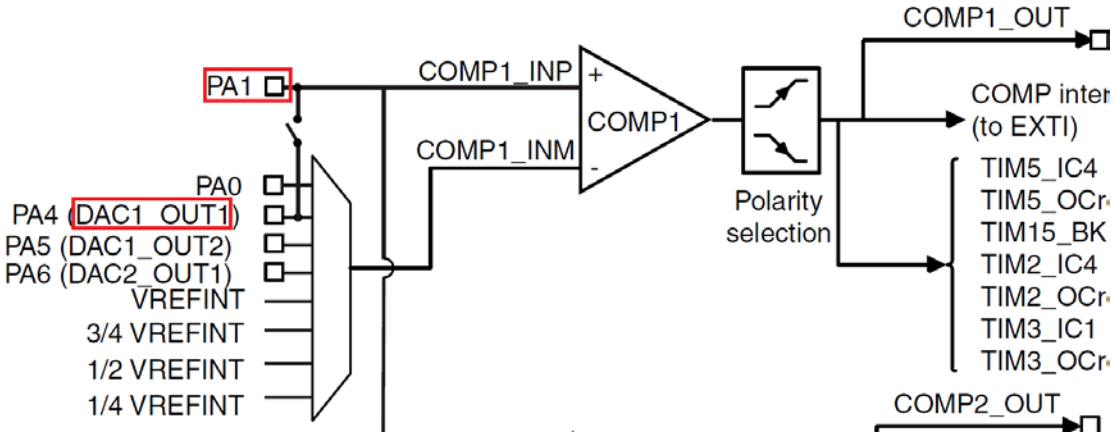
➢ PA6 ピンは USER キーで正弦波または三角波出力を切り替える：





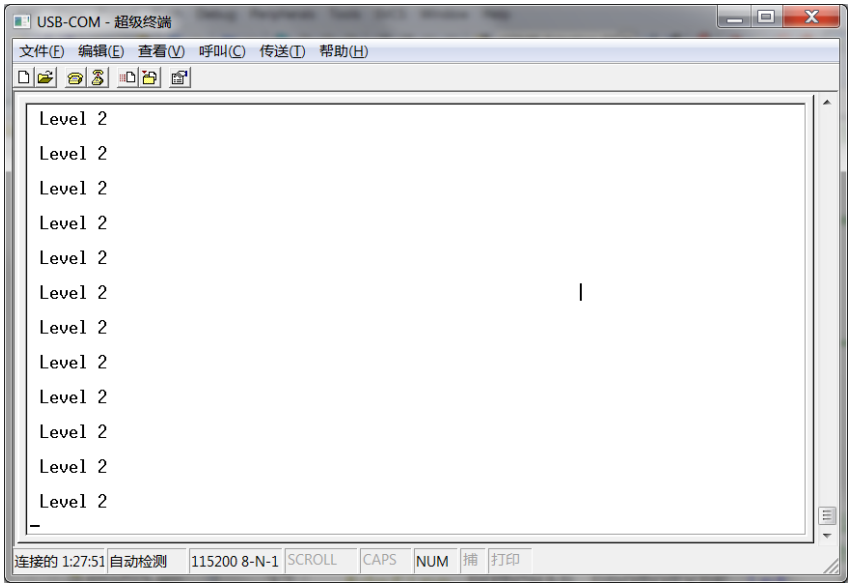
4.2.13. ¥COMP_LDR¥MDK-ARM

STM32F37x プロセッサには 2 つのコンパレータが内蔵し、そして DAC と組み合わせて使用することができる。プログラムは PA1 ピンとコンパレータ COMP1 の非反転入力ピンと接続し；DAC1 出力ピンと COMP1 の反転入力ピンと接続する。DAC1 の出力を 0.7 近くの電圧と設定する。PA1 ピンと接続感光抵抗上の光強度の相違により、PA1 ピンの電圧値は異なっている。プログラムは設定時間間隔で、コンパレータを介し PA1 ピンの電圧値を 0.7V と比較し、そしてハイ/ローの回数で光の強弱を判断する。



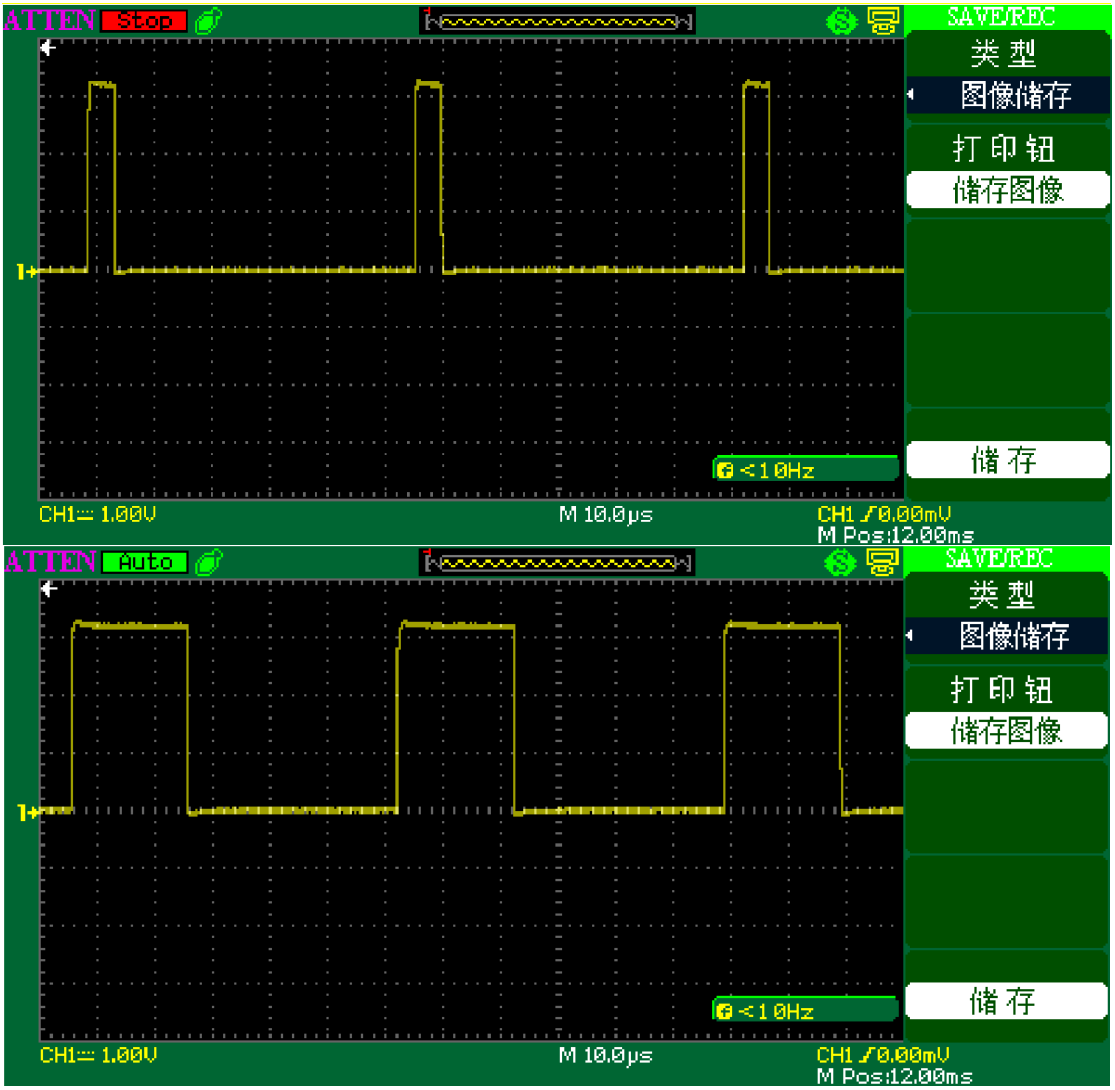
➢ フォトレジスタを遮蔽すると、スクリーンのレベルバーが対応変化する：

➢ ハイパーターミナルボーレート 115200、ハードウェアフロー制御なし。光強度の確認できる：



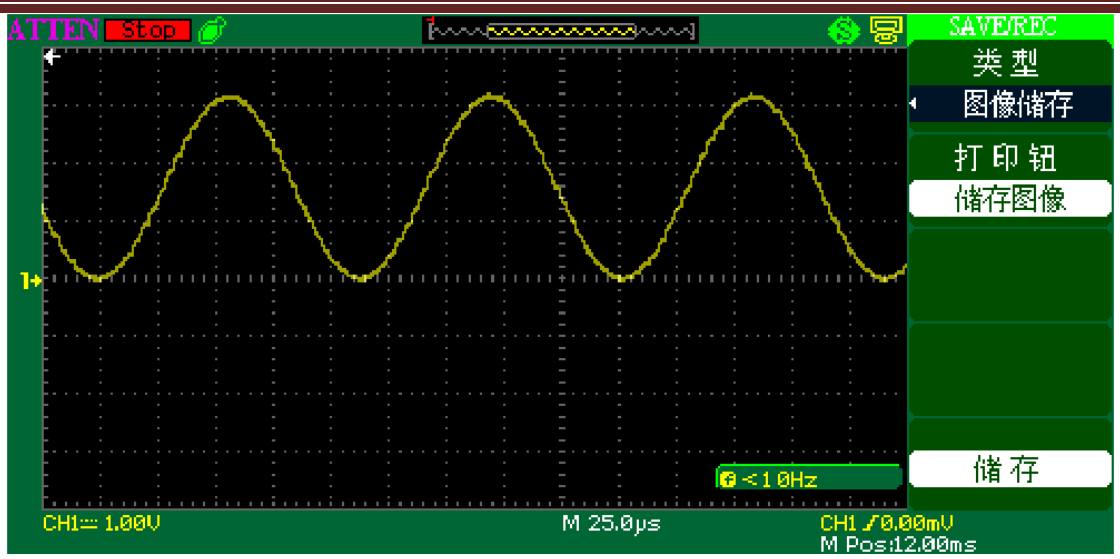
4.2.14. ¥DMA_ADCToTIM3Transfer¥MDK-ARM

本ディレクトリ下のプログラムはDMA伝送モードを介し、ADCから取得するデータをTIM3へ持続伝送する。TIM3のチャンネル3はPWM信号出力と設定する。ADCチャンネル9は可変抵抗R39と接続、ADC変換値はTIM3へ持続伝送し、PWMのデューティサイクルを変更する。可変抵抗の抵抗値はオシロスコープで確認できる。PB0ピン(R74測定)はハイレベルの占有時間が変化する。



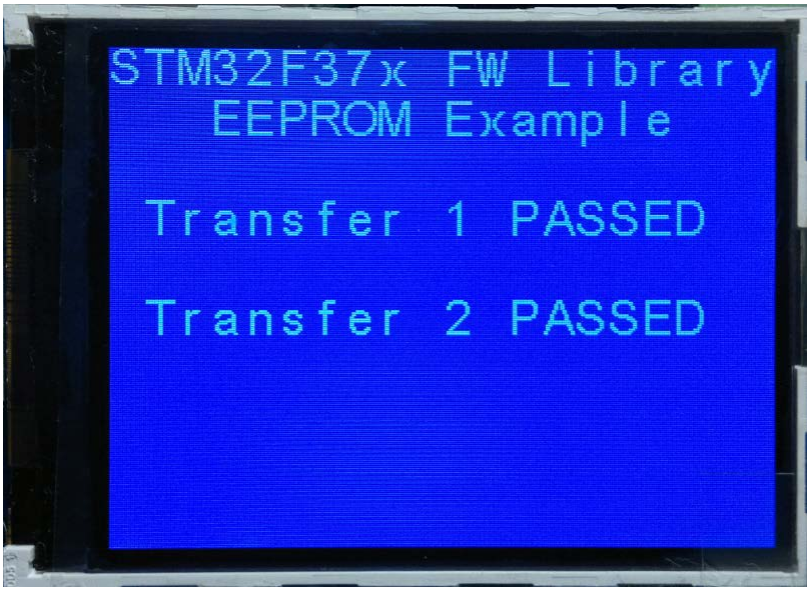
4.2.15. ¥DMA_RAMToDACTransfer¥MDK-ARM

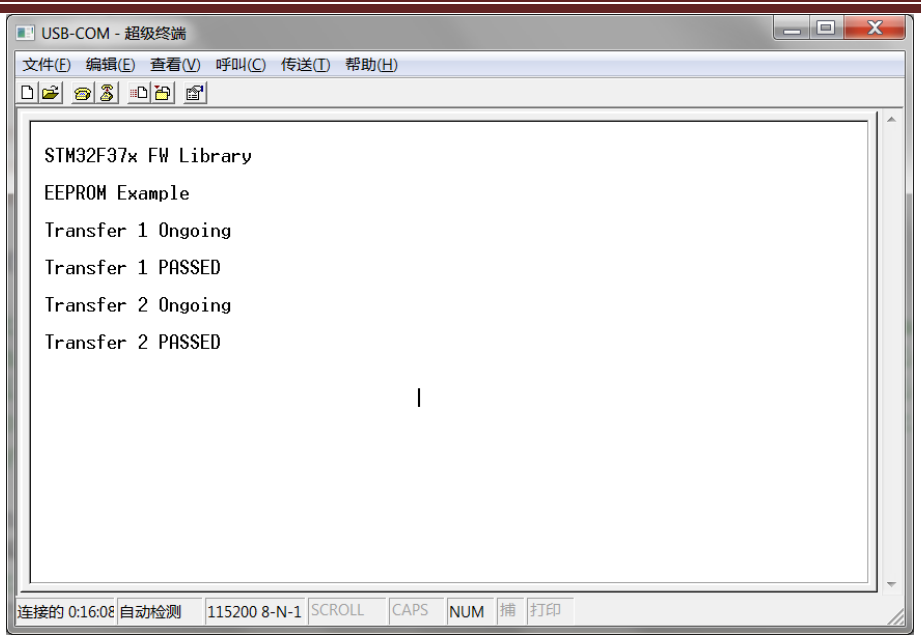
本ディレクトリ下のプログラムはRAM上の正弦波波形データ・DMAからDACチャンネル、オシロスコープで2つのDACチャンネルが正弦波出力を確認できる。



4.2.16. ¥I2C_EEPROM¥MDK-ARM

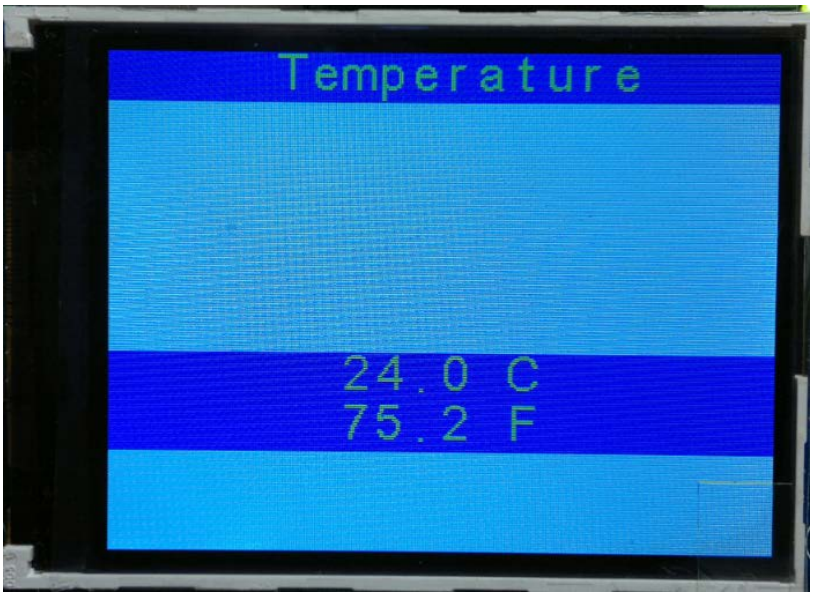
本ディレクトリ下は開発ボードの I2C インタフェースの EEPROM のテストプログラム。プログラムは EEPROM に固定文字列を書き込み、そしてそれを読み出し、比較する。書き込みデータと読み出しデータが一致する場合、スクリーンに Transfer PASSED を表示、でないと、Transfer FAILED を表示する。ハイパーターミナルにも同じ情報をプリントアウトする。ハイパーターミナルのボーレートは 115200、ハードウェアフロー制御なし。

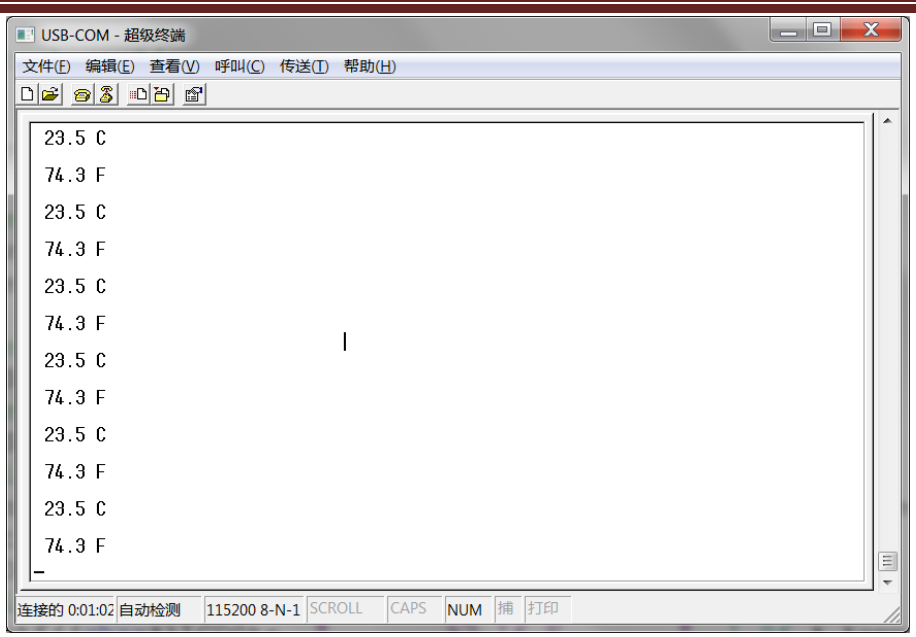




4.2.17. ¥I2C_TSENSOR¥MDK-ARM

本ディレクトリ下は開発ボードの I2C インタフェースの温度センサのテストプログラム。プログラムは収集温度をスクリーンに表示する。指先で温度センサ U8 を触ると、関連数値も対応変化する。ハイパーターミナルも同じ情報をプリントアウトする。ハイパーターミナルのボーレートは 115200、ハードウェアフロー制御なし。





4.2.18. ¥SPI_MSD¥MDK-ARM

SPI インタフェースの SD カード底層駆動テストプログラム。プログラムはデータをアドレス 0 からの 512 bytes にスペースに書き込み、また読み取り、比較する。データ正確 LED1 点灯；データエラーLED2 点灯。

4.2.19. ¥WWDG_Example¥MDK-ARM

本ディレクトリ下のプログラムはシステムウォッチドッグ通常状態で WWDG カウンタを更新する方法及びフィード時間を超える場合の WWDG リセットのシミュレーションをデモする。

プログラムは WWDG タイムアウトを 58.2 ミリ秒に設定し、43 ミリ秒毎フィードする。LED2 も 43 ミリ秒毎一回フラッシュする。USER キーが押された場合、プログラムは 1 回無効レジスタ書き込み動作を実行し、プログラムは無限ループのノーオペレーションスペースに切り替えたため、58.2 ミリ秒内でフィード動作が実行せず、WWDG がリセットされ、プログラム再実行し、RCC_GetFlagStatus 関数で WWDG リセットを判断し、開発ボードの LED1 点灯する。

4.2.20. ¥IWDG_Example¥MDK-ARM

本ディレクトリ下のプログラムはシステムウォッチドッグ通常状態で IWDG カウンタを更新する方法及びフィード時間を超える場合の IWDG リセットのシミュレーションをデモする。

プログラムは IWDG タイムアウトを 250 ミリ秒に設定され、240 ミリ秒毎フィードする。LED2 も 240 ミリ秒毎一回フラッシュする。USER キーが押された場合、プログラムは 1 回無効レジスタ書き込み動作を実行し、プログラムは無限ループのノーオペレーションスペースに切り替えたため、250 ミリ秒内でフィード動作が実行せず、IWDG がリセットされ、プログラム再実行し、RCC_GetFlagStatus 関数で IWDG リセットを判断し、

開発ボードの LED1 点灯。

4.2.21. ¥NVIC_IRQMask¥MDK-ARM

本ディレクトリ下のプログラムはネスト型ベクタ割り込みコントローラ（NVIC）の IRQ チャネルの設定、及び異なる IRQ のアクティブ/シールド動作の使い方をデモする。

- プログラムは TIM2 の割り込みで LED1 を 1 秒毎フラッシュさせる。TIM3 の割り込みで LED2 を 2 秒毎フラッシュさせる。TIM4 の割り込みで LED3 を 3 秒毎フラッシュさせる。
- USER キーを一回押し、LED4 点灯、TIM2、TIM3、TIM4 の割り込みオフで LED2、LED3、LED4 消灯。もう一回押すと、LED4 消灯、TIM2、TIM3、TIM4 の割り込みオンで LED2、LED3、LED4 フラッシュする。
- 5 方向のナビゲーションボタンの中間キーで BASEPRI レジスタを設定する。中間キーを一回押し、TIM3 と TIM4 割り込みをマスクする、TIM2 割り込みだけ有効なので LED1 だけフラッシュする。もう一回押すと、TIM3 と TIM4 割り込みオン、LED2、LED3 もフラッシュする。

4.2.22. ¥NVIC_WFIMode¥MDK-ARM

本ディレクトリ下のプログラムはシステムが WFI に入り、外部割り込みでシステムをウィックアップ動作をデモする。

プログラム実行後 LED3 フラッシュ、USER キーを押し、LED3 消灯、プロセッサが WFI モードに入る。電流計で開発ボードの電流が小さくなると確認できる。もう一回 USER キーを押し、プロセッサが WFI モード終了、LED3 フラッシュ。

4.2.23. ¥PWR_Standby¥MDK-ARM

本ディレクトリ下のプログラムはプロセッサは Standby モードに入る、また RTC アラームでプロセッサウィックアップ動作する。

プログラム実行後 LED3 フラッシュ、USER キーを押し、LED3 消灯、プロセッサが Standby モードに入る。電流計で開発ボードの電流が小さくなると確認できる。3 秒後 RTC アラームイベント発生、プロセッサがウィックアップされ、LED3 フラッシュ。

4.2.24. ¥PWR_Stop¥MDK-ARM

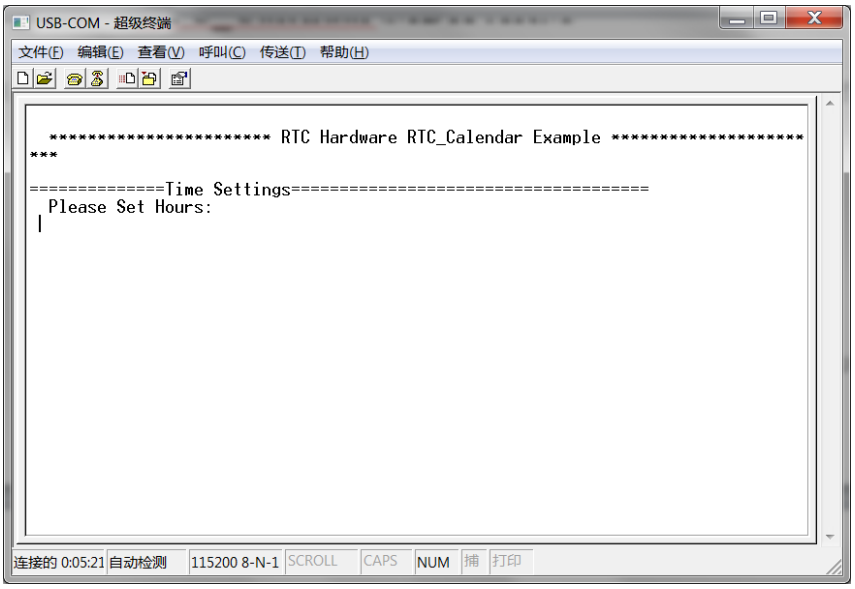
本ディレクトリ下のプログラムはプロセッサは Stop モードに入る、また外部割り込み及び RTC アラームでプロセッサウィックアップ動作する。

プログラムは RTC アラームにより、プロセッサを 5 秒毎 Stop モードに入り、5 秒後また RTC アラームにウィックアップされる。また、プロセッサは Stop モード中 USER キーでウィックアップできる。キーウィックアップ後 LED1 と LED2 点灯。RTC アラームウィックアップした場合、LED1 と LED4 点灯。

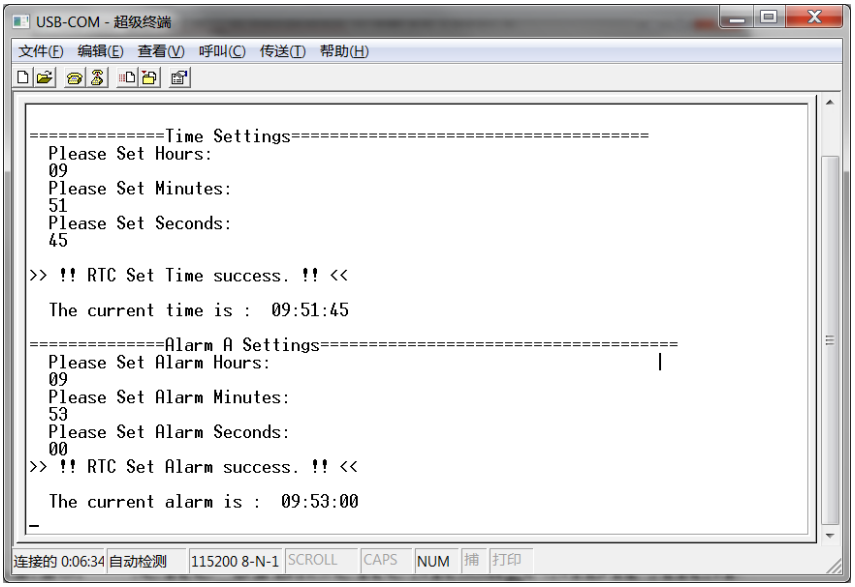
4.2.25. ¥RTC_Calendar¥MDK-ARM

本ディレクトリ下のプログラムは RTC 時間設定とアラームタイム設定する。プログラムはシリアルポートでタイムセットを実行する。

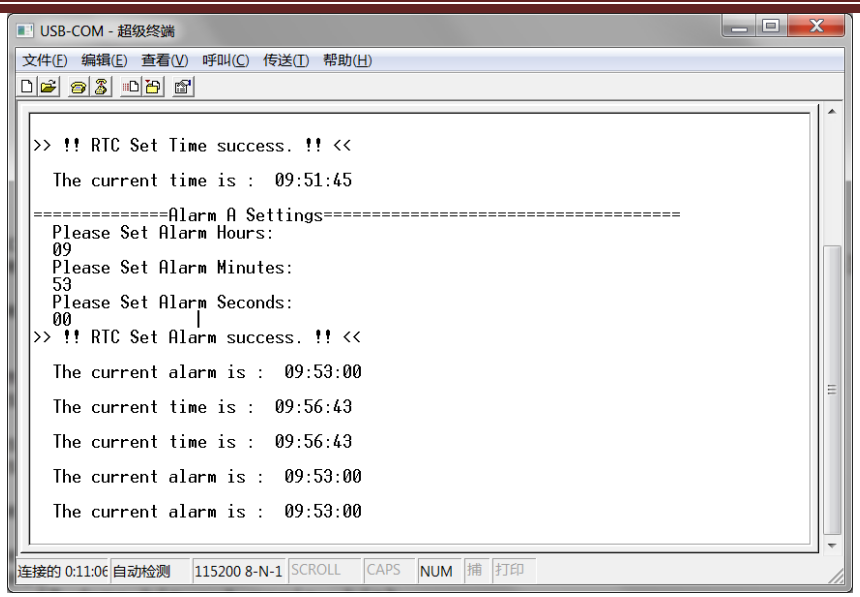
➢プログラム実行前に、ボタン電池を電池ホルダーにセットアップする。シリアルケーブルを開発ボードの DB9 インタフェースと接続、ハイパーターミナルボーレート 115200、ハードウェアフロー制御なし。プログラム実行後 LED2 点灯、ハイパーターミナルで現在の時間（hours）設定を提示する。



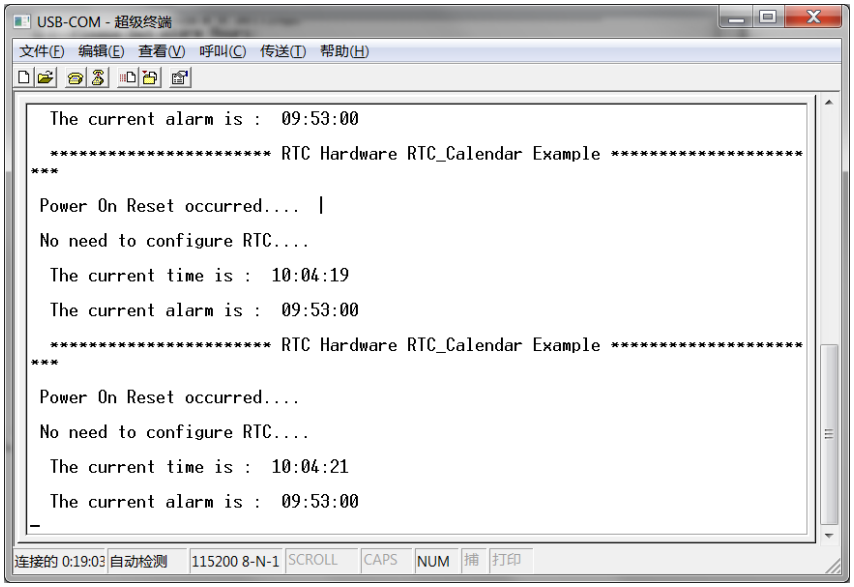
➢提示に従い時、分、秒と RTC アラームのタイムを設定する。下記図の通り、RTC アラームの時間はシステム時間の 3 分後に設定する。設定時間となると、RTC アラームが動作し、LED1 点灯。



➢開発ボードの USER キーでハイパーターミナルで現在時間をプリントアウト、5 方向のナビゲーションボタンの UP キーで、設定する RTC アラームの時間をプリントアウトする：



➢開発ボードのリセットボタンまた電源切断する場合、ハイパーターミナルは現在時間と RTC アラーム時間をプリントアウト、再設定する必要がない。



4.2.26. ¥RTC_LSI ¥MDK-ARM

本ディレクトリ下のプログラムは LSI クロックソースの自動補正、RTC クロックを正確にする。

➢LSI のクロックには誤差があるため、TIM14 でクロックを自動補正する。プログラム実行後、開発ボードの LED1 が 1 秒 1 回フラッシュする。UEDR キーを押すと、プログラムは LSI クロックを自動補正する。完了後 LED2 点灯。

4.2.27. ¥RTC_Tamper¥MDK-ARM

本ディレクトリ下のプログラムは RTC バックアップデータレジスタの読み取りおよび書き込み動作する。

プログラム実行後、バックアップデータレジスタに 0xA53C を書き込み、また読み出し、比較する。データが同じな場合、LED1 点灯、違う場合 LED4 点灯。

4.2.28. ¥RTC_Timer¥MDK-ARM

本ディレクトリ下のプログラムは時間砂時計と LCD 描画機能をデモする。

➢ プログラム実行後、下記の内容が表示する：

➢ スクリーンの提示に従い、5 方向のナビゲーションボタンの左キーでリセット、右キーで実行、上/下キーでタイマー設定。実行プロセスは下記の通り：

4.2.29. ¥TIM_TimeBase¥MDK-ARM

本ディレクトリ下のプログラムは TIM3 CC1、TIM3 CC2、TIM3 CC3、TIM3 CC4 レジスタの値を変更し、CC1、CC2、CC3、CC4 のアップデート周波数を 73.24 Hz、109.8 Hz、219.7 Hz、439.4 Hz に設定する。割り込みサービスルーチン TIM3_IRQHandler で、開発ボードの 4 つの LED が 73.24 Hz、109.8 Hz、219.7、439.4 Hz 間

隔の点滅の切り替え、切り替え周波数はオシロスコープで確認できる。

4.2.30. ¥CRC_32BitsCRCMessage¥MDK-ARM

STM32F373 プロセッサは CRC 計算ユニットを内蔵し、8 ビットまたは 32 ビットモードを構成できる。プログラム実行後、CRCBuffer 配列内のデータを CRC 計算を行い、完了後予想された結果と比較し、同じな場合 LED1 点灯、違い場合 LED3 点灯。

以上。