STM32 KEIL 강좌[2]: 편리한 기능 찾아보기

KEIL의 매력에 빠져봅시다.

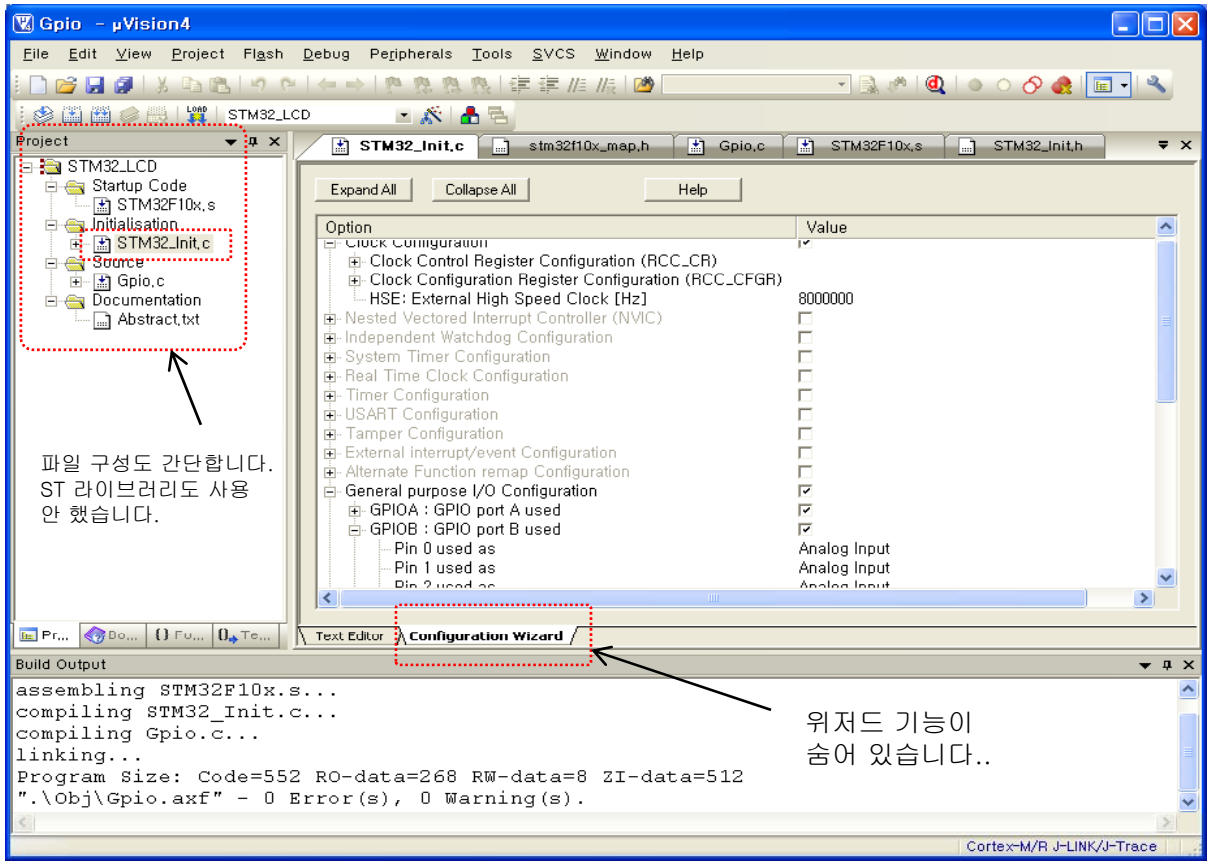
RealSYS

작성일자: 2010.6.16

현재 국내에서 ARM 개발에 KEIL보다는 IAR를 더 많이 사용하는 것 같아요. 하지만 이 강좌를 보시면 IAR보다 KEIL를 더 좋아하게 되지 않을까 생각됩니다. 믿거나 말거나 ~ㅎ ㅎ.
그러면 지금부터 KEIL의 편리한 점을 소개해드리도록 합니다.

[1]. 소스 생성 위저드 기능이 있다 ? (초기화 함수에 대해서)

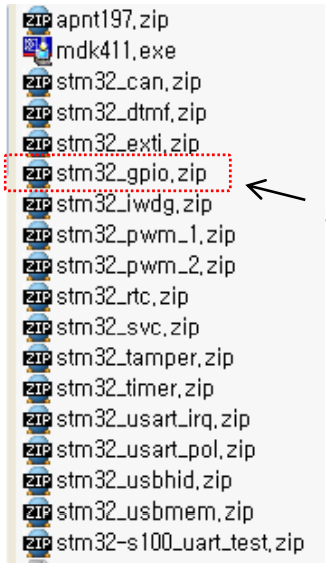
사용자가 원하는 설정에 따라 소스를 생성해준다면, 초기 개발 과정에 개발자에게 매우 많은 도움이 됩니다.
입출력 포트의 설정, 타이머의 설정, 클럭 설정 등을 할 때, 데이터 시트를 보고 직접 하려면 머리가 아프죠?
그래서 이전의 예제를 보고하는데, 예제가 복잡하면 분석하는데 시간이 많이 걸립니다.



IAR에는 이러한 소스 생성 기능이 없습니다.
하지만 KEIL을 잘 살펴보면 살짝 숨어 있어요.
그리고 그걸 잘 활용하면 이제 초보 엔지니어는
물 만난 하마가 됩니다.

월~ 이렇게 호들갑을 떠는가???

재미있게 진행해 볼려구 기를 쓰고 있어요.



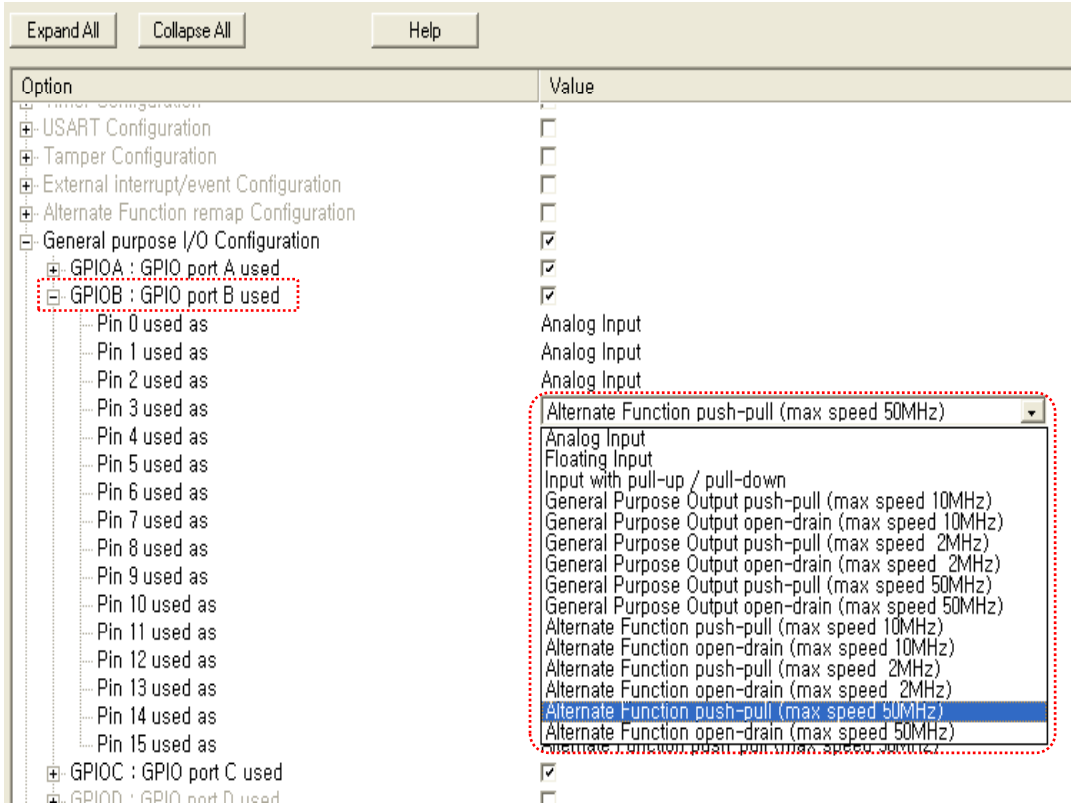
KEIL에서 가져온
예제 중 하나
Stm32_gpio를
보드에 맞게 조금
수정하여
진행해 보겠습니다.

리얼시스 www.realsys.co.kr Tel:031-342-3000, Fax:031-343-0003 경기도 안양시 동안구 호계동 1027 안양 IT밸리 504호

소스 생성 위저드 기능 - 아시는 분만 아는

KEIL 웹에서 ARM 관련 예제 및 기타 유용한 유틸리티를 가져옵니다.
이 예제를 사용하여도 되고, 아니면 이전에 IAR로 작업하던, ST사의 예제를 사용하여도 됩니다.

<1> STM32 주요 내부 장치에 대해서 설정이 가능하며, GPIOB 설정에 대해서 살펴봅니다. 포트 각 비트에 대해서 섬세한 입출력 선택이 가능합니다.



따라서 코드 크기도 작고, 처리 시간도 빠르겠죠?
이전 강좌에서 ST 라이브러리를 사용하는 IAR과 비교하면 실행 코드가 엄청 작아지겠지요?
그리고 이 소스를 분석하는데 그렇게 머리가 아프지 않겠지요?

<2> 설정 값에 따라 define이 생성되고

```
#define __GPIO_SETUP            1
#define __GPIO_USED             0x07
#define __GPIOA_CRL             0xBBBBBB33
#define __GPIOA_CRH             0xBBBBBBB3
#define __GPIOB_CRL             0x333FB000
#define __GPIOB_CRH             0xBBBBFB33
#define __GPIOC_CRL             0x33088888
#define __GPIOC_CRH             0x3333BB33
#define __GPIOD_CRL             0x00000000
#define __GPIOD_CRH             0x00000000
#define __GPIOE_CRL             0x00000000
#define __GPIOE_CRH             0x00000000
#define __GPIOF_CRL             0x00000000
#define __GPIOF_CRH             0x00000000
#define __GPIOG_CRL             0x00000000
#define __GPIOG_CRH             0x00000000
```

<3> 이 값이 바로 레지스터로 설정됩니다.

```
__inline static void stm32_GpioSetup (void) {

  if (__GPIO_USED & 0x01) {
    RCC->APB2ENR |= RCC_APB2ENR_IOPAEN;
    GPIOA->CRL = __GPIOA_CRL;
    GPIOA->CRH = __GPIOA_CRH;
  }

  if (__GPIO_USED & 0x02) {
    RCC->APB2ENR |= RCC_APB2ENR_IOPBEN;
    GPIOB->CRL = __GPIOB_CRL;
    GPIOB->CRH = __GPIOB_CRH;
  }

  if (__GPIO_USED & 0x04) {
    RCC->APB2ENR |= RCC_APB2ENR_IOPCEN;
    GPIOC->CRL = __GPIOC_CRL;
    GPIOC->CRH = __GPIOC_CRH;
  }
```

ST 라이브러리: GPIO 초기화 함수 – 좀 복잡해요

RealSYS

아래의 함수는 ST에서 제공하는 GPIO 초기화 함수입니다. 좀 복잡하죠? 일반성을 고려하다 보니 파라미터 검사. 여러 포트에 맞게 고려해서 작성되었는데, IAR에는 위저드 기능은 없고, 데이터 시트 읽어서 작성하기도 머리 아프고, 그래서 그냥 이전에는 이 함수를 사용하죠.
이에 비해서 앞장의 초기화 함수는 어떤가요?

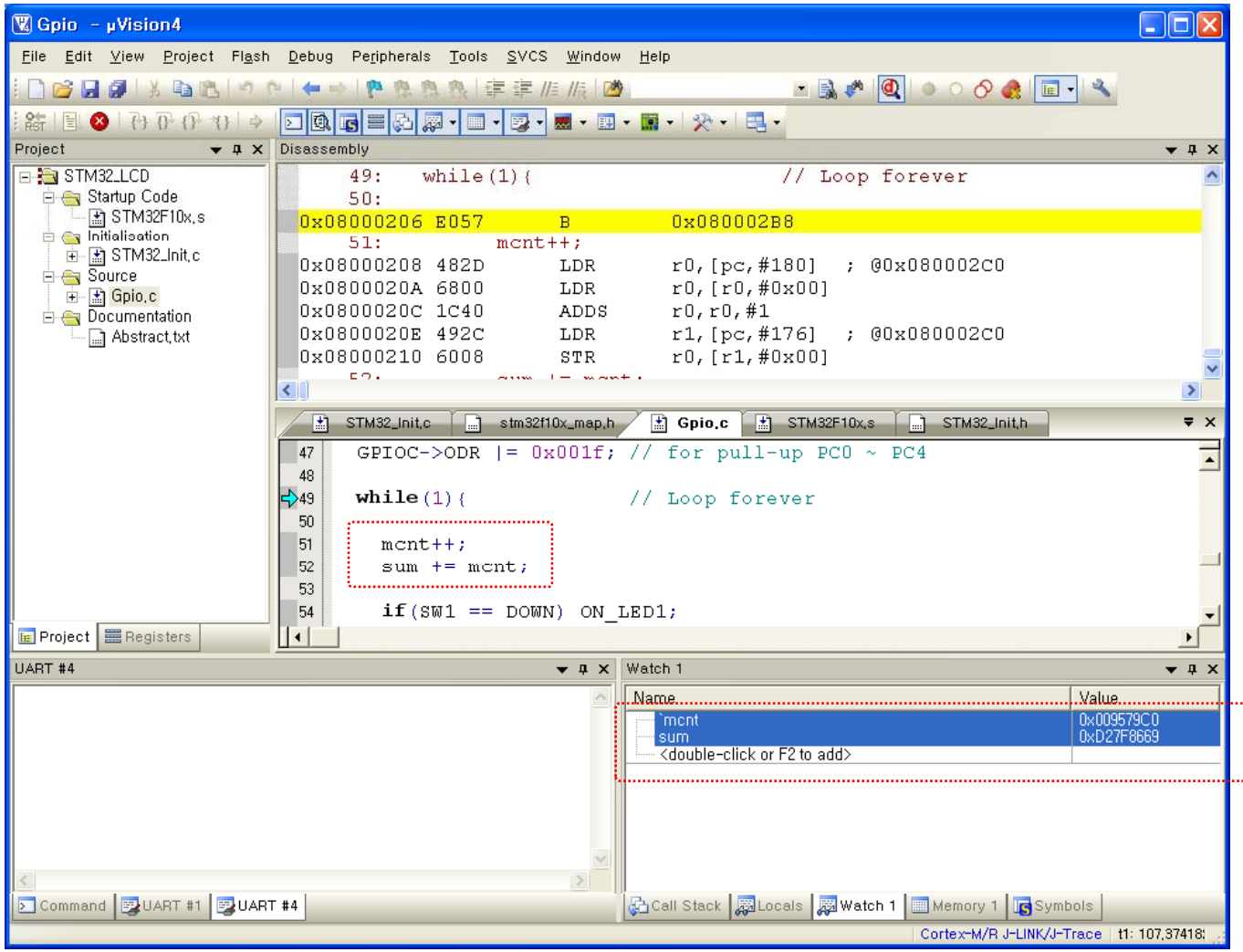
```
void GPIO_Init(GPIO_TypeDef* GPIOx, GPIO_InitTypeDef* GPIO_InitStruct)
{
  uint32_t currentmode = 0x00, currentpin = 0x00, pinpos = 0x00, pos = 0x00;
  uint32_t tmpreg = 0x00, pinmask = 0x00;
  /* Check the parameters */
  assert_param(IS_GPIO_ALL_PERIPH(GPIOx));
  assert_param(IS_GPIO_MODE(GPIO_InitStruct->GPIO_Mode));
  assert_param(IS_GPIO_PIN(GPIO_InitStruct->GPIO_Pin));

/*---------------------------- GPIO Mode Configuration -----------------------*/
  currentmode = ((uint32_t)GPIO_InitStruct->GPIO_Mode) & ((uint32_t)0x0F);
  if ((((uint32_t)GPIO_InitStruct->GPIO_Mode) & ((uint32_t)0x10)) != 0x00)
  {
    /* Check the parameters */
    assert_param(IS_GPIO_SPEED(GPIO_InitStruct->GPIO_Speed));
    /* Output mode */
    currentmode |= (uint32_t)GPIO_InitStruct->GPIO_Speed;
  }
/*---------------------------- GPIO CRL Configuration ------------------------*/
  /* Configure the eight low port pins */
  if (((uint32_t)GPIO_InitStruct->GPIO_Pin & ((uint32_t)0x00FF)) != 0x00)
  {
    tmpreg = GPIOx->CRL;
    for (pinpos = 0x00; pinpos < 0x08; pinpos++)
    {
      pos = ((uint32_t)0x01) << pinpos;
      /* Get the port pins position */
      currentpin = (GPIO_InitStruct->GPIO_Pin) & pos;
      if (currentpin == pos)
      {
        pos = pinpos << 2;
        /* Clear the corresponding low control register bits */
        pinmask = ((uint32_t)0x0F) << pos;
        tmpreg &= ~pinmask;
        /* Write the mode configuration in the corresponding bits */
        tmpreg |= (currentmode << pos);
        /* Reset the corresponding ODR bit */
        /* Set the corresponding ODR bit */
        if (GPIO_InitStruct->GPIO_Mode == GPIO_Mode_IPU)
        {
          GPIOx->BSRR = (((uint32_t)0x01) << pinpos);
        }
      }
    }
    GPIOx->CRL = tmpreg;
  }
/*---------------------------- GPIO CRH Configuration ------------------------*/
  /* Configure the eight high port pins */
  if (GPIO_InitStruct->GPIO_Pin > 0x00FF)
  {
    tmpreg = GPIOx->CRH;
    for (pinpos = 0x00; pinpos < 0x08; pinpos++)
    {
      pos = (((uint32_t)0x01) << (pinpos + 0x08));
      /* Get the port pins position */
      currentpin = ((GPIO_InitStruct->GPIO_Pin) & pos);
      if (currentpin == pos)
      {
        pos = pinpos << 2;
        /* Clear the corresponding high control register bits */
        pinmask = ((uint32_t)0x0F) << pos;
        tmpreg &= ~pinmask;
        /* Write the mode configuration in the corresponding bits */
        tmpreg |= (currentmode << pos);
        /* Reset the corresponding ODR bit */
        if (GPIO_InitStruct->GPIO_Mode == GPIO_Mode_IPD)
        {
          GPIOx->BRR = (((uint32_t)0x01) << (pinpos + 0x08));
        }
        /* Set the corresponding ODR bit */
        if (GPIO_InitStruct->GPIO_Mode == GPIO_Mode_IPU)
        {
          GPIOx->BSRR = (((uint32_t)0x01) << (pinpos + 0x08));
        }
      }
    }
    GPIOx->CRH = tmpreg;
  }
}
```


리얼시스 www.realsys.co.kr Tel:031-342-3000, Fax:031-343-0003 경기도 안양시 동안구 호계동 1027 안양 IT밸리 504호

동작 중 변수 모니터링 가능: 좋은 기능입니다.

[2]. 프로그램 RUN 중에도 변수 값을 보거나 수정할 수 있다.

J-LINK를 사용하여 IAR로 프로그램을 디버깅할 때, Single-Step / Break-point 기능은 됩니다. 하지만 RUN 중에 변수 값은 볼 수 없어요
저는 그렇게 알고 있는데(아니면 알려주세요), KEIL에서는 같은 J-LINK 디버거를 가지고 RUIN 중에 변수 값을 볼 수 있습니다.



이 기능은 개발자 입장에서 엄청나게 편리한 기능입니다.
동작 중에 값을 볼 수 있으면, LCD나 통신 포트를 사용하지 않고도 원하는 값을 보거나 수정할 수 있으니…

이 기능에서 IAR에서 KEIL로 마음이 기울죠???

이거 너무 KEIL 영업해주는 것 같습니다.

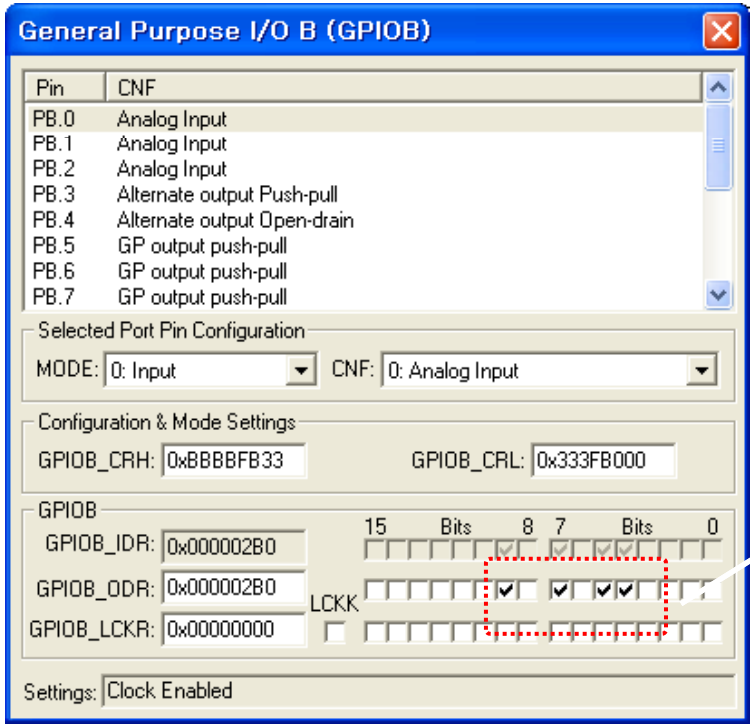
RUN 중에 변수 값 관찰 가능

리얼시스 www.realsys.co.kr Tel:031-342-3000, Fax:031-343-0003 경기도 안양시 동안구 호계동 1027 안양 IT밸리 504호

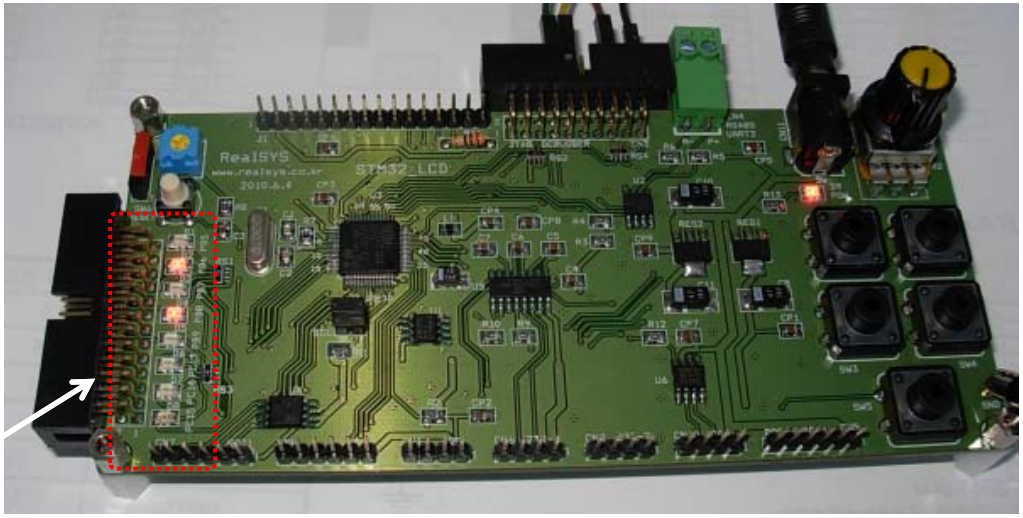
편리한 Target 보드의 입출력 액세스

[3]. 디버깅을 중지하고 수동으로 입출력을 직접 액세스 할 수 있다.

KEIL에는 Target 보드의 입출력을 직접 제어할 수 기능이 있어요. IAR에는 없는 기능이죠



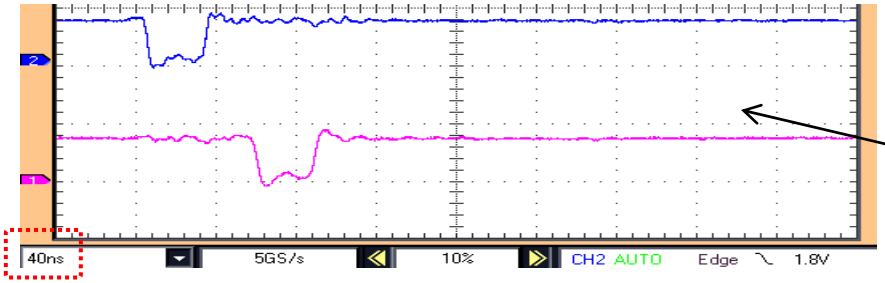
리얼시스(www.realsys.co.kr) STM32_LCD 보드



이런 동작을 해보면서 레지스터 구조나 동작에 친숙하게 되겠지요?

이제 동작 속도를 관찰해 보겠어요.

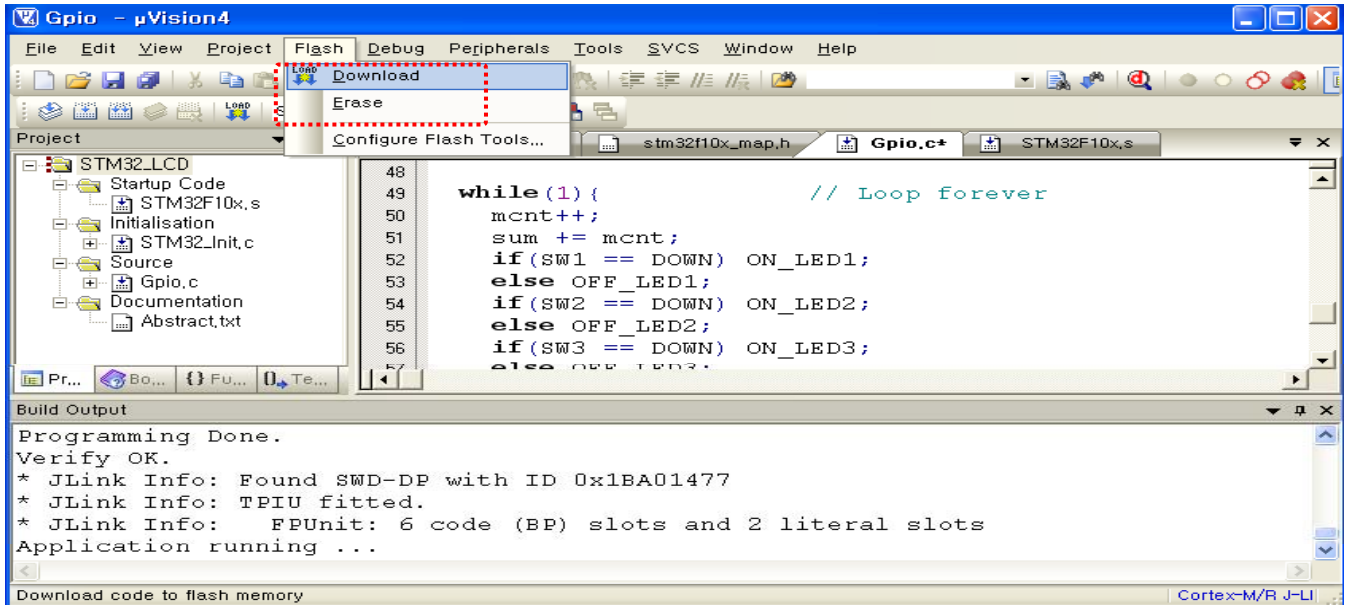
Optimize 하지 않은 상태에서도 30nsec 이하의 빠른 응답 성을 보입니다.




```
while(1){                    // Loop forever
  mcnt++;
  sum += mcnt;
  if(SW1 == DOWN) ON_LED1;
  else OFF_LED1;
  if(SW2 == DOWN) ON_LED2;
  else OFF_LED2;
  if(SW3 == DOWN) ON_LED3;
  else OFF_LED3;
  if(SW4 == DOWN) ON_LED4;
  else OFF_LED4;
  if(SW5 == DOWN) ON_LED5;
  else OFF_LED5;
  ON_LED6;      OFF_LED6;
  ON_LED7;      OFF_LED7;
  T_LED8;
 } // end while
} // end main
```

Flash ROM 다운로드가 편리하다

[4]. 버튼 한 동작에 바로 Flash ROM을 프로그램할 수 있어, 제품 양산 시 편리하게 사용 가능



결론:
KEIL이 IAR에 비해서 여러가지 섬세한 기능을 가지고 있어서 잘 활용하면 엔지니어의 개발 시간을 단축할 수 있다.

그래서 너무 IAR만 고집하지 마시고
KEIL도 사용해보시기를…

[5]. 기타 Logic Analyzer 기능, UART 기능, 평가 기능 등이 있는 것 같은데, 아직 사용을 안 해봐서 잘 모르겠군요.