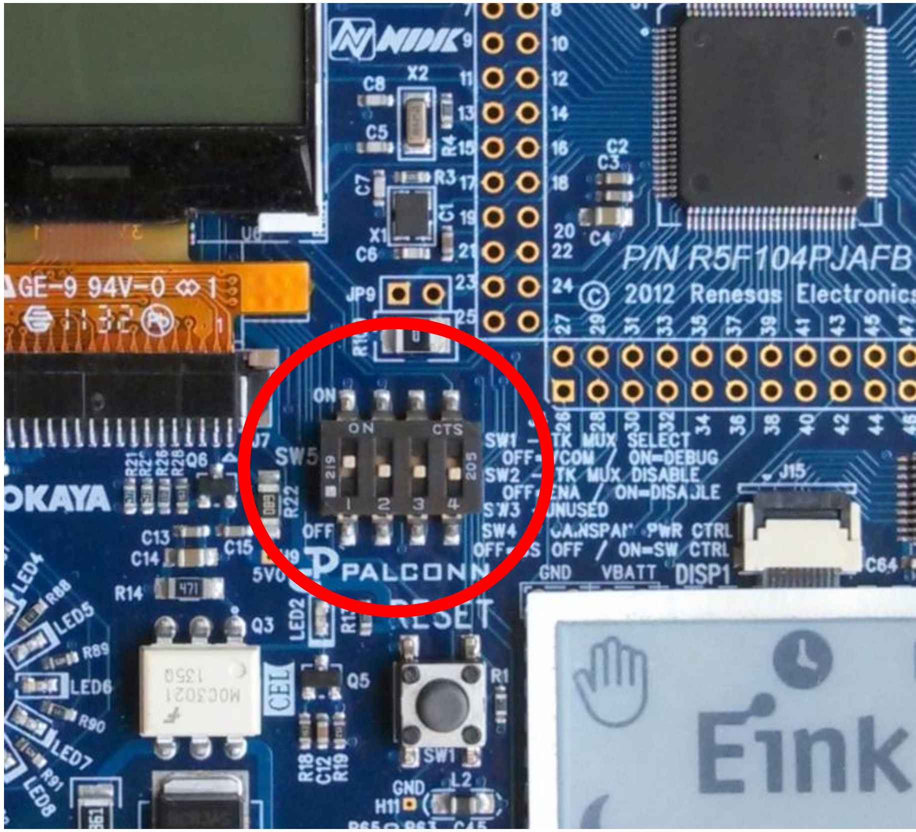
# Getting Started with the RDK

# Set Up

- On your PC
  - Download and unzip a project
    - E.g LCDDemo-G13-G14.zip or LCDDemo-G14.zip
  - Open the EW Workspace file with IAR Embedded Workbench
    - LCDDemo/LCDDemo-G14/LCDDemo.eww

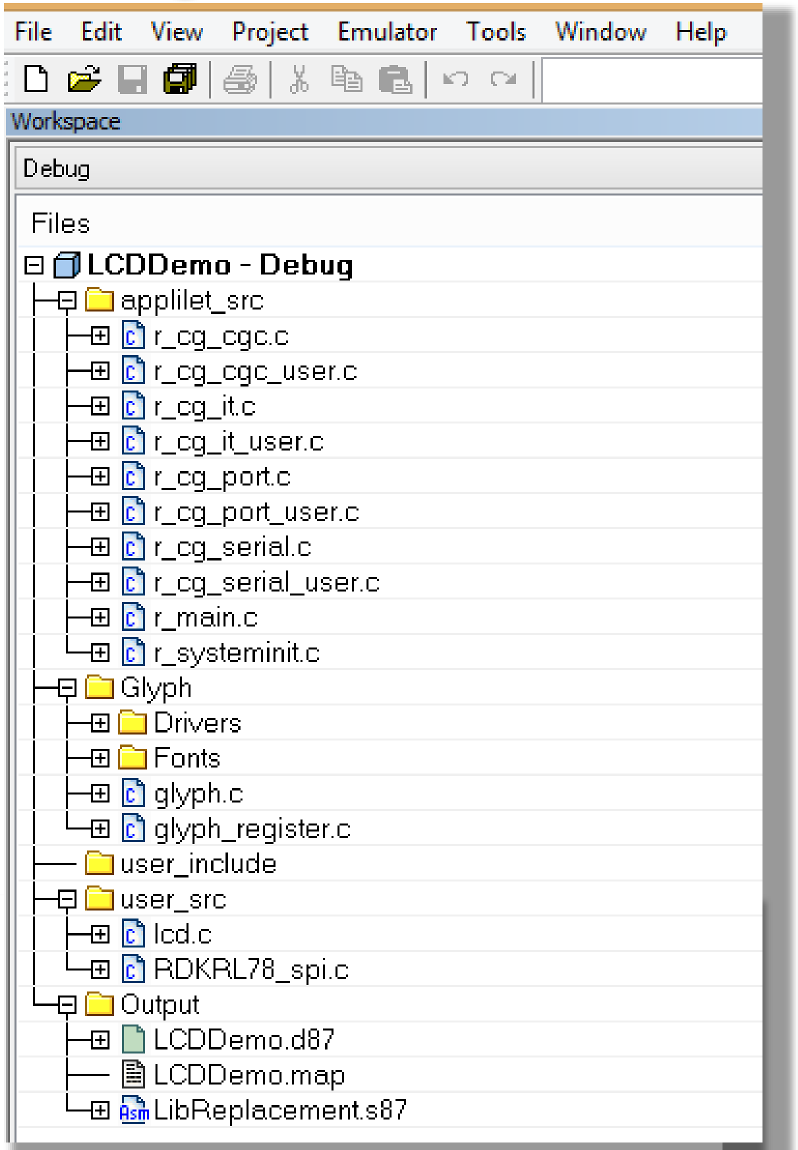
LCDDemo.eww

- On your RDK
  - Set SW5 (the 8 pin DIP switch between the LCD and the Eink display) to enable EW to control the MCU
    - #1 should be ON (up)
    - #2 should be OFF (down)
- Connect the RDK to your PC with a USB cable



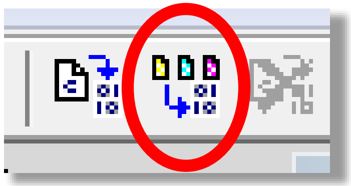
# EW – Building a Program

- Examine source files using the workspace browser

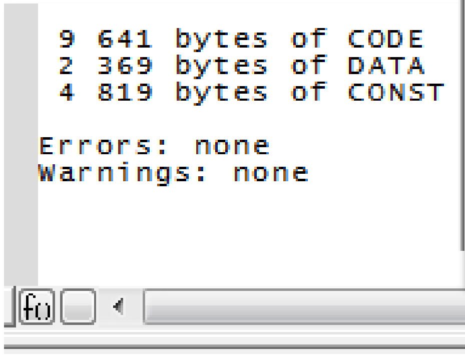


```
⊟ LCDDemo - Debug
  ⊟ applilet_src
    ⊞ r_cg_cgc.c
    ⊞ r_cg_cgc_user.c
    ⊞ r_cg_it.c
    ⊞ r_cg_it_user.c
    ⊞ r_cg_port.c
    ⊞ r_cg_port_user.c
    ⊞ r_cg_serial.c
    ⊞ r_cg_serial_user.c
    ⊞ r_main.c
    ⊞ r_systeminit.c
  ⊟ Glyph
    ⊞ Drivers
    ⊞ Fonts
    ⊞ glyph.c
    ⊞ glyph_register.c
    user_include
  ⊟ user_src
    ⊞ lcd.c
    ⊞ RDKRL78_spi.c
  ⊟ Output
    ⊞ LCDDemo.d87
      LCDDemo.map
    ⊞ LibReplacement.s87
```

- Build the project – F7
  - Compile all source files which have changed since last build, links object files to create executable image



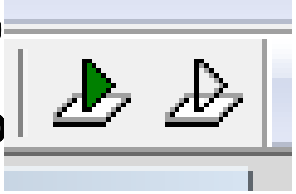
- Verify that build succeeded without errors or warnings



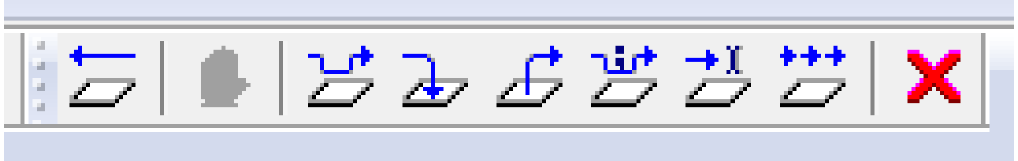
```
 9 641 bytes of CODE
 2 369 bytes of DATA
 4 819 bytes of CONST

Errors: none
Warnings: none
```

# EW – Downloading and Running a Program

- Download and start debugger – Ctl-D
- Empty triangle – Download but do no debugger

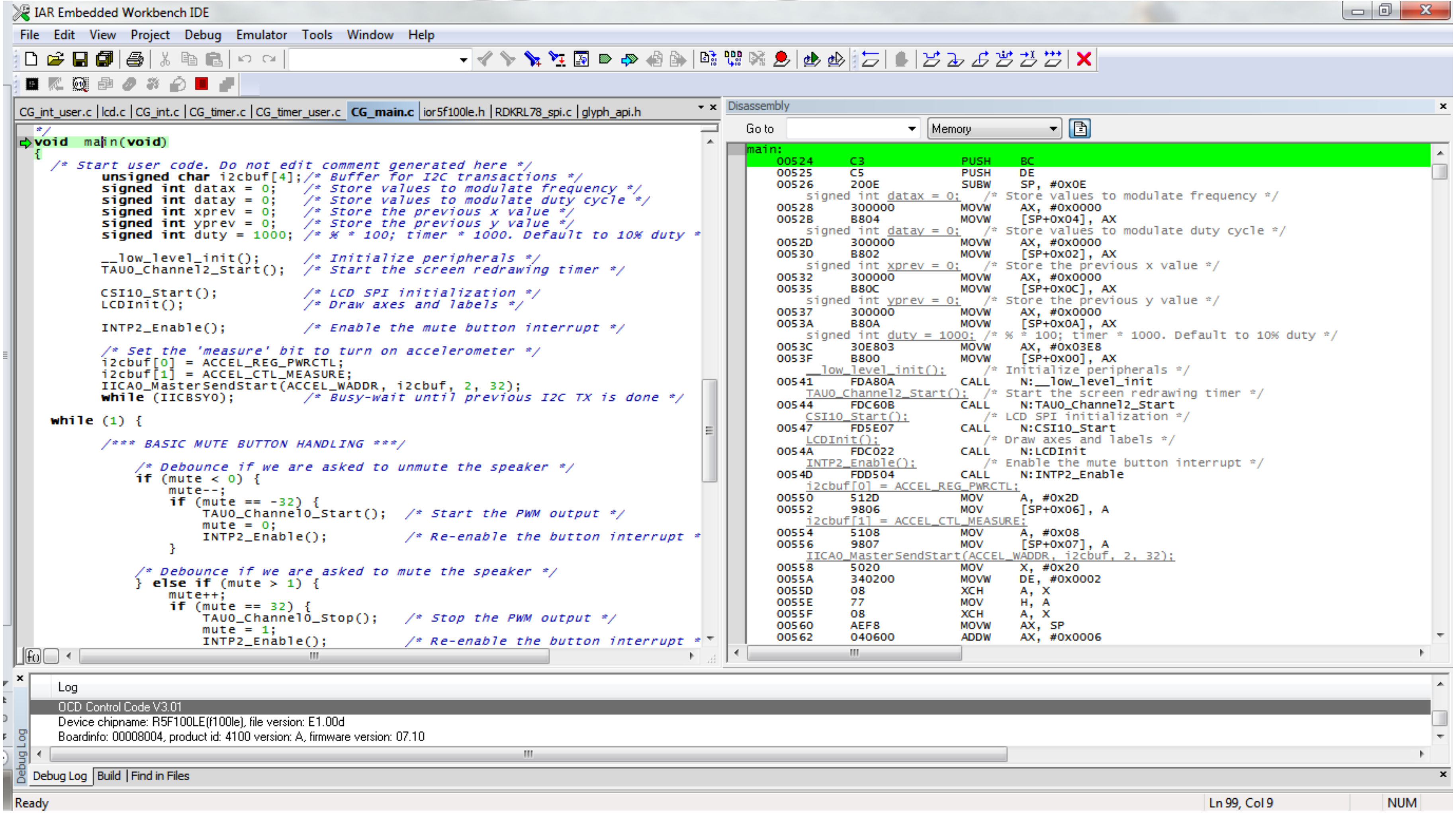


- Running a program



  - Reset
  - Stop a running program
  - Step over function
  - Step into function
  - Step out of function
  - Execute next source code statement
  - Run to cursor
  - Run until stopped or breakpoint reached
  - Exit debugger

# Debugger - Program Views


IAR Embedded Workbench IDE
File Edit View Project Debug Emulator Tools Window Help
CG_int_user.c | lcd.c | CG_int.c | CG_timer.c | CG_timer_user.c | CG_main.c | ior5f100le.h | RDKRL78_spi.c | glyph_api.h
Disassembly
Go to
Memory
Log
Debug Log | Build | Find in Files
Ready
Ln 99, Col 9
NUM

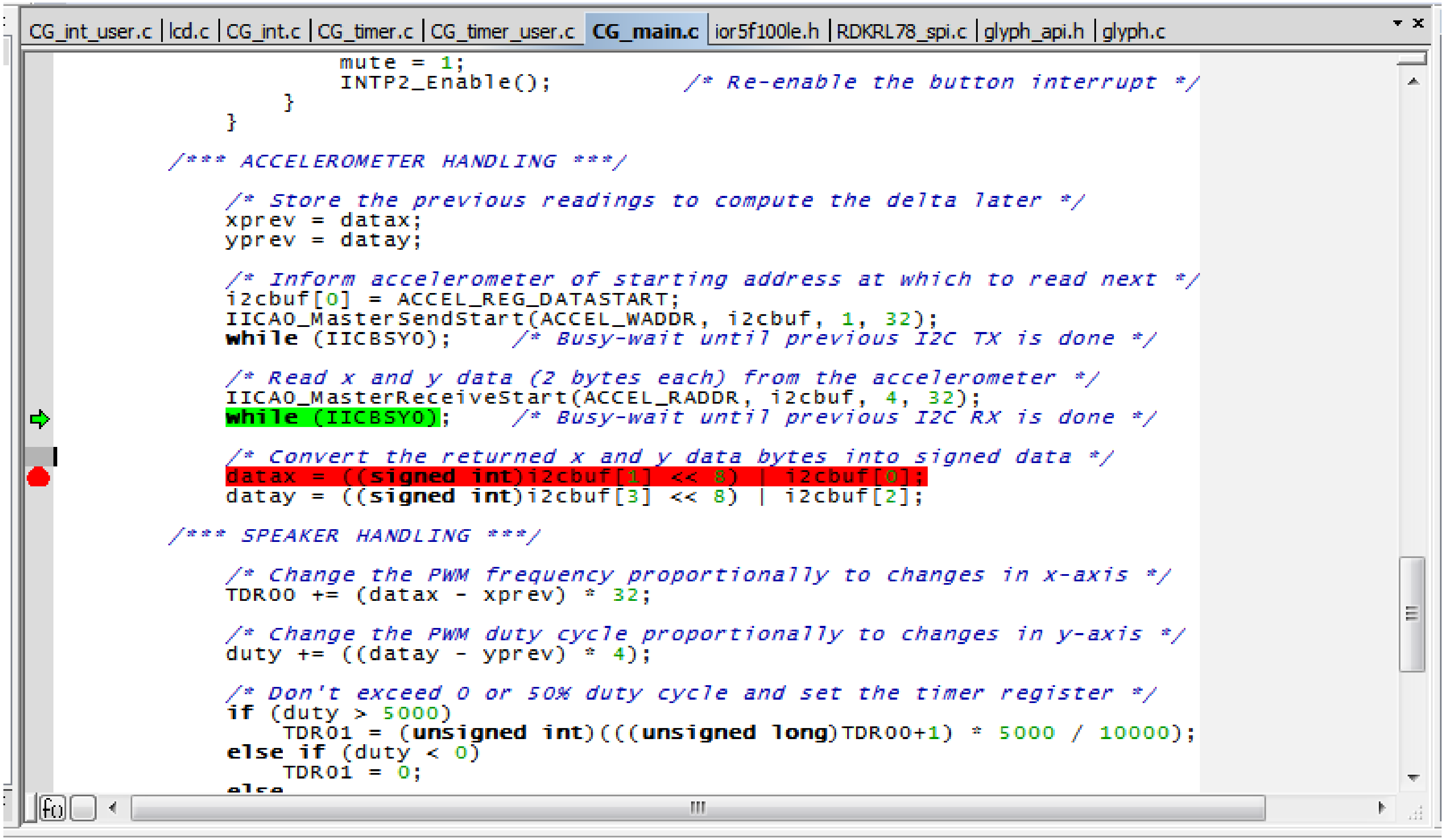
```
  */
void  main(void)
{
  /* Start user code. Do not edit comment generated here */
        unsigned char i2cbuf[4];/* Buffer for I2C transactions */
        signed int datax = 0;   /* Store values to modulate frequency */
        signed int datay = 0;   /* Store values to modulate duty cycle */
        signed int xprev = 0;   /* Store the previous x value */
        signed int yprev = 0;   /* Store the previous y value */
        signed int duty = 1000; /* % * 100; timer * 1000. Default to 10% duty *

        __low_level_init();     /* Initialize peripherals */
        TAU0_Channel2_Start();  /* Start the screen redrawing timer */

        CSI10_Start();          /* LCD SPI initialization */
        LCDInit();              /* Draw axes and labels */

        INTP2_Enable();         /* Enable the mute button interrupt */

        /* Set the 'measure' bit to turn on accelerometer */
        i2cbuf[0] = ACCEL_REG_PWRCTL;
        i2cbuf[1] = ACCEL_CTL_MEASURE;
        IICA0_MasterSendStart(ACCEL_WADDR, i2cbuf, 2, 32);
        while (IICBSY0);        /* Busy-wait until previous I2C TX is done */

  while (1) {

        /*** BASIC MUTE BUTTON HANDLING ***/

            /* Debounce if we are asked to unmute the speaker */
            if (mute < 0) {
                mute--;
                if (mute == -32) {
                    TAU0_Channel0_Start();  /* Start the PWM output */
                    mute = 0;
                    INTP2_Enable();         /* Re-enable the button interrupt *
                }

            /* Debounce if we are asked to mute the speaker */
            } else if (mute > 1) {
                mute++;
                if (mute == 32) {
                    TAU0_Channel0_Stop();   /* Stop the PWM output */
                    mute = 1;
                    INTP2_Enable();         /* Re-enable the button interrupt *
```

```
main:
    00524      C3                PUSH     BC
    00525      C5                PUSH     DE
    00526      200E              SUBW     SP, #0x0E
        signed int datax = 0;    /* Store values to modulate frequency */
    00528      300000            MOVW     AX, #0x0000
    0052B      B804              MOVW     [SP+0x04], AX
        signed int datay = 0;    /* Store values to modulate duty cycle */
    0052D      300000            MOVW     AX, #0x0000
    00530      B802              MOVW     [SP+0x02], AX
        signed int xprev = 0;    /* Store the previous x value */
    00532      300000            MOVW     AX, #0x0000
    00535      B80C              MOVW     [SP+0x0C], AX
        signed int yprev = 0;    /* Store the previous y value */
    00537      300000            MOVW     AX, #0x0000
    0053A      B80A              MOVW     [SP+0x0A], AX
        signed int duty = 1000; /* % * 100; timer * 1000. Default to 10% duty */
    0053C      30E803            MOVW     AX, #0x03E8
    0053F      B800              MOVW     [SP+0x00], AX
        __low_level_init();      /* Initialize peripherals */
    00541      FDA80A            CALL     N:__low_level_init
        TAU0_Channel2_Start();   /* Start the screen redrawing timer */
    00544      FDC60B            CALL     N:TAU0_Channel2_Start
        CSI10_Start();           /* LCD SPI initialization */
    00547      FD5E07            CALL     N:CSI10_Start
        LCDInit();               /* Draw axes and labels */
    0054A      FDC022            CALL     N:LCDInit
        INTP2_Enable();          /* Enable the mute button interrupt */
    0054D      FDD504            CALL     N:INTP2_Enable
        i2cbuf[0] = ACCEL_REG_PWRCTL;
    00550      512D              MOV      A, #0x2D
    00552      9806              MOV      [SP+0x06], A
        i2cbuf[1] = ACCEL_CTL_MEASURE;
    00554      5108              MOV      A, #0x08
    00556      9807              MOV      [SP+0x07], A
        IICA0_MasterSendStart(ACCEL_WADDR, i2cbuf, 2, 32);
    00558      5020              MOV      X, #0x20
    0055A      340200            MOVW     DE, #0x0002
    0055D      08                XCH      A, X
    0055E      77                MOV      H, A
    0055F      08                XCH      A, X
    00560      AEF8              MOVW     AX, SP
    00562      040600            ADDW     AX, #0x0006
```

```
OCD Control Code V3.01
Device chipname: R5F100LE(f100le), file version: E1.00d
Boardinfo: 00008004, product id: 4100 version: A, firmware version: 07.10
```

- Can show source code (C) and corresponding disassembled object code (assembly language)
- Allows controlled execution of program

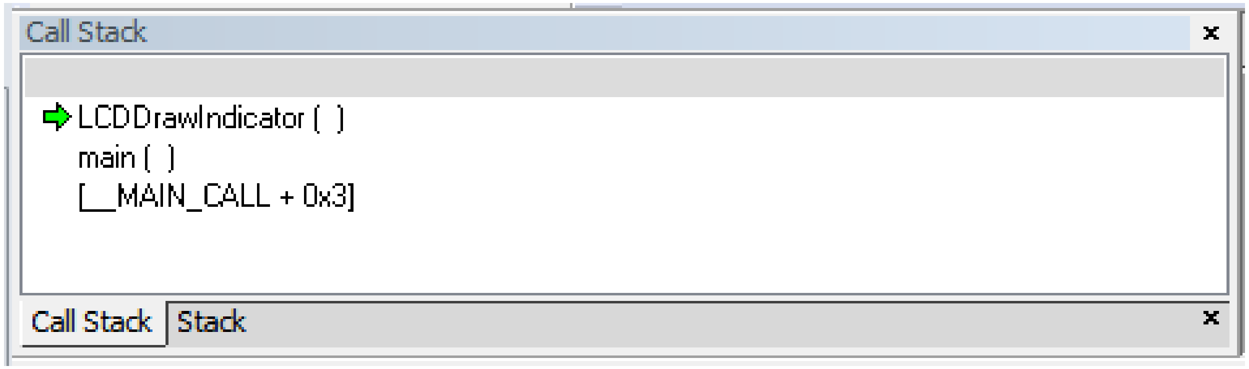
# Debugger - Program Control



```
            mute = 1;
            INTP2_Enable();         /* Re-enable the button interrupt */
        }
    }

/*** ACCELEROMETER HANDLING ***/

    /* Store the previous readings to compute the delta later */
    xprev = datax;
    yprev = datay;

    /* Inform accelerometer of starting address at which to read next */
    i2cbuf[0] = ACCEL_REG_DATASTART;
    IICA0_MasterSendStart(ACCEL_WADDR, i2cbuf, 1, 32);
    while (IICBSY0);    /* Busy-wait until previous I2C TX is done */

    /* Read x and y data (2 bytes each) from the accelerometer */
    IICA0_MasterReceiveStart(ACCEL_RADDR, i2cbuf, 4, 32);
    while (IICBSY0);    /* Busy-wait until previous I2C RX is done */

    /* Convert the returned x and y data bytes into signed data */
    datax = ((signed int)i2cbuf[1] << 8) | i2cbuf[0];
    datay = ((signed int)i2cbuf[3] << 8) | i2cbuf[2];

/*** SPEAKER HANDLING ***/

    /* Change the PWM frequency proportionally to changes in x-axis */
    TDR00 += (datax - xprev) * 32;

    /* Change the PWM duty cycle proportionally to changes in y-axis */
    duty += ((datay - yprev) * 4);

    /* Don't exceed 0 or 50% duty cycle and set the timer register */
    if (duty > 5000)
        TDR01 = (unsigned int)(((unsigned long)TDR00+1) * 5000 / 10000);
    else if (duty < 0)
        TDR01 = 0;
    else
```

- Green - next line of source code to execute
- Red - user-defined breakpoint (right click on source code to set or delete)

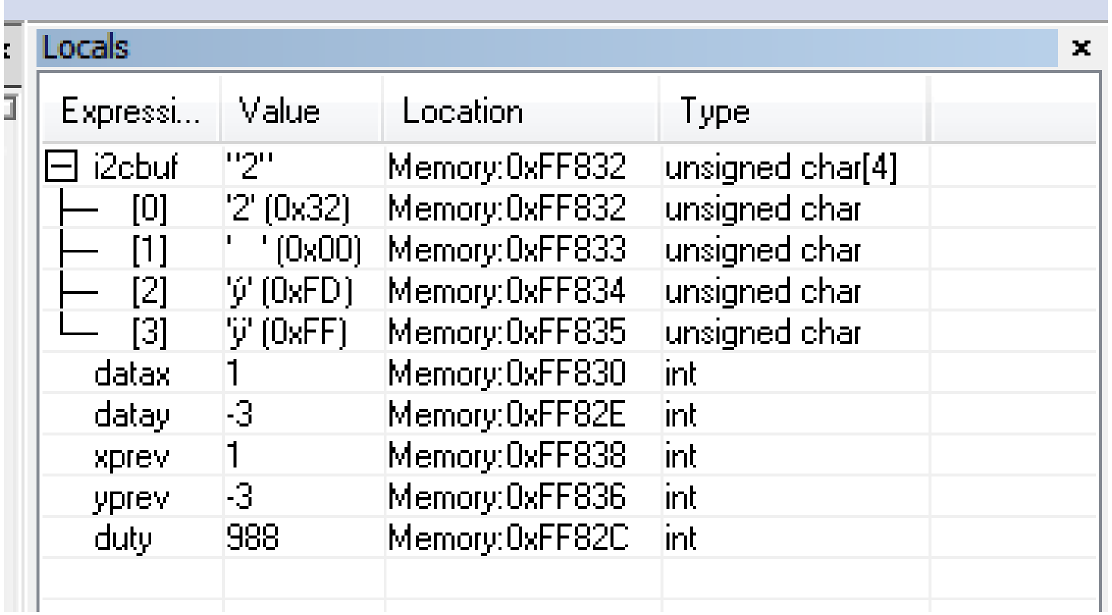
# Function Call Stack



```
LCDDrawIndicator ( )
main ( )
[__MAIN_CALL + 0x3]
```

- Currently executing LCDDrawIndicator
- … which was called by main
- … which was called by an instruction at address MAIN_CALL+3

# Debugger - Data Examination & Modification



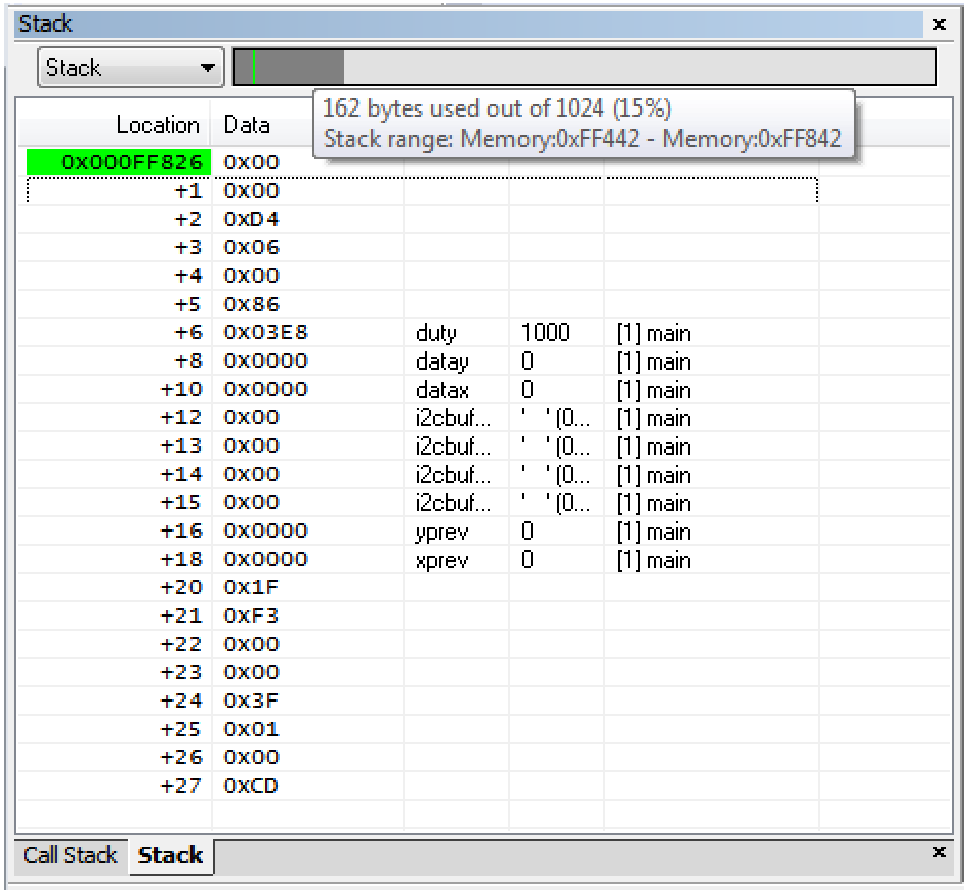
Locals

| Expressi... | Value | Location | Type |
|---|---|---|---|
| ⊟ i2cbuf | "2" | Memory:0xFF832 | unsigned char[4] |
| ├─ [0] | '2' (0x32) | Memory:0xFF832 | unsigned char |
| ├─ [1] | '  ' (0x00) | Memory:0xFF833 | unsigned char |
| ├─ [2] | 'ý' (0xFD) | Memory:0xFF834 | unsigned char |
| └─ [3] | 'ÿ' (0xFF) | Memory:0xFF835 | unsigned char |
| datax | 1 | Memory:0xFF830 | int |
| datay | -3 | Memory:0xFF82E | int |
| xprev | 1 | Memory:0xFF838 | int |
| yprev | -3 | Memory:0xFF836 | int |
| duty | 988 | Memory:0xFF82C | int |

- Can use debugger to examine variables (locals, globals) and parameters
- Value can be displayed in various formats (char, string, hex, integer, etc.)

# Debugger - Stack Contents

- Can use debugger to examine what’s on the stack
- Not just automatic variables: return address, stack arguments, temporary storage, etc.
- Can see how much of stack space is used

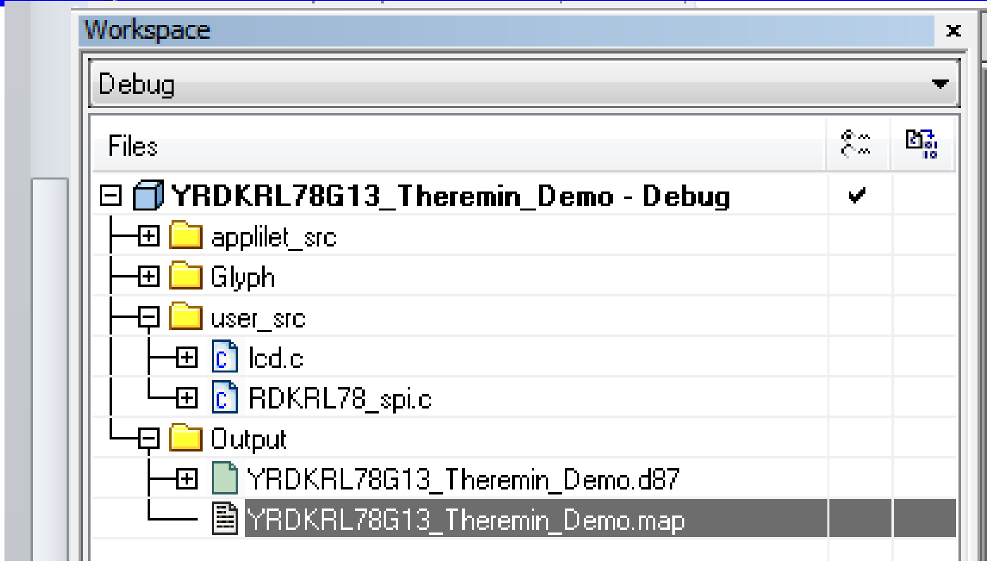


162 bytes used out of 1024 (15%)
Stack range: Memory:0xFF442 - Memory:0xFF842

| Location | Data | | | |
|---|---|---|---|---|
| 0x000FF826 | 0x00 | | | |
| +1 | 0x00 | | | |
| +2 | 0xD4 | | | |
| +3 | 0x06 | | | |
| +4 | 0x00 | | | |
| +5 | 0x86 | | | |
| +6 | 0x03E8 | duty | 1000 | [1] main |
| +8 | 0x0000 | datay | 0 | [1] main |
| +10 | 0x0000 | datax | 0 | [1] main |
| +12 | 0x00 | i2cbuf... | '  '(0... | [1] main |
| +13 | 0x00 | i2cbuf... | '  '(0... | [1] main |
| +14 | 0x00 | i2cbuf... | '  '(0... | [1] main |
| +15 | 0x00 | i2cbuf... | '  '(0... | [1] main |
| +16 | 0x0000 | yprev | 0 | [1] main |
| +18 | 0x0000 | xprev | 0 | [1] main |
| +20 | 0x1F | | | |
| +21 | 0xF3 | | | |
| +22 | 0x00 | | | |
| +23 | 0x00 | | | |
| +24 | 0x3F | | | |
| +25 | 0x01 | | | |
| +26 | 0x00 | | | |
| +27 | 0xCD | | | |

# Project Output Files

- Output of linker
  - Executable program
  - Map file describing memory use



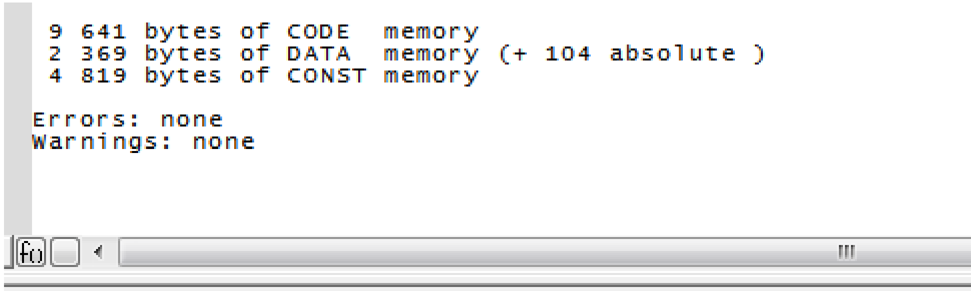
Workspace

Debug

Files

```
⊟ YRDKRL78G13_Theremin_Demo - Debug   ✓
  ├⊞ applilet_src
  ├⊞ Glyph
  ├⊟ user_src
  │  ├⊞ lcd.c
  │  └⊞ RDKRL78_spi.c
  └⊟ Output
     ├⊞ YRDKRL78G13_Theremin_Demo.d87
     └─ YRDKRL78G13_Theremin_Demo.map
```

- Example of map file
  - CODE - Instructions - ROM
  - DATA - Data which may be changed - RAM
  - CONST - Data which will not be changed - ROM



```
  9 641 bytes of CODE   memory
  2 369 bytes of DATA   memory (+ 104 absolute )
  4 819 bytes of CONST  memory

Errors: none
Warnings: none
```