液晶オシロスコープ

・どんなの作るの？
・ハードウェア
・ファームウェア

Goji(goji2100@gmail.com)

どんなの作るの？

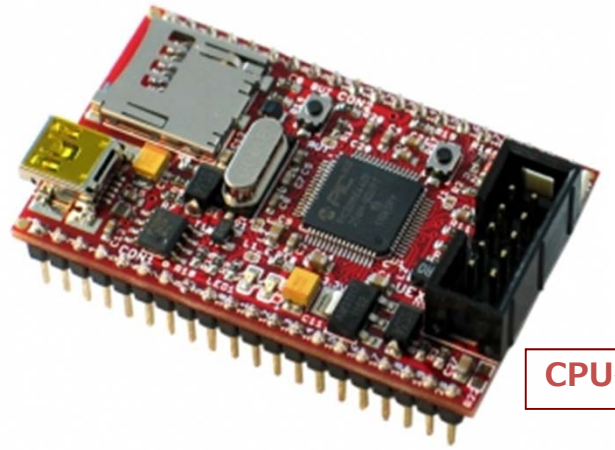
- **電子工作に手軽に使えるオシロがあれば便利**
  - 高速シリアル(115kbps)が確認できればいい
  - 液晶表示で、操作はタッチパネルがいい
  - PCに取込めるのがいい
  - 小型がいい
- **パイプラインADCが速そうだが、使い方が・・**
  - 後閑先生のアナログ･リッチPICの記事が出た！

ハードウェア構成(1)

・**CPUボード**

PIC24FJ128GC006(Flash:128KB, RAM:8KB)

- Clock:32MHz
- 12-Bit, High-Speed, Pipelined Analog-to-Digital (ADC)
- Two 10-Bit Digital-to-Analog Converters (DAC)
- Two Rail-to-Rail, General Purpose Operational Amplifiers



OLIMEX PIC32-PINGUINO-MICRO
PIC32MX440F256H
PINGUINO DEVELOPMENT BOARD
19.95EUR

CPUをPIC24FJ128GC006に換装

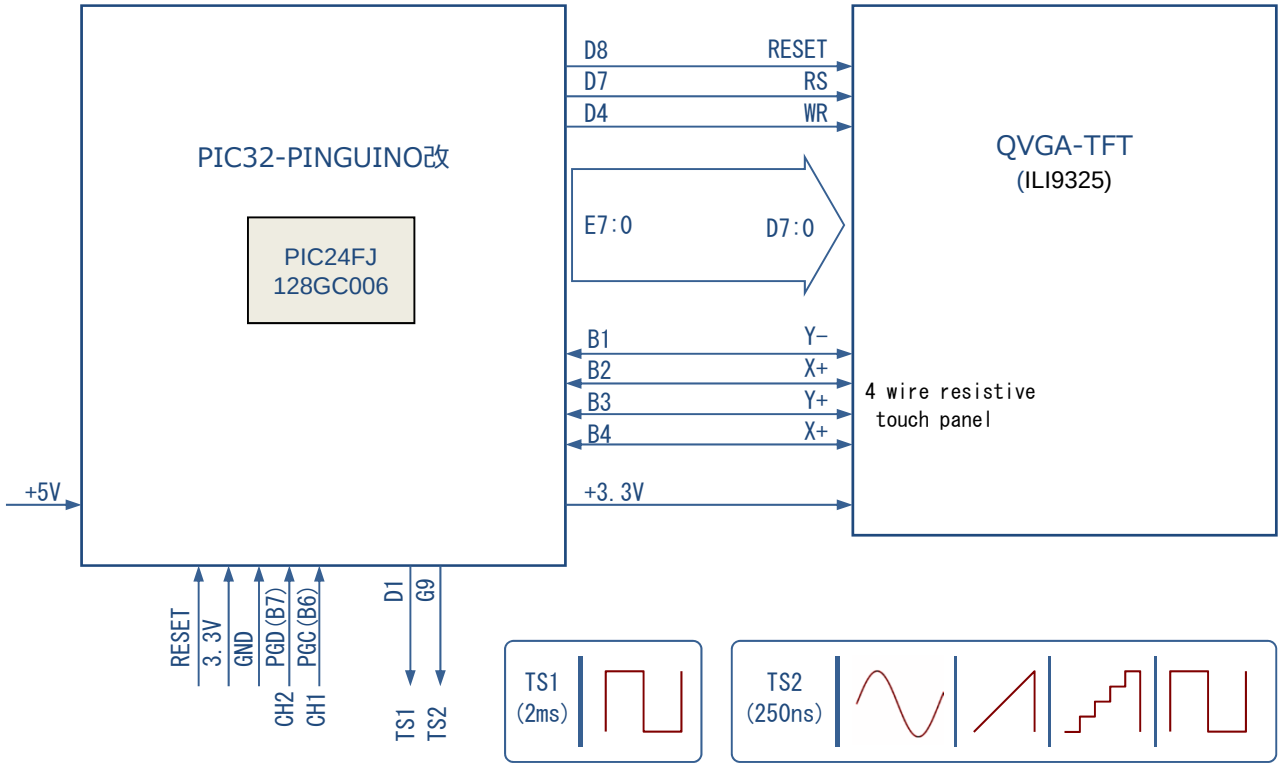
ハードウェア構成(2)

- **液晶**
  aitendo 液晶with基板（2.4インチforさくら）

GR-SAKURA用液晶
2.4インチTFT液晶モジュール(解像度:240x320)
4線抵抗膜型タッチスクリーン付き
8/16ビット表示モード
SDソケット取付可能
1,980円

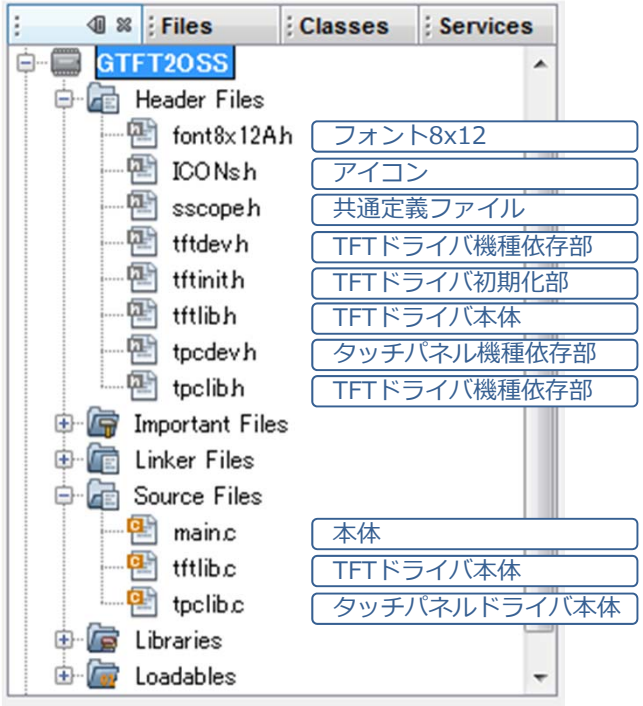
ハードウェア構成(3)

・**接続**



ファームウェア構成(1)

・モジュール構成



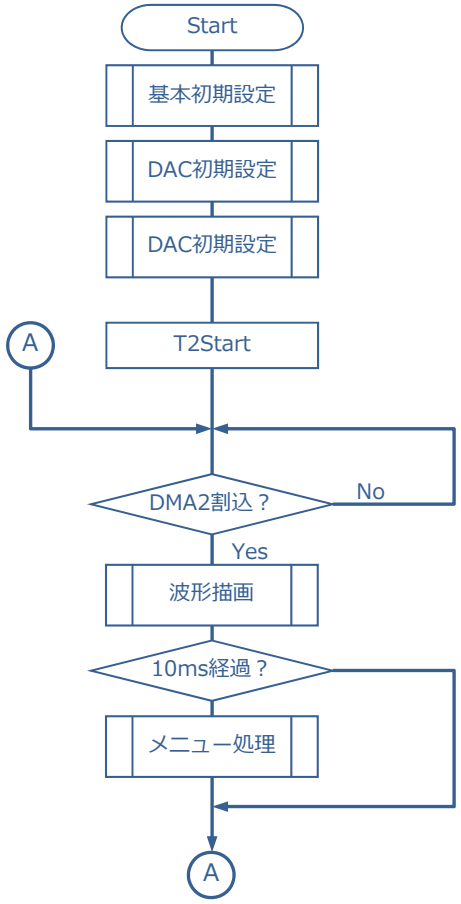

```
// -- Platform Select --------
//#define __PIC32MX__
//#define __PIC24FJ128GC006__
//#define GR_SAKURA
//#define ARCH_PRO
```

```
// -- TFT LCD Select ---------
//#define HX8352A
  #define ILI9325 // aitendo 2.4" for SAKURA
//#define SSD1289 // aitendo 3.2"
//#define ST7783  // aitendo 2.6" for UNO
//#define ST7783R // Seeedstudio 2.8"
```

```
// -- Touch I/O Select -------
  #define PIC_ADC
//#define ADS7843
//#define XPT2046
```

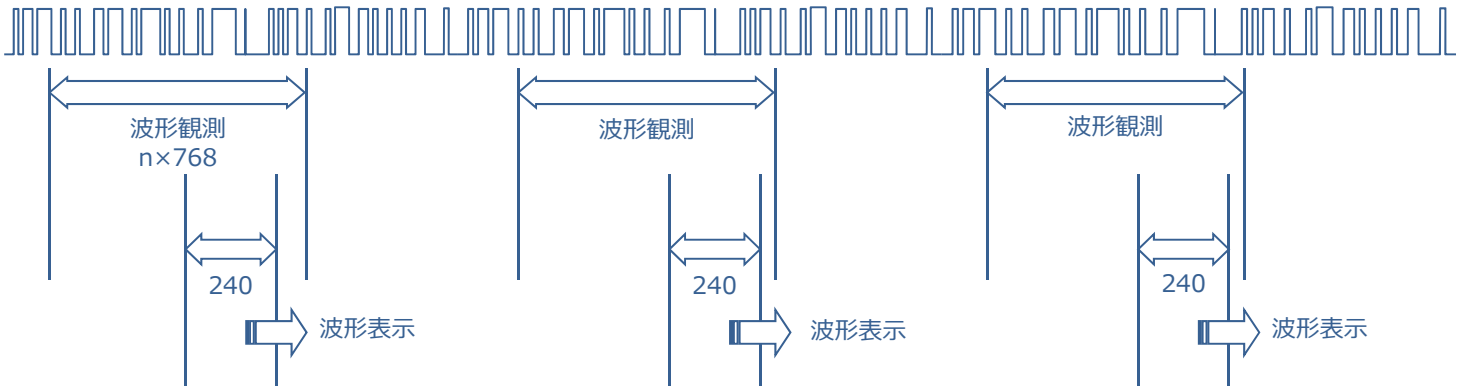
ファームウェア構成(2)

- **処理概要**



ファームウェア構成(3)

・波形観測、表示処理時間



トリガー条件を満足する波形の変化点を検索する：

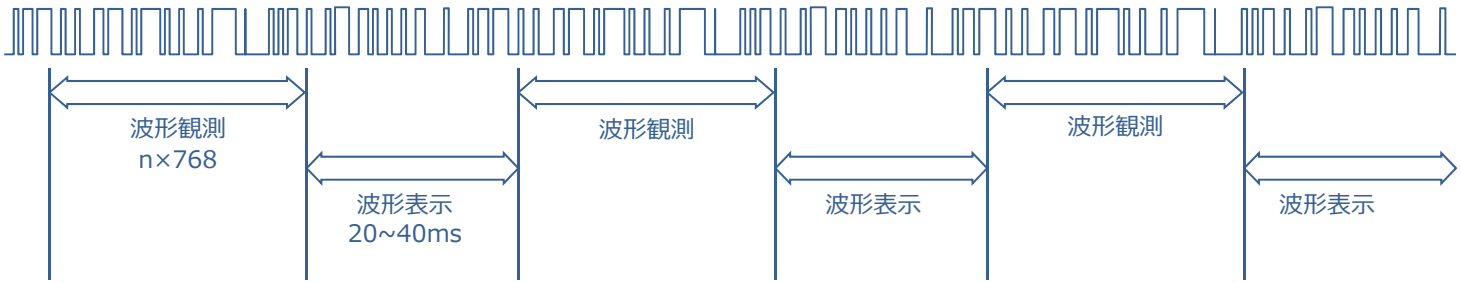
・ADCからの取込みバッファーの(32[ゴミ]＋表示領域の半分[X方向]＋水平シフト)以降から検索

・トリガー条件を満足した前後の表示領域分[X方向]を描画する

この方法では、トリガーポイントが後方にある場合、トリーガ条件を満足しても描画されない

ファームウェア構成(4)

・トリーガ処理



波形観測時間：

n×512 ⇒ n(HDIV/20)×768

ex)20u：20/20x768=768us、20ms：20/20x768=768ms

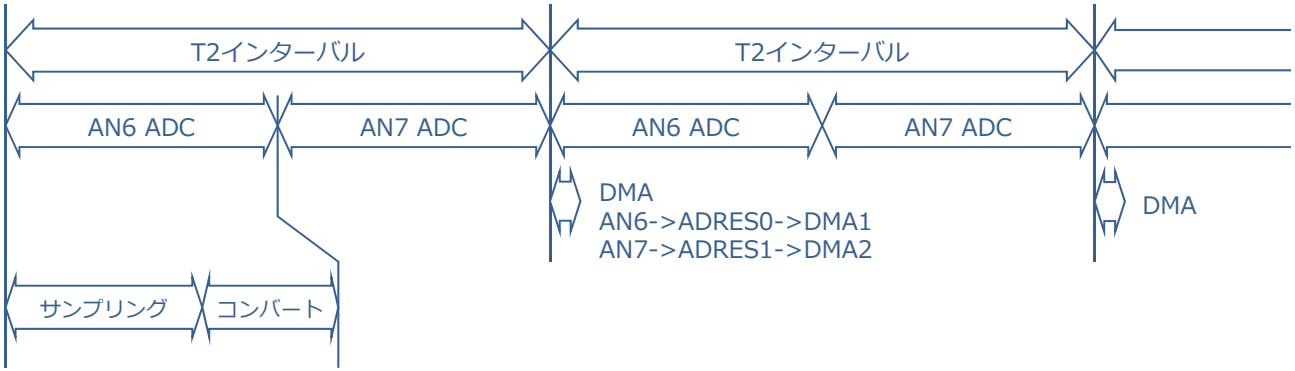
波形表示時間：

(描画開始位置設定＋(点描画×(8×20)))×(12×20)=(4+4+(2×160))×240=78,720バイト書込み

≒20～60ms(PIC24FJ)　≦20ms(PIC32MX)

ファームウェア構成(5)

・DMAによるADC変換結果の取込み



DIV	T2	PR2	ADCS	SAMC	Total	us/pixel
100us	5000ns	80Tsys	8Tsys	2Tad	16Tsys	5
50us	2500ns	40Tsys	4Tsys	2Tad	8Tsys	2.5
20us	1000ns	16Tsys	2Tsys	0.5Tad	4Tsys	1
10us	500ns	8Tsys	2Tsys	0.5Tad	4Tsys	0.5
5us	250ns	4Tsys	2Tsys	0.5Tad	4Tsys	0.25

ファームウェア構成(6)

```
DMACONbits.PRSSEL = 1;      // Fixed Priority
DMAL = 0x0800;              // Lower Limit
DMAH = 0x3000;              // Upper Limit

// DMA CH1 for Scope CH1
DMACH1 = 0;                 // Stop Channel
DMACH1bits.RELOAD = 1;      // Reload DMASRC, DMADST, DMACNT
DMACH1bits.TRMODE = 1;      // Repeat One shot
DMACH1bits.SAMODE = 0;      // Source Addrs No Increment
DMACH1bits.DAMODE = 1;      // Dist Addrs Increment
DMACH1bits.SIZE = 0;        // Word Mode(16bit)
DMASRC1 = (unsigned int)&ADRES0;// From ADC Buf0 select
DMADST1 = (unsigned int)&ADCvals[0][0]; // To Buffer select
DMACNT1 = (ADC_VALS + 64);  // DMA Counter
DMAINT1 = 0;                // All Clear
DMAINT1bits.CHSEL = 0x2F;   // Select Pipeline ADC
DMACH1bits.CHEN = 1;        // Channel Enable
IFS0bits.DMA1IF = 0;        // Flag Reset

// DMA CH2 for Scope CH2
DMACH2 = 0;                 // Stop Channel
DMACH2bits.RELOAD = 1;      // Reload DMASRC, DMADST, DMACNT
DMACH2bits.TRMODE = 1;      // Repeat One shot
DMACH2bits.SAMODE = 0;      // Source Addrs No Increment
DMACH2bits.DAMODE = 1;      // Dist Addrs Increment
DMACH2bits.SIZE = 0;        // Word Mode(16bit)
DMASRC2 = (unsigned int)&ADRES1;// From ADC Buf0 select
DMADST2 = (unsigned int)&ADCvals[1][0]; // To Buffer select
DMACNT2 = (ADC_VALS + 64);  // DMA Counter
DMAINT2 = 0;                // All Clear
DMAINT2bits.CHSEL = 0x2F;   // Select Pipeline ADC
DMACH2bits.CHEN = 1;        // Channel Enable
IFS1bits.DMA2IF = 0;        // Flag Reset

// ADC Setting
ADCON1 = 0;                 // All Clear
ADCON1bits.FORM = 0;        // unsigned Integer
ADCON1bits.PWRLVL = 1;      // Full Power
```

```
ADCON2 = 0;
ADCON2bits.ADPWR = 3;       // Always powered
ADCON2bits.PVCFG = 0;       // Vref+=AVDD
ADCON2bits.NVCFG = 0;       // Vref-=VSS
ADCON2bits.BUFORG = 1;      // Use Indexed Buffer
ADCON2bits.BUFINT = 0;      // No buffer Interrupt

ADCON3 = 0;

// Sample List #0
ADL0CONH = 0;
ADL0CONHbits.ASEN = 1;      // Enable auto-scan

ADL0CONHbits.SLINT = 1;     // Interrupt after every convert
ADL0CONHbits.WM = 0;        // All conversion results saved
ADL0CONHbits.SAMC = 0;      // Acquisition Time) = 0.5 TAD

ADL0CONL = 0;
ADL0CONLbits.SLEN = 1;      // Enable Trigger
ADL0CONLbits.SLTSRC = 5;    // Timer2 Trigger
ADL0CONLbits.SLSIZE = 1;    // 2 channel in the List

// Select Channel
ADL0PTR = 0;                // Start from first
ADTBL0  = 6;                // PGEC2(RB6)
ADTBL1  = 7;                // PGED2(RB7)

// Execute Calibration ADC
ADCON1bits.ADON = 1;        // ADC Enable
while(ADSTATHbits.ADREADY == 0);
ADCON1bits.ADCAL = 1;       // Calibration Start
while(ADSTATHbits.ADREADY == 0);
ADL0CONLbits.SAMP = 1;      // Start Sample
ADL0CONLbits.SAMP = 0;      // Start Conversion

// Peripheral Initialization]
CAPT_WAVE_OFF();            // timer2 stop
setupPR2(sv_horz_unit);
```

これから

・正しい波形観測
・アナログ部の組み込み（ゲンイ調整、レベルシフト、校正）
・PCへの観測波形の送出（波形データだけ）
・各種設定値を保存（プログラム領域またはEEPROM）

・PIC32MZ・・・
12-bit ADC module:
-28 Msps with six Sample and Hold (S&H) circuits
Graphics interfaces: EBI or PMP