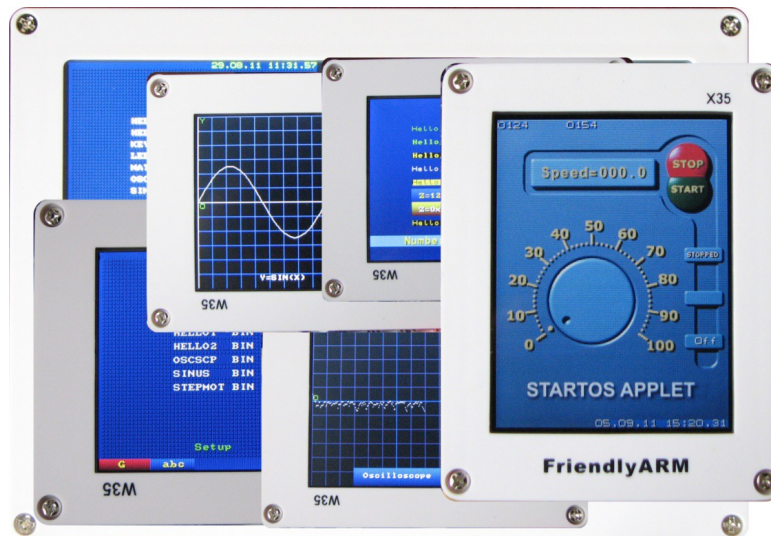


RT-210

(Formerly STARTOS)

User's Manual



May 5, 2015

Introduction

The system may be ported and adapted on any ARM platform. Previous realization was for ARM9 and Mini2440 board and is known as StartOS (The StartOS name was registered by Russia PO 19 Aug 2011). Here the version for ARM Cortex-A8 will be considered. It intended for the Samsung S5PV210 SoC and the FriendlyARM's Mini210s board.

The FriendlyARM's board "Mini210s" attracts users by its possibilities and low cost.

Mini210s is useful for developing industrial automation systems.

Many specialists try to carry the properties of mainframe computers to the Friendly ARM. But there is also another approach, from simple to complex i.e. from rapidly developing MCUs.

Computer market offers desktop computers, notebooks, Net books, tablets. At the present time in industrial automation field the Mini2440 and Mini210s board has fewer number of competitors, taking in consideration the set of external devices mounted aboard, jacks for their connection and wide range of LCD screens.

Many users successfully working with traditional computers and microcontrollers meet difficulties developing and applying ARM systems.

All that impelled the author to create not complex system for industrial automation, laboratory purposes and as an instrumental aid for scientific experiments. STARTOS was created by engineer and for engineers.

A system aimed to contain functions, which we like to use: simplicity, reliability, launching programs from external drives, providing the user with the system functions through the mechanism of program interrupts. The system does not limit the user possibilities but gives him the services. It executes all duty work allowing the user to think about his main task.

It was built to be clear, easy to use, quick for learning and applying by engineers, students and specialists in other non-programming fields.

Functions and developing process are widely illustrated with working tested examples.

Quick start

In order to briefly verifying system abilities, not in details, it's necessary to do the following:

First, please watch a short video about it: <https://youtu.be/z9cYu7sYwV0>

Go to Download page http://physicalcomputing.ru/articles_en/p_rt-210_en.html

- Download SD-Flasher utility, FriendlyARM's Superboot binary file, SD Card .RAR Archive;
- Prepare a SD card (it may be SDSC or SDHC) using SD-Flash Utility;
- Write the folder "images", binary example files and .BMP file onto SD Card Root;
- Put the SD Card in the Mini210s slot, move switch to SD boot position;
- Turn the power switch On.

First time System will ask to calibrate the touch screen.

Press [>] in the left lower corner of the screen. List of files from SD card will appear.

Next, click on the file name with the stylus, for example, tap Hello1.BIN. It must begin to work.

System properties

The system is written using C and Assembler languages. Actually it is the kernel with the User interface and minimal set of necessary functions. That is why it had no beautiful appearance with clock and calendar. (This may be done by making file Start.bin - it will be playing the Shell's role. For industrial purposes it's not needed).

Start time after power on:	about 1 s
Size of program:	less than 33 KBytes
SDRAM loading address:	0x3F500000.

System occupies a small volume in the upper SDRAM area providing the user's standard programs loading from address 0x20000000 (very beginning of the RAM) and memory volume of 0x3F500000-0x20000000= 525336576 Bytes (almost all 512 MB).

Autorun User's program after power on presents
If the file named START.BIN is present on the SD card, it *will be launched automatically*.

The user is granted to have full control over hardware resources.

System functions

- Initialization the SoC (system on chip) Samsung S5PV210 external devices;
- User programs loading and inter-working with system functions;
- Working with the LCD in text and graphic modes (output Text, Pixels, Lines, Boxes, BMP pictures, Saving / Restoring Screen areas);
- Reading pen X, Y coordinates of the touch screen;
- Working with the RTC (real time clock);
- Reading data from analog to digital converters (ADC);
- Reading / Writing Data to / from digital ports;
- Reading / Writing Bytes and Strings of the COM Port;
- Working with pulse-width modulators (PWM) and Piezo buzzer, Sound output 8 bits per sample 16KHz sample rate is supported as well;
- Reading/writing files from/to the SD card.

Networking (Receive/Transmit Ethernet packets, TCP IP Web Client/Server) as was in StartOS did not implemented yet, may be done if there will be the Users interest.

Screen calibration data are stored in EEPROM.

Work with SD cards is very useful for embedded systems for changing and upgrading programs. It also preserves the NAND Flash memory with limited write cycles and soldered into the system board.

System was tested and verified on SDSC and SDHC cards with the FAT32 file system. Card's Class (data transfer speed) determines the loading time.

The system is using the Interrupts Driven model. The user must avoid implementing system's software interrupts in Interrupt Service Routines (ISR). The ISRs must be short and work with Hardware resources only and set Flags for Main Program. However, it can be changed. Doing this, the System must understand from which Processor Mode it came into the Interrupt and set the Processor Mode in proper status. It may be done automatically.

Also, the system can be forced to execute Data arrays as instructions. It is useful for experimenting with Evolutionary Algorithms (Genetic Algorithms in particular), for creating the self-modified Program Code.

Programmer's interface

Function description

The system is on developing with adding new functions.

Exit (void);

Exit the user's program.

Delay (int ms); Delay_uS(int uS);

Delays execution of the program. Parameters : number of milliseconds or uSeconds

Screen operations

Fill_Screen (int color);

Fill screen by the color. Parameter: color (depth 16 bits per pixel)

Text

Set_Font (char Bold, int Fore_Col, int Back_Col);

Changes the current font. Parameters: 1st: 1-bold, 0- normal; 2nd: color of symbols, 3rd: color of the background, (if equal 1, background will be transparent).

Print_String (int x0, int y0, char *txt);

Outputs the String of text to the LCD. Parameters: 1st and 2nd - abscissa and ordinate of the line beginning position in pixels, 3rd: pointer to text buffer or the line itself in quotation-marks.

Examples:

```
char txt [ 5 ] = { "Hello" };      // or char *txt="Hello"
```

```
Print_String ( 30, 50, txt );
```

```
Print_String ( 30, 70, "Hello, World!" );
```

The text is printed by previously defined font.

```
Printf (int x, int y, char *fmt,...);
```

Prints out the line of formatted text

Examples:

```
z=0xCDEF12;
```

```
Printf (80, 170, "Z=0x%x", z);
```

Graphics

Set_Color (int Color);

Establishes a current color for functions Put_Pixel, Line and painted rectangle (Box).

Put_Bmp (int x0, int y0, char *bmp);

Put out bitmap image on the LCD with the coordinates of the left upper corner x0 and y0. 3rd parameter points onto the image buffer with type "char". First 4 bytes in the buffer must represent: first pair - width of the image, second pair - height.

Example:

```
char run [ ] =          // w = 0032, h = 0006
```

```
{ 32, 0, 6, 0,   0xff, 0xf0, 0xff, 0xf0, 0xff, 0xf0, 0x00, 0x00, 0x54, 0x14, 0xff, 0xf0, 0x00,
  0x00, 0x54, 0x14, 0xff, 0xf0, 0x00, 0x00, 0xf.....}
```

Put_Bmp (198, 139, run);

Draws image with = 32 and height = 6 pixels in the screen area with the coordinates of upper left corner x=198, y=139. Pixels of images are described in user char array run [].

Put_Pixel (int x, int y, int color);

Puts one dot pixel on the LCD with coordinates "x" and "y" and color "color".

Get_Pixel (int x, int y);

The function returns color value of the screen point with coordinates x and y in user variable, for example:

```
int      pix_color;
```

```
pix_color = Get_Pixel (60, 80);
```

Put_Line (int x0, int y0, int x1, int y1);

Draws the line on the LCD which begins at x0 and y0 and ends at x1 and y1 coordinates. Color is determined by last operator Set_Color.

Save_Box (int x0, int y0, int w, int h);

Stores rectangular area of the screen with coordinates of left upper corner x0 and y0, width w and height h in system's inner buffer.

Restore_Box (int x0, int y0);

Restores the rectangular area onto the screen with the coordinates of the left upper corner x0 and y0, with width w and height h which was stored in the system's inner buffer.

Box (int x0, int y0, int x1, int y1);

Draws rectangle on screen with coordinates x0, y0, x1, y1. The color of rectangle monotonously subsides from value given by last operator Set_Color from top to bottom. May be useful for drawing buttons, bars, etc.

LCD Control

TS (char brit);

Set the screen back light LEDs brightness.

The parameter must be in 0...127 limits.

Example.

TS (120); - set LCD brightness as 120

Touch Screen


```
int    TS ( );
```

Returns the Stylus and X, Y state and values

Example: usage please see at examples

```
int    pen;
```

```
int    x,y;
```

```
pen = TS( 0x40 );    // Read Touch Screen Status, X, Y
```

```
if    (pen )
```

```
{
```

```
    x = pen&0x3FF; y = (pen>>10 );    // each 10 bit long
```

```
    Printf ( 350, 20, "x = %4d", x ); Printf ( 350, 40, "y = %4d", y );
```

```
}
```

Real Time Clock (RTC)

```
Get_RTC (char *txt);
```

The function fills the txt char array by date and time in format: dd.mm.yy hh:mm:ss Then the user program can change the order of digits if needed and use its' values.

Timer

```
Get_ms ( );
```

Returns number of milliseconds have passed since the last system start. The Timer4 is initialized and used as the system timer. Also it may be used for switching between tasks.

Example.

```
ms=Get_ms();
```

SD card

```
int          Read_SD    (char *RAM_address, U32 SD_address, U16 block);
```

```
int          Write_SD   (char *RAM_address, U32 SD_address, U16 block);
```

Direct reads/writes areas on the SD card.

Input: pointer to the buffer or physical address in RAM, SD card address, number of blocks. One block equal 512 Bytes. Useful for custom file systems or fast transfer.

Files

int Open_File (char *fname);

Opens the file with specified file name. Returns the length or 0 if file not exists.

F_Len = Open_File(snd_tst);

```
int Read_File(char *f_buf, int len);
```

Reads len number of Bytes from the open file into the buffer or address.

Example.

```
int      F_Len;

char     SND_Buffer  [100000];           // Snd File_Buffer

char     PIC_Buffer  [260000];           // Pic File_Buffer

char     snd_tst[11]  = {'S','O','U','N','D','T','S','T','D','A','T'};
```

```
F_Len = Open_File( snd_tst );
```

```
Printf (0,50,"Sound File Len = %04d", F_Len );
```

```
Read_File (SND_Buffer, F_Len);
```

```
PSound ( SND_Buffer, F_Len );
```

```
int Write_File (char *f_buf, int len);
```

Example.

```
char    txt1_tst[11]    = {'T','E','X','T',' ','I','N',' ','T','X','T'}; // put spaces when no letters
char    txt2_tst[11]    = {'T','E','X','T',' ','O','U','T','T','X','T'};

F_Len = Open_File( txt2_tst );           //      open file for writing

Write_File ( W_Buff, F_Len );             //      write the data
```

Notes:

1. File for writing must be previously created on your PC and have sufficient size
2. The buffer for file's data must be big enough to accept the data. It must be aligned by the cluster size. For instance, you plan to write the BMP file for 240x320 and 24 bpp. It has size 230454 Bytes including the Head. So, the buffer will be at least 245760 Bytes. The number $245760 = 15 \text{ clusters} \times 16384 \text{ Bytes in one cluster}$. 16384 is the max cluster size for FAT32. You may choose from 512 to 16384 while formatting the card. More the cluster size, more the speed.

Simple method: just look at the file's size on your PC and add some value.

Analog To Digital Donverter (ADC)

Get_ADC (char ADC_chan);

Returns measured voltage applied to the input pins of ADC (channels 0...5) into user variable.

Example:

```
int    ADC_data;  
  
ADC_data = Get_ADC( 0 );  
  
Set_Font ( 1, Yellow, 20);  
  
Printf ( 50, 70, "ADC Channel 0 = %03d", ADC_data );
```

Pulse width modulator (PWM) and Piezo Buzzer

PWM_out2 (U8 val); // this is the PIN28 at CON6

Outputs the pulse width modulated signal on corresponding pins. Val must be in 0...255 limits. Useful for analog output and the servo motor control.

Buzz (int freq, int ms);

Connects the on board Piezo buzzer to the first PWM channel output and the Buzzer produces sound with frequency freq and duration ms.

Buzzer_Freq (int freq);

Defines the frequency produced by PWM in Hertz.

Buzzer_Out (char on_off);

Connects (if parameter=1) or disconnects (if parameter=0) PWM Out from Piezo Buzzer. So it will produce sound or keep silence.

Sounds using PWM, Pezospeaker or Amplifier with Loudspeaker

void PSound (U8 *buf, int len);

Plays Sound Data from buffer *buf* with length *len*.

The function uses the PWM Timer0 in Interrupts Mode. Immediately returns control to the User program while keeping the sound playing.

Example:

```
PSound      ( sound1, 5946 );
```

```
PSound      ( sound1+offset, 5946-offset ); // Plays not from the beginning
```

Sound Data is prepared by Utility supplied. It must be 8-bits per sample, 16 kHz sample rate.

You will hear sounds, produced by on board Piezospeaker.

Do append sequence ...128,100,80,60,40,20 at the End of Sound array to avoid "click" if needed.

Working with the IIC bus

```
U8          IIC_read      ( U8 slave_addr, int addr);  
void        IIC_write     ( U8 slave_addr, int addr, U8 data);
```

The first reads and returns the byte of data from a device on the IIC bus with pointed slave address and inner address in the device's memory raw.

The second writes byte of data into the device.

Some devices support autoincrement of addresses. In such cases the pointer to raw data must be used and the number of bytes to be transferred. It's more complex to novices. Interested users may order this method or develop their own.

The *compass* sketch shows how to work with EEPROM and accelerometer/compass chip.

Universal asynchronous receiver-transmitter **(UART, COM Port)**

```
Uart_Init (char chan, int BaudRate);
```

Initializes UART and assigns Data exchange speed.

Example: Uart_Init (0 , 115200); - sets channel 0 and speed 115 KBod

```
Uart_Select (char chan);
```

Select one of three UART channels

Uart_Printf (char *fmt, ...);

Sends a line of formatted text into a COM port.

Uart_TxEmpty (char chan);

Flushes the UART shift register data in the channel pointed. (Clears it and prepare to work).

char Uart_GetChar (char chan);

Gets UART data byte, stores it in User variable with waiting for the data's appearance.

char Uart_GetKey (char chan);

Gets UART data byte, moves it in User variable without waiting of data get ready.

If the data in register exist, they will be read, if no, program continues its execution.

Uart_GetString (char *string);

Inputs line of symbols from the port until buffer of the line will ends or will be accepted symbol \r (<lf>+<cr>, key [Enter]).

Uart_SendByte (char data);

Sends a byte of data into a COM port.

Uart_SendString (char *pt);

Sends a line of symbols into COM port. Parameter – pointer to line or line by itself closed in quotation-marks.

Miscellaneous functions

Dec2Asc (int num, char *txt);

Translates decimal number num into char type and places it in txt array.

Hex2Asc (int num, char *txt);

Translates hexadecimal number num into char type and writes it into the txt array.

Network functions*

Networking is in StartOS, in RT-210 is not implemented yet, but may be done

Network interface controller initialization

char NIC_init (char *MAC_addr);

res = NIC_Init (macaddr);

Initialize the NIC with the MAC address.

Returns error codes or the connection parameters (the speed, half/full duplex).

Packet receive

int NIC_rx (U16 *RX_buffer);

rx_len = NIC_rx(RX_data));

Reads the Ethernet packet into the buffer, returns the length of the packet or zero.

Transmit packet

void NIC_tx (U16 *TX_buffer, U16 Nbytes);

NIC_tx (TX_data, 70);

Send the packet from buffer with length in Bytes.

The Web Server

Server (80, Mini210s_ip, page1, strbuf);

Input: the Port number, the source IP address, the buffer with Page.

Returns the String (up to 10 Bytes) which contains arguments got from the Browser to perform analyze and processing.

The Client

res = Client (Mini210s_ip, server_ip, request, buf);

Input: the source, destination IP addresses, request to Server.

Output: the pointer to buffer containing the Page received from the Server and res=number of Bytes received.

More detailed Networking description see in **Appendix 2**.

Program examples

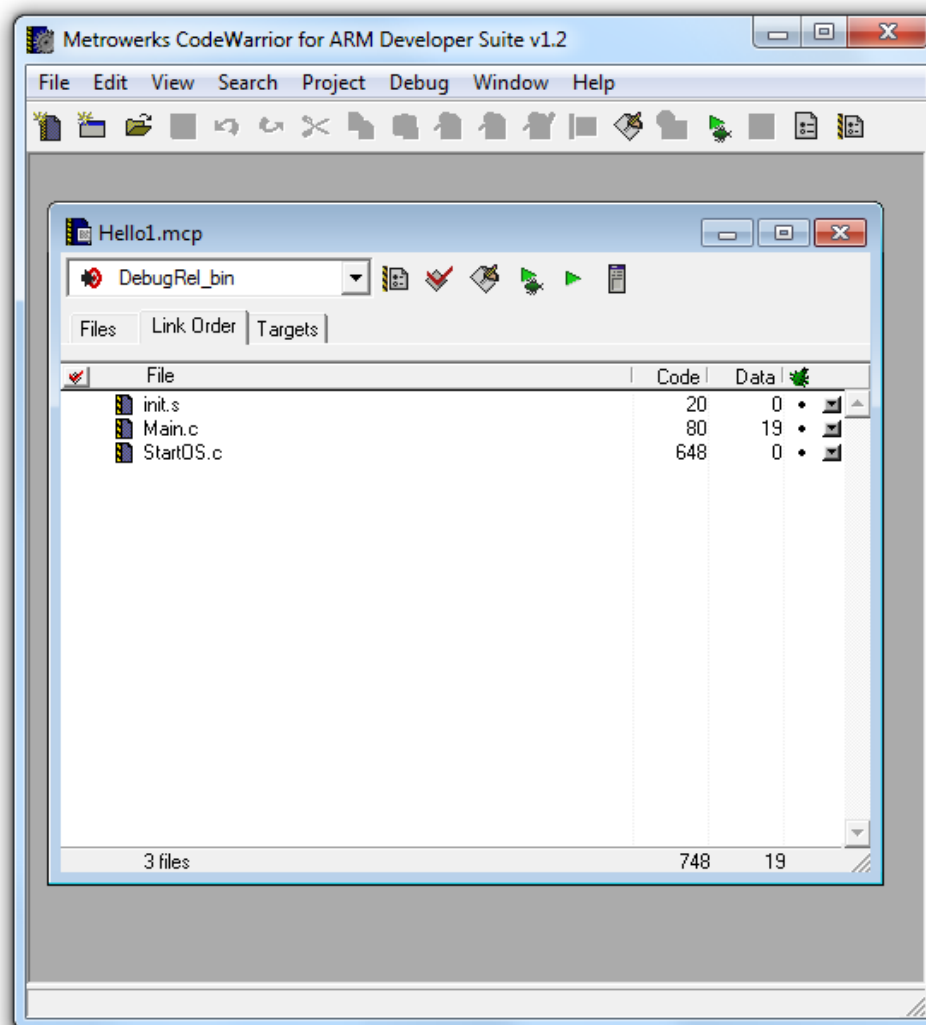
Please download Sketches Examples and write them into C:/Exam_210 folder, then go inside some example folder and double click the .MCP file. Edit the Main.c file, press [F7] and you will get .BIN file.

Some examples even does not need an assembler initialization file.

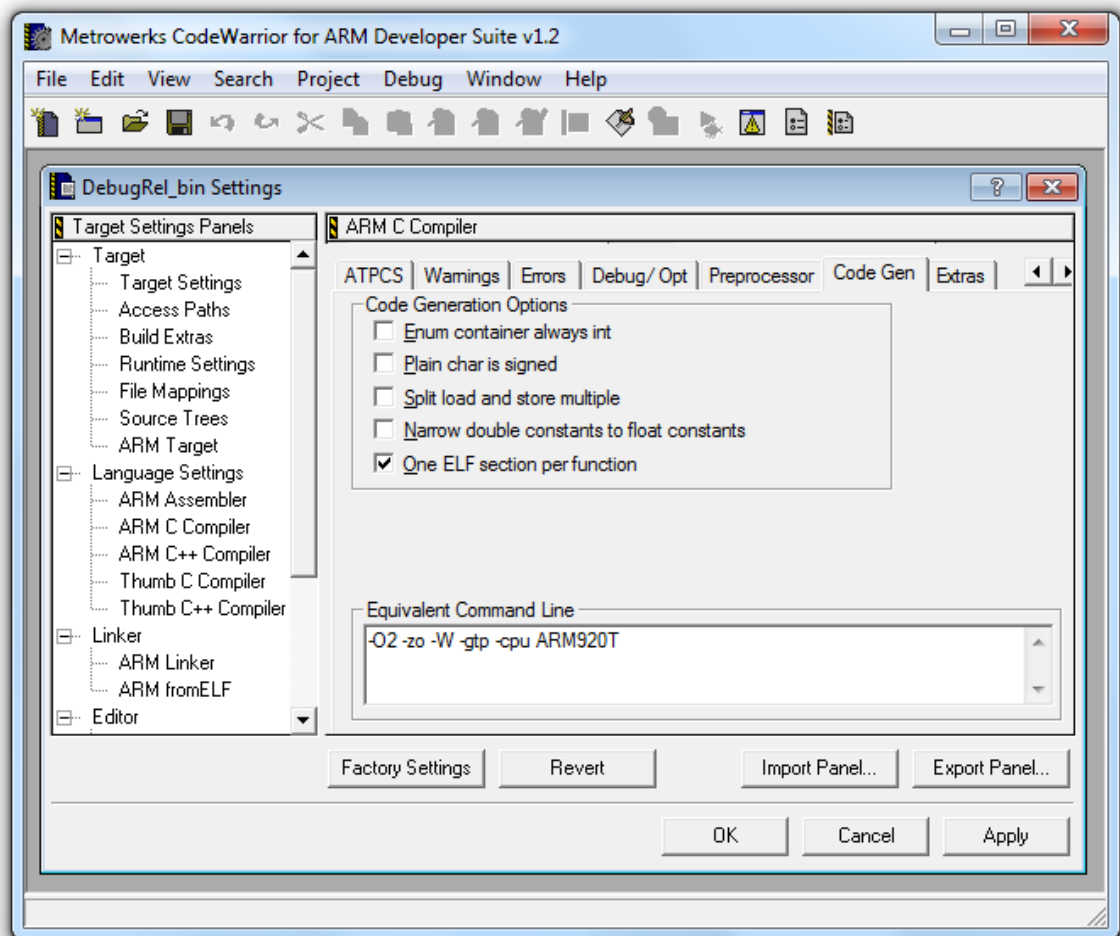
Projects were prepared in Metrowerks CodeWarrior for ARM Developer Suite v1.2. It is assumed, that environment is installed on your computer. Also some projects were ported in Keil uVision4 IDE and IAR.

There are enough only three files for a project:

- start.s - system Initialization;
- StartOS.c - description and system functions;
- Main.c - your program itself



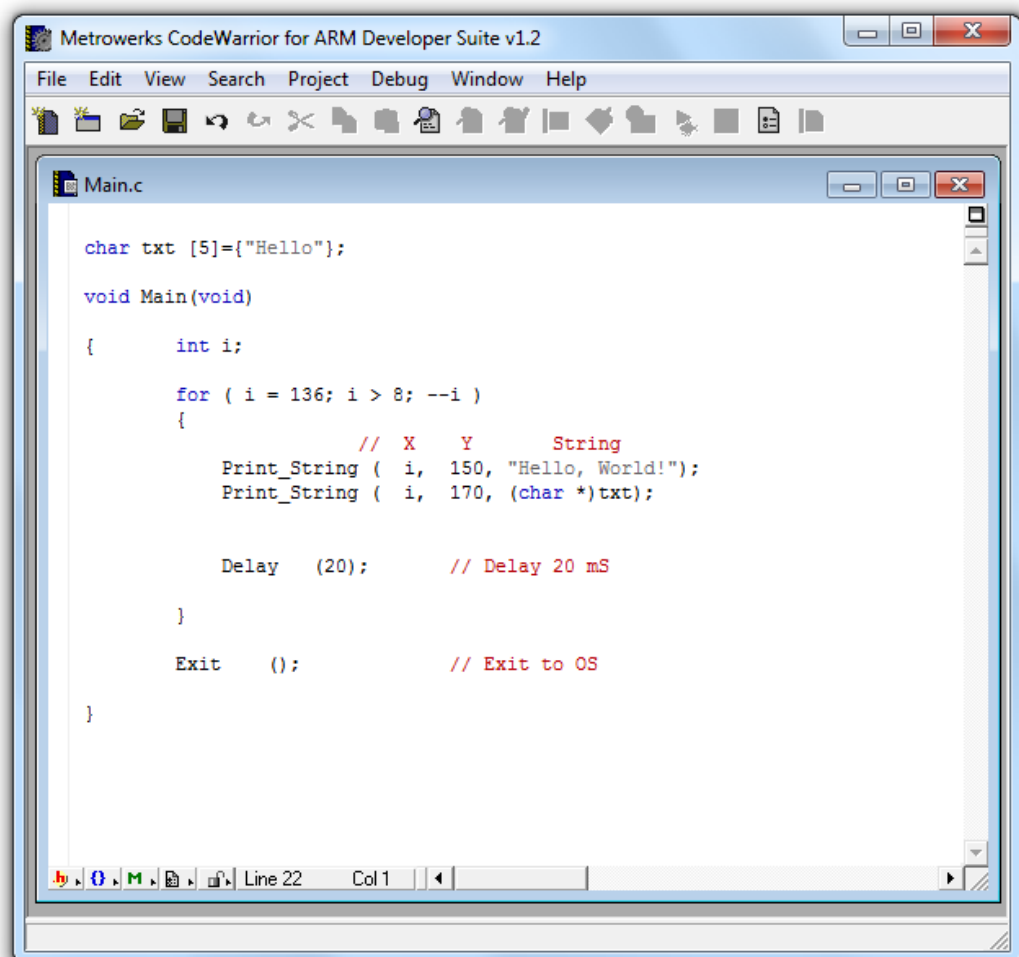
Size of executing binary module also depends from option "One ELF section per function". If it is checked, program size, for example, Hello1.bin decreases from 776 to 180 bytes. More complex programs in this case may not work, then simply cancel (uncheck) the option.



You can also use standard init files from ADS, Keil or IAR.

For work with examples is necessary to do:

- Find and launch the file of project with .mcp extension, for example, Hello1.mcp.



Edit file Main.c on your consideration, taste and style. Also you can edit the names of the system calls in the file StartOS.c as you wish.

For build project Press [F7].

In folder C:\sketches\HELLO1\Hello1_Data\DebugRel_bin will appear file Hello1.bin.

Write Hello1.bin in root of SD card. Insert card in Friendly ARM, turn power on.

During 1 second will appear the system screen.

Press [>] in the left lower corner of the screen.

List of files on SD then will appear.

Tap Hello1.bin by Stylus.

The program must begin to work.

Next you can experiment with other programs.

In case of any questions, please contact author.

All educational projects were verified on operability. They can be used as base for building your programs. Simply Edit text of Main.c, translate and get ready programs.

Folder Pictures contains primitives for examples STEPMOTOR and HELLO2.

In folder BMP2C you will find programs for converting BMP files onto source text in C language. This utility is from DVD supplied with Friendly ARM.

The GLCD FontCreator and Evafont utilities let you make your own font.

The WaveEditor, AudioConverter, Converter WAV to C source are used for embedding Sounds in your Projects.

Getting started with the ADS1.2

Install the Metrowerks CodeWarrior for ARM Developer Suite v1.2

Copy the sketch which you plan to work with on C:\

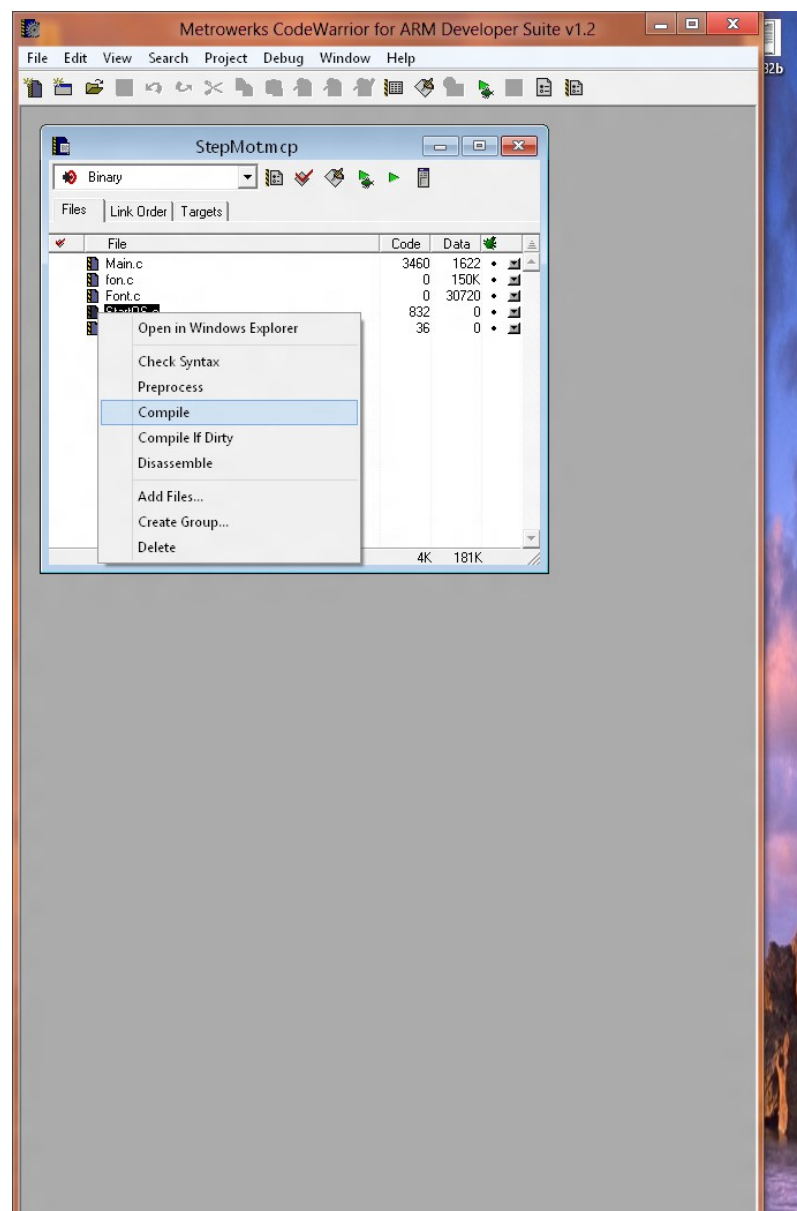
For instance, it will be C:\STEPMOTOR

Go to C:\STEPMOTOR\StepMot.mcp and launch it

You'll see Files, included in the project

First time files may be marked with [v] sign at the left side. It means that they were "touched" and are needed to be compiled.

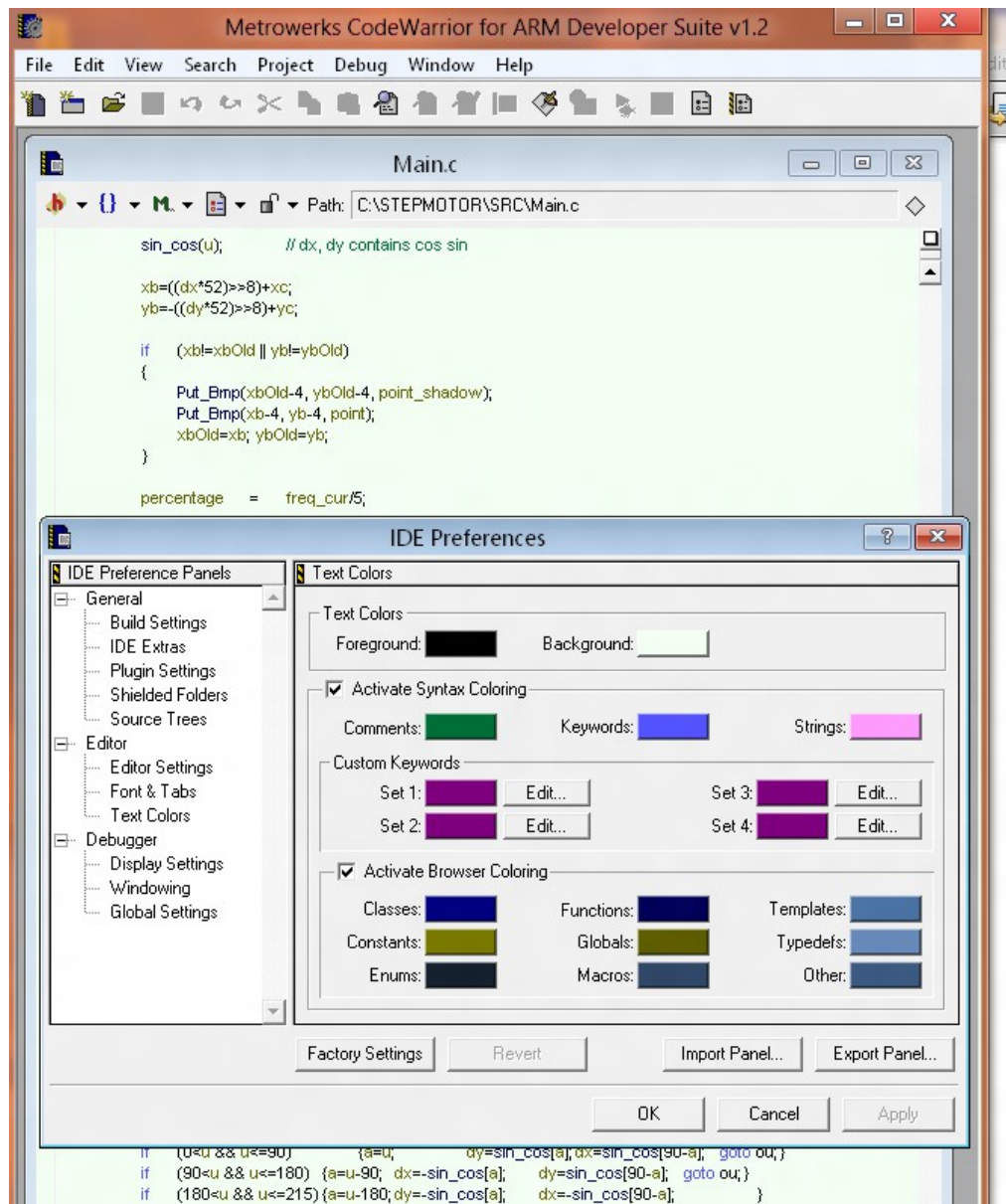
Right-Click on the filenames one by one and do perform "Compile"



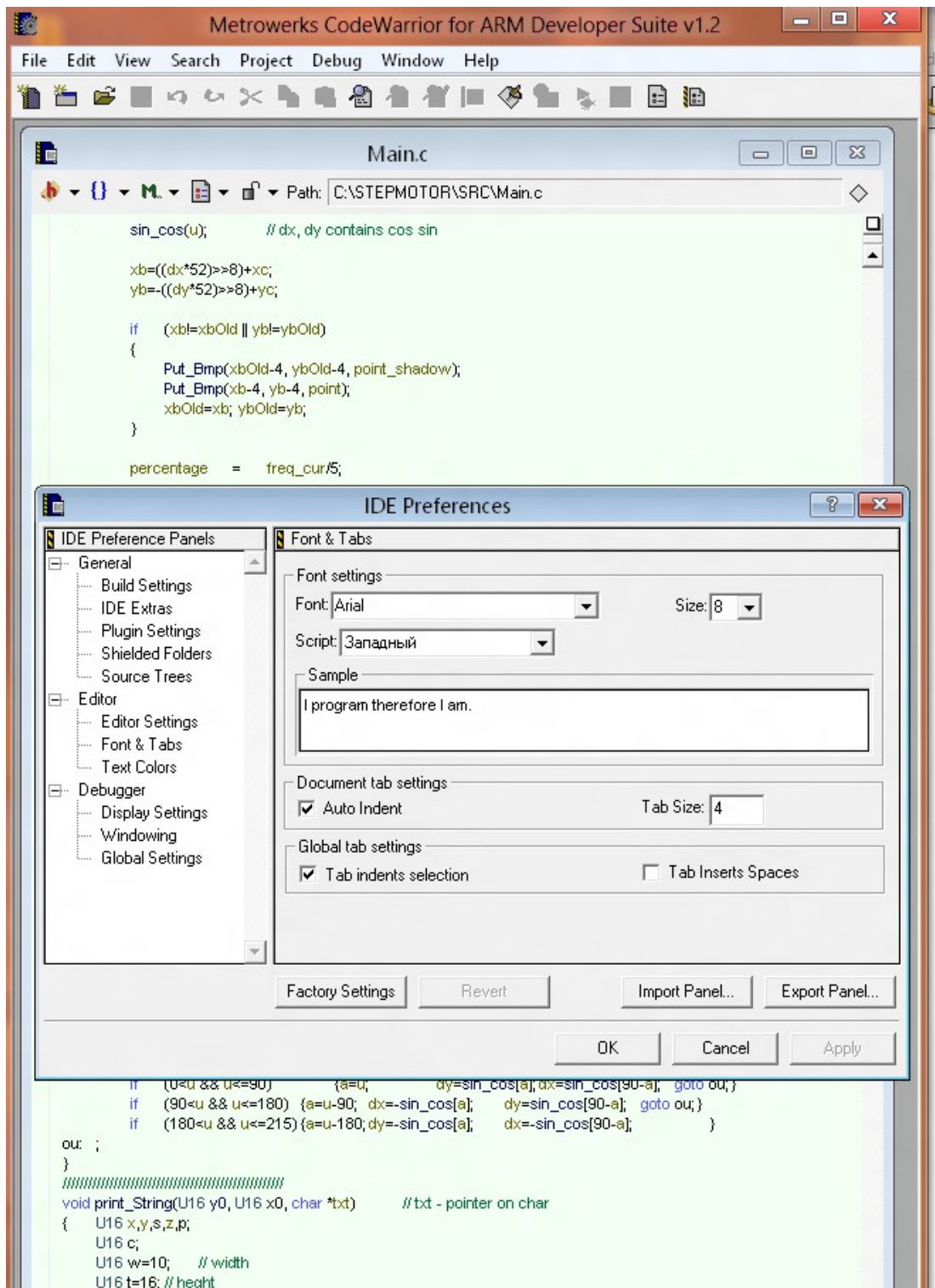
Next, double click with the Left mouse key on the file you want to edit. For example, main.c

Rotate your monitor in the Portrait position if it's possible.

Change the screen colors as you prefer.

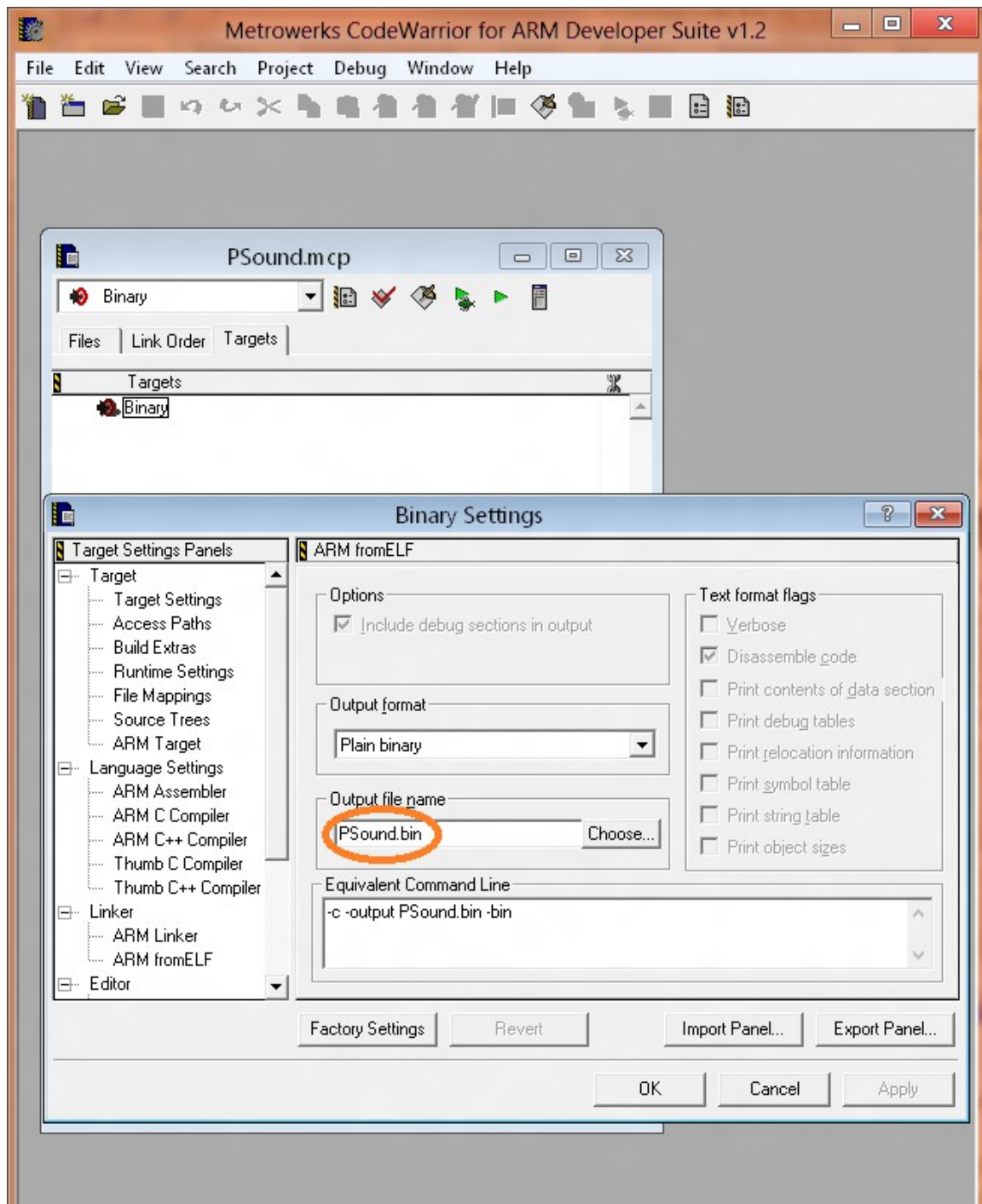


Select your favorite font (I use Arial size 8 because I think that fonts such as Courier have additional elements which can overload you visual sensors and the brain).



In the Control Panel->Screen-> ClearType try to unselect this option. The text on your screen must become sharp with no “wool” borders.

Change the name of Binary file as you like:



Last, press [F7] to compile your project

Metrowerks CodeWarrior for ARM Developer Suite v1.2

File Edit View Search Project Debug Window Help

Path: C:\STEPMOTOR\SRC\Main.c

```
int rr;
int dx, dy;

int yc=189;
int xc=101;
volatile int

int x, y, pen;
int percent;
int percent;
int xOld, yOld;
int xOld, yOld;

int running;
U32 freq = 0;
U32 freq_cur;

void Main(void)
{
    Buzz(80);

    Put_Bmp;
    Put_Bmp;
    Set_Fort;
    Print_String(20, 25, "A"); // total to OS area of screen
    Set_Fort(0, 0xFFFF, 25);

    ///////////////////////////////////////////////////
    u=215;
    v=1;
    freq = v*20; // 20...5000 Hz
    sin_cos(u); // dx, dy contains cos sin

    xt=((dx*34)>>8)+xc; // as sin*cos multiplied by 256 in table
    yt=((dy*34)>>8)+yc;
    Put_Bmp(xt-5, yt-5, ball);
    xOld=xt; yOld=yt;

    Again:
    pen_down=Read_TS(&x,&y); // Put X,Y addresses and Read_TS will fill them with X,Y
    if (pen_down)
    {
        Dec2Asc(x,t);
        Print_String(0,10,t);
        Dec2Asc(y,t);
        Print_String(72,10,t);

        dy=70-y; dx=200-x; // dx & dy of center of Start/Stop Knob
        rr=dx*dx+dy*dy;

        if (rr<=324) // 18*2
        {
            running=1;
            Buzzer_Out(1);
            print_String(64, 46, "Running ...");
            Put_Bmp(198, 139, run); // "Start"
        }

        /////////////////////////////////////////////////// Stop ///////////////////////////////////////////////////
        dy=45-y; dx=200-x; // dx & dy of center of Start/Stop Knob
        rr=dx*dx+dy*dy;
        if (rr<=324) // 18*2
        {
            Buzzer_Out(0);
            running=0;
            freq_cur=20;
            Put_Bmp(198, 139, stop); // "Stop"
            print_String(64, 46, "Stopped ...");
        }

        ///////////////////////////////////////////////////
        dy=y-c;
        dx=x-c;
        rr=dx*dx+dy*dy;
        if ( ((rr>900)&&(rr<1600)) // Borders of Ring 30<R<40
        {
            if (dx>=0&&dy>0) { u=90*dy/(dx+dy); } // I
            if (dx<0&&dy>0) { u=90*dy/(-dx+dy); u=180-u; } // II
            if (dx<0&&dy<0) { u=90*dy/(-dx-dy); u=180-u; } // III
            if (dx>0&&dy<0) { u=90*dy/(dx-dy); } // IV

            // u contains Angle from 215 to -35 getted from dx & dy
            if (u<=-35) u=-35; if (u>215) u=215;

            v=215-u; // Abs Volume (1...250)

            if (v==0) v=1;
            freq = v*20; // 20...5000 Hz

            sin_cos(u); // dx, dy contains cos sin

            xt=((dx*34)>>8)+xc; // as sin*cos multiplied by 256 in table
            yt=((dy*34)>>8)+yc;

            if (x!=xOld || y!=yOld)
            {
                Put_Bmp(xOld-5, yOld-5, ball_shadow);
                Put_Bmp(xt-5, yt-5, ball);
                xOld=xt; yOld=yt;
            }
        }

        if (freq_cur==freq) goto nothing;

        if (running)
        {
            if (freq_cur<freq) { freq_cur=freq_cur+20; }
        }
    }
}

Line 170 Col 1
```

Errors & Warnings

0 0 1 Errors and warnings fo...

Translation to Plain binary format successful.

Path:

S3C2440A.pdf - Adobe Reader

File View Window Help

413 / 602 75%

S3C2440A RISC MICROPROCESSOR

15 LCD CONTROLLER

OVERVIEW

The LCD controller in the S3C2440A consists of the logic for transferring located in system memory to an external LCD driver.

The LCD controller supports monochrome, 2-bit per pixel (4-level gray scale) mode on a monochrome LCD, using a time-based dithering algorithm and Fr it can be interfaced with a color LCD panel at 8-bit per pixel (256-level color) for interfacing with STN LCD.

It can support 1-bit per pixel, 2-bit per pixel, 4-bit per pixel, and 8-bit per pixel color LCD panel, and 16-bit per pixel and 24-bit per pixel for non-palletized tr

The LCD controller can be programmed to support different requirements on horizontal and vertical pixels, data line width for the data interface, interface t

FEATURES

STN LCD displays:

- Supports 3 types of LCD panels: 4-bit dual scan, 4-bit single scan, and 8
- Supports the monochrome, 4 gray levels, and 16 gray levels
- Supports 256 colors and 4096 colors for color STN LCD panel
- Supports multiple screen size
- Typical actual screen size: 640 x 480, 320 x 240, 160 x 160, and others
- Maximum virtual screen size is 4Mbytes.
- Maximum virtual screen size in 256 color mode: 4096 x 1024, 2048 x 204

TFT LCD displays:

- Supports 1, 2, 4 or 8-bpp (bit per pixel) palletized color displays for TFT
- Supports 16, 24-bpp non-palletized true-color displays for color TFT
- Supports maximum 16M color TFT at 24bit per pixel mode
- Supports multiple screen size
- Typical actual screen size: 640 x 480, 320 x 240, 160 x 160, and others
- Maximum virtual screen size is 4Mbytes.
- Maximum virtual screen size in 64K color mode: 2048 x 1024 and others

SAMSUNG

ELECTRONICS

Preliminary product information describes products that a for which full characterization data and associated errata i Specifications and information herein are subject to char

2004.03.15

LCD CONTROLLER

S:

COMMON FEATURES

The LCD controller has a dedicated DMA that supports to fetch the image data i memory. Its features also include:

- Dedicated interrupt functions (INT_FrSyn and INT_FICnt)
- The system memory is used as the display memory.
- Supports Multiple Virtual Display Screen (Supports Hardware Horizontal/Ve
- Programmable timing control for different display panels
- Supports little and big-endian byte ordering, as well as WinCE data formats
- Supports 2-type SEC TFT LCD panel
- (SAMSUNG 3.5" Portrait / 256K Color / Reflective and Transflective -a-Si TFT
- LT3S5001-PD1: TFT LCD panel with touch panel and front light unit (R
- LT3S5001-PD2: TFT LCD panel only
- LT3S5001-PE1: TFT LCD panel with touch panel and front light unit (T
- LT3S5001-PE2: TFT LCD panel only

NOTE

WinCE doesn't support the 12-bit packed data format.

Please check if WinCE can support the 12-bit color-mo

EXTERNAL INTERFACE SIGNAL

STN	TFT	SEC (LT3S500)
VFRAME (Frame sync. Signal)	VSYNC (Vertical sync. Signal)	ST
VLINE (Line sync pulse signal)	HSYNC (Horizontal sync. Signal)	CP
VCLK (Pixel clock signal)	VCLK (Pixel clock signal)	LCD_P
VD[23:0] (LCD pixel data output ports)	VD[23:0] (LCD pixel data output ports)	VD[2
VM (AC bias signal for LCD driver)	VDEN (Data enable signal)	TF
-	LEND (Line end signal)	ST
LCD_PWREN	LCD_PWREN	LCD_P1
-	-	LPC_
-	-	LPC_
-	-	LPC_F

In the Folder C:\STEPMOTOR\StepMot_Data\Binary you will find the binary file StepMot.bin.

Copy it onto an SD or MicroSD card, put in the FriendlyARM Mini210s board and use.

Note. All necessary options are already set for these example projects. For your new projects you can just copy one of them onto another folder named by your project's name. Rename the project file with .MCP extension, add/remove sources and you are done.

P.S. It must be programs for Linux (Ubuntu) letting you to work with the Windows applications.

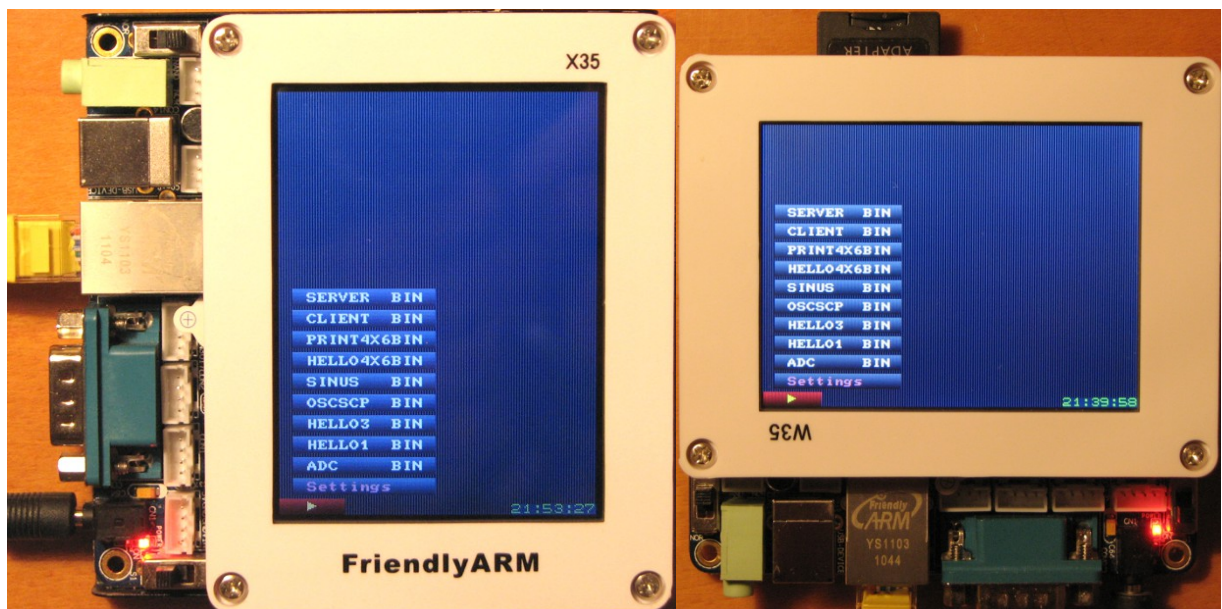
Networking (Not for Mini210s, but for Mini2440)

The STARTOS has embedded Web Server and Client functions.

The Server

Connect the FriendlyARM Mini210s board to PC or to the Router via the Ethernet cable.

Put the server.bin or the client.bin files onto the SD/MicroSD card. They can be found in Projects folders, Binary section.



Launching the server.bin you can see it window in your favorite browser:



The Page strings of the source file may contain the Data of sensors attached to the board. You can put there any items from Hardware/Software of your board.

```
U8 page1[] = "HTTP/1.0 200 OK\nContent-Type: text/html\n\n"
             "<meta http-equiv=\"\"refresh\"\" content=\"\"1\"\">"
             "<center><p><h1>The mini210s Web Server</h1></p> "
             "<hr><br> <h2><font color=\"blue\">your board is connected to the World!"
             "<br>now you can use the Internet of Things"
             "<br>read the board's digital&analog sensors";
```

Also, you can **put** parameters to the board from the Browser line:

<http://192.168.1.17/?parameter>

String with parameters is up to 10 Symbols (may be increased). Depending of the Parameter or its value you can control your board - turn something On/Off, playing sounds, etc.

Parameter (value) will be displayed on LCD after “?” sign.

Pressing the [Tap to play...] buttons in the Browser window will force your Mini210s playing sounds.

The source code

Board is initialized with addresses.

```
U8 macaddr[6] = {0xAA,0xAD,0xBE,0xEF,0xFE,0xED}; // Assign your MAC to your Board or left it
U8 Mini210s_ip[4] = {192,168,1,17}; // up to you (refer values)
```

```
res = NIC_Init ( macaddr ); // initializing the NIC
```

Main function includes Port number, source IP, destination IP, pointer to Page, pointer to String buffer returned:

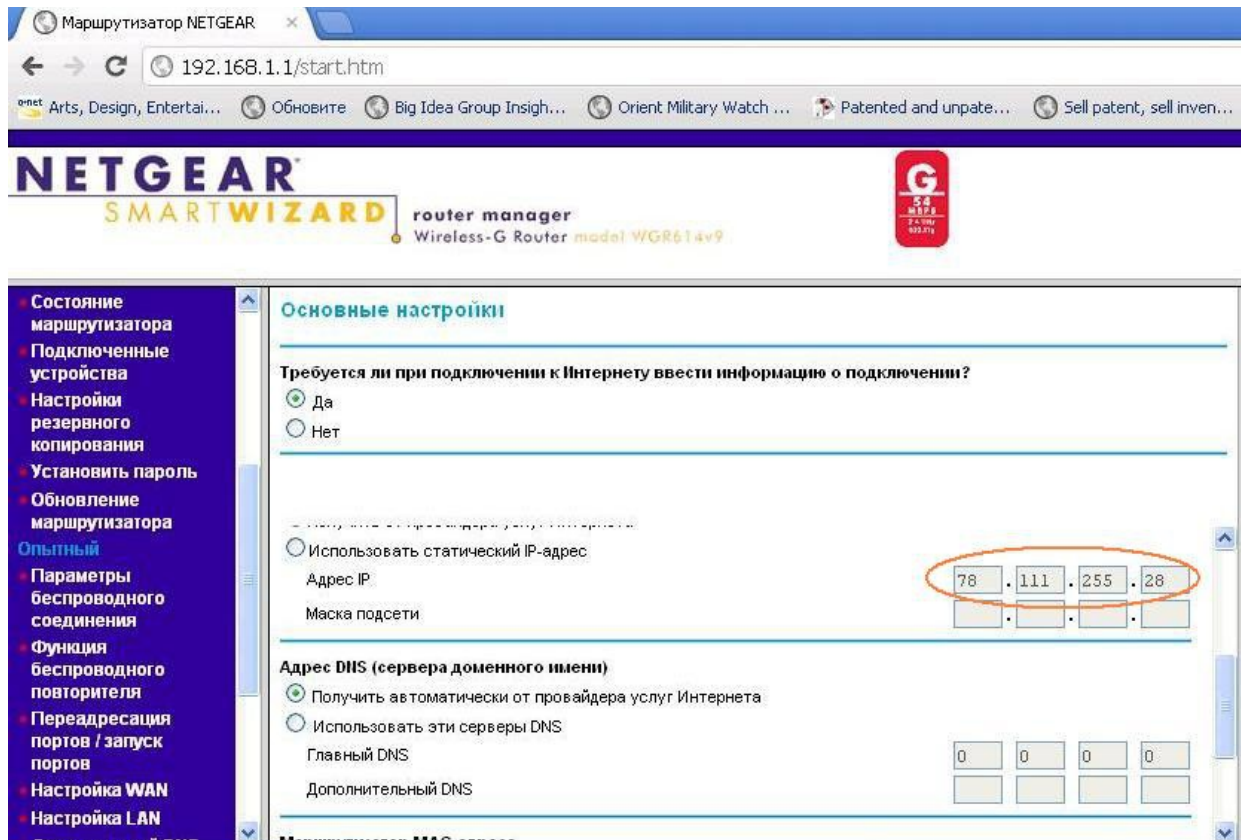
```
Server ( 80, Mini210s_ip, page1, strbuf ); // Mini210s_ip as the Server IP
```

The function returns Pointer to strbuf, so you can find parameters transmitted from Browser to Board for analyzing and performing some actions.

It's all for Mini210s-PC direct connection.

The internet service provider usually demands the Name and Password to log in. Thus, I use the Router. Additionally to doing login, it gives me the Home Network. All devices are connected by cable or wireless.

For outer access to your Board from any place try to find the IP address assigned to your Router. Then in the Browser window type this address.



Sorry for non English letters. In my Netgear router the access address is 192.168.1.1 You can see that my ISP assigned me an IP address 78.11.255.28.

An address 192.168.1.17 is used in the Sketch to initialize the FriendlyARM board.

Маршрутизатор NETGEAR

192.168.1.1/start.htm

NETGEAR SMARTWIZARD router manager Wireless-G Router model WGR614v9

Установка пароля
Обновление маршрутизатора
Опытный
Параметры беспроводного соединения
Функция беспроводного повторителя
Переадресация портов / запуск портов
Настройка WAN
Настройка LAN
Динамический DNS
Статические маршрутизаторы
Удаленное управление
UPnP
Интернет поддержка

Настройка TCP/IP LAN

Адрес IP: 192 . 168 . 1 . 1
Маска подсети: 255 . 255 . 255 . 0

☒ **Использовать маршрутизатор в качестве сервера DHCP**

Начальный IP-адрес: 192 . 168 . 1 . 2
Конечный IP-Адрес: 192 . 168 . 1 . 254

Резервирование адреса

	#	Адрес IP	Имя устройства	MAC-адрес
<input type="radio"/>	1	192.168.1.126	IPCAM	00:A8:F9:00:73:FF
<input type="radio"/>	2	192.168.1.3	IPHONE	64:B9:E8:38:3F:14
<input type="radio"/>	3	192.168.1.18	AP	00:1F:33:F6:7B:33
<input type="radio"/>	4	192.168.1.17	Arduino	AA:AD:BE:EF:FE:ED

Добавить Редактировать Удалить

Применить Отмена

To prevent attempts of dynamically addressing by the router, this address was reserved by me in the router for the Arduino or Mini210s boards. Also it remembers the MAC address for filtering aliens' devices.

And last, some Switches/Routers don't allow access from anybody. Then put an address 192.168.1.17 to enable outer connections. This will set the DMZ Server default address.



Finally, your board will be accessed from any place of the World by external address which was given by your ISP (in my case 78.111.255.28). The default port number is 80.



The Client

To begin this work if you are a novice in this field, please perform some steps.

First, for experiments, I recommend you to download the Apache Web Server from their Site (this procedure is quick and easy and it's free).



The screenshot shows the Apache HTTP Server Project website in a web browser. The browser's address bar displays "httpd.apache.org". The website features the Apache logo, a colorful feather, and the title "Apache HTTP SERVER PROJECT".

Essentials

- [About](#)
- [License](#)
- [FAQ](#)
- [Security Reports](#)

Download!

- [From a Mirror](#)

Documentation

- [Version 2.4](#)
- [Version 2.2](#)
- [Version 2.0](#)
- [Trunk \(dev\)](#)
- [Wiki](#)

Get Support

- [Support](#)

Get Involved

- [Mailing Lists](#)
- [Bug Reports](#)
- [Developer Info](#)

Subprojects

- [Docs](#)
- [Test](#)
- [Flood](#)
- [libapreq](#)
- [Modules](#)
- [mod_fcgid](#)
- [mod_ftp](#)

Miscellaneous

- [Contributors](#)
- [Sponsors](#)
- [Sponsorship](#)

The Number One HTTP Server On The Internet

The Apache HTTP Server Project is an effort to develop and maintain an open-source HTTP server for modern operating systems including UNIX and Windows NT. The goal of this project is to provide a secure, efficient and extensible server that provides HTTP services in sync with the current HTTP standards.

Apache httpd has been the most popular web server on the Internet since April 1996, and celebrated its 17th birthday as a project this February.

The Apache HTTP Server ("httpd") is a project of [The Apache Software Foundation](#).

Apache httpd 2.4.2 Released 2012-04-17

The Apache Software Foundation and the Apache HTTP Server Project are pleased to [announce](#) the release of version 2.4.2 of the Apache HTTP Server ("Apache"). This version of Apache is our 2nd GA release of the new generation 2.4.x branch of Apache HTTPD and represents fifteen years of innovation by the project, and is recommended over all previous releases. This version of Apache is principally a security and bug fix release.

This version of httpd is a major release of the 2.4 stable branch, and represents the best available version of Apache HTTP Server. [New features](#) include Loadable MPMs, major improvements to OSCP support, mod_lua, Dynamic Reverse Proxy configuration, Improved Authentication/Authorization, FastCGI Proxy, New Expression Parser, and a Small Object Caching API.

[Download](#) | [New Features in httpd 2.4](#) | [Complete ChangeLog for 2.4](#) | [ChangeLog for just 2.4.2](#)

Apache httpd 2.2.22 Released 2012-01-31

The Apache HTTP Server Project is proud to [announce](#) the release of version 2.2.22 of the Apache HTTP Server ("httpd"). This version is principally a security and bugfix release. There is an official [vulnerability list](#) of those issues fixed in this release.

This version of httpd is a major release of the 2.2 stable branch. [New features](#) include Smart Filtering, Improved Caching, AJP Proxy, Proxy Load Balancing, Graceful Shutdown support, Large File Support, the Event MPM, and refactored Authentication/Authorization.

[Download](#) | [New Features in httpd 2.2](#) | [ChangeLog for 2.2.22](#) | [Complete ChangeLog for 2.2](#)

Apache httpd 2.0.64 Released 2010-10-19

The Apache HTTP Server Project [announces](#) the legacy release of version 2.0.64 of the Apache HTTP Server ("httpd").

This version of httpd is principally a security and bugfix release.

For further details, see the [announcement](#).

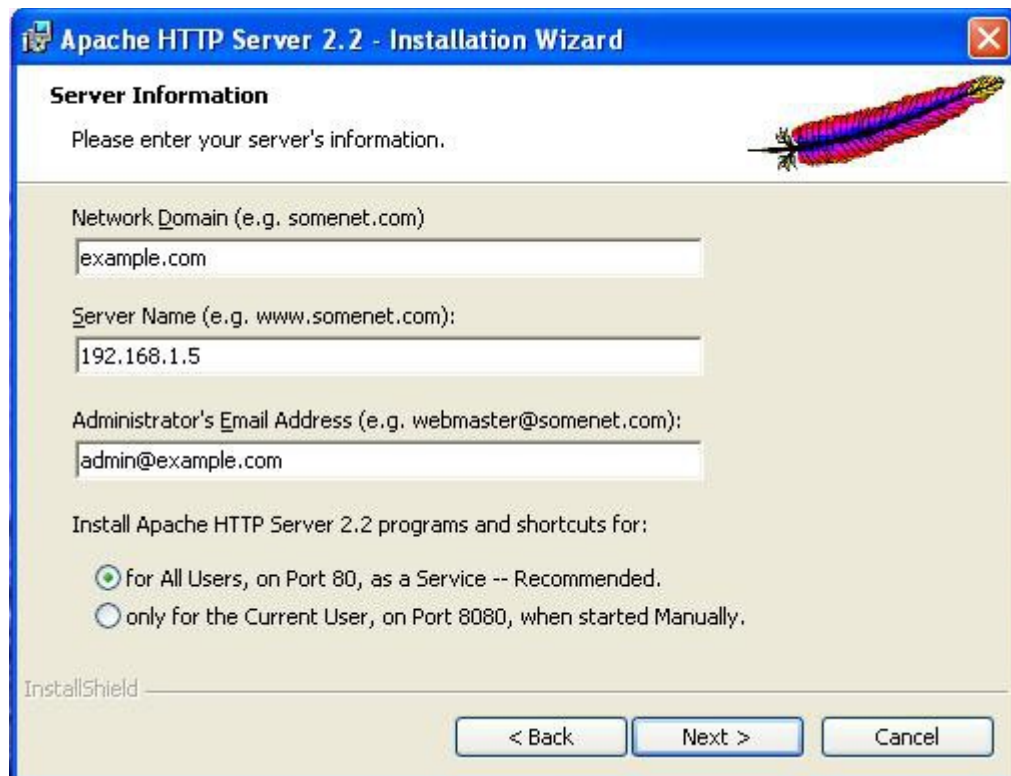
[Download](#) | [New Features in httpd 2.0](#) | [ChangeLog for 2.0.64](#) | [Complete ChangeLog for 2.0](#)

Want to try out the Apache HTTP Server?

Great! We have updated our [download page](#) in an effort to better utilize our mirrors. We hope that by making it easier to use our mirrors that we will be able to provide a better download experience.

Install it on your desktop PC for instance in C:\W folder.

Use the Server name as 192.168.1.5.



Apache HTTP Server 2.2 - Installation Wizard

Server Information

Please enter your server's information.

Network Domain (e.g. somenet.com)
example.com

Server Name (e.g. www.somenet.com):
192.168.1.5

Administrator's Email Address (e.g. webmaster@somenet.com):
admin@example.com

Install Apache HTTP Server 2.2 programs and shortcuts for:

☒ for All Users, on Port 80, as a Service -- Recommended.

☐ only for the Current User, on Port 8080, when started Manually.

InstallShield

< Back Next > Cancel

Configure your PC with an IP address 192.168.1.5 for example.

Sometimes other programs don't let the Apache server to start. You can quit them and try again. In my case I quitted the Skype. After launching Apache service I restarted Skype. On the next System booting nothing needed to do. Both of them work fine.

The file **C:\W\htdocs\index.html** is the file which will be accessed first (of course, you can get other files).

If things work properly, in your Browser window you'll see that:



The function:

```
res = Client ( Mini210s_ip, server_ip, request, buf ); // Mini210s_ip is the Client IP
```

```
U8 request[] =      "GET /index.html HTTP/1.0\n"  
                    "Host:192.168.1.5\n"  
                    "User-Agent: startos\n"  
                    "Accept: text/html\n"  
                    "Keep-Alive: 300\n"  
                    "Connection: keep-alive\n\n";
```

Will get the index.html file from your Server. (Or any other file as you wish).

Input parameters are: the source and destination IP addresses, string variable containing the Request to the Server and the buffer used for transferring data. When the Server will give the Page, the function return res (result) number –the number of bytes in the buffer with ready to use information.

Indeed, this function consists of several procedures. After passing one of them it changes its Status. Next pass it moves Status to the next state and so on, until the Server will send the page demanded.

Function don't loops and waits this data, every pass goes in milliseconds. After calling the function the User program can execute some activities.

start:

```
res = Client ( Mini210s_ip, server_ip, request, buf );
```

```

if      ( res==0 )      // Page is not ready

{
    //      perform some functions if needed
    //      .....
    //      .....

    goto  start; }

//      and here we have got the Page

// You can print it      Print_Page (0,0, buf+0x36 );    // Data begin at 0x36 offset in the buffer
// or process it

```

At the bottom of this document there is the source for just printing the page on the LCD.

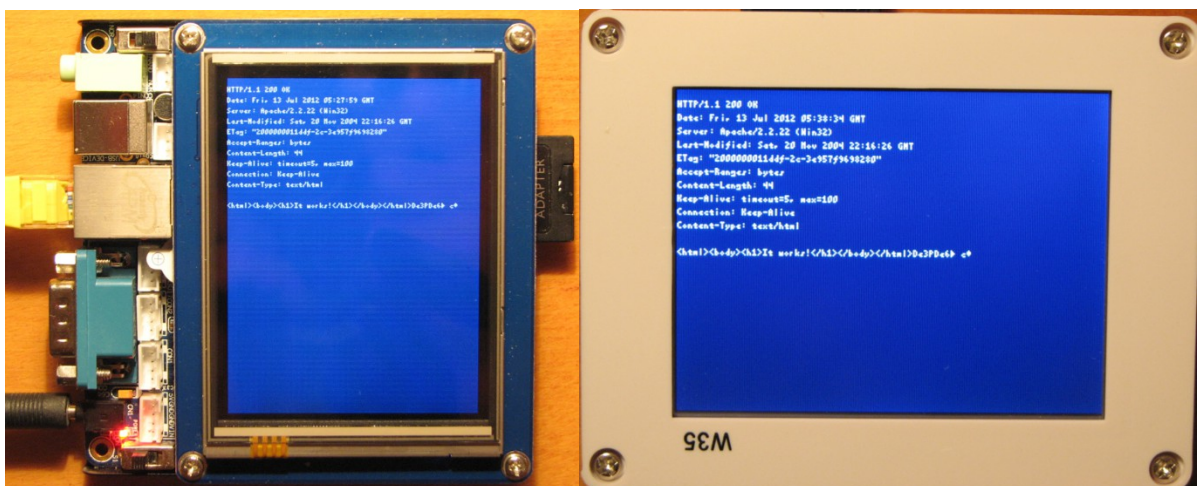
The Server will give you the Page in some seconds. All the time the User program can perform useful work.

You will see the following text on your LCD Screen:

```

HTTP/1.1 200 OK
Date: Tue, 10 Jul 2012 14:03:21 GMT
Server: Apache/2.2.22 (Win32)
Last-Modified: Sat, 20 Nov 2004 11:16:26 GMT
ETag: "c000000002021-2c-3e94ec1117680"
Accept-Ranges: bytes
Content-Length: 44
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html
<html><body><h1>It works!</h1></body></html>

```



This was getting from the Apache Server you have installed on a desktop PC.

Editing the file index.html you will get not only "It works!" but any other text in real time.

Using the Print_Page function with Font 4x6 you can get output onto Mini210s LCD or remove remarks and it will be outputted onto connected COM Port.

DNS are not supported by now, so you must find the Server's IP - you can use the **ping** command:

ping www.example.com and it will give you an IP address of the Site.

Getting the page, you can retrieve the data page contains (temperature, pressure, and so on.)

And you can make a simple PHP script on your own Internet Site. This program will collect all data for your needs and send them to your FriendlyARM board.

And last - this thing is interesting for playing with.

There is a simple SCADA video at: http://www.youtube.com/watch?v=Q2mjBy_Yor4

I began to think of making some toy project - the "Smart Doll's House".

It will control LEDs in the rooms, air fan, speak and show status on distant iPhone, Android phone or Notebook.

Note. DHCP not implemented so I connect my Mini210s to the router.

It's not a big problem – the ISP can assign you a permanent IP address. It cost ~\$5 per month in my location.

By now I can connect Mini210s to Netgear wireless AP and Netgear Router and plan to connect the Wi-Fi module.

On the pages below there is the source text.

```

//      WEB Client / startos

#define U32 unsigned int
#define U16 unsigned short
#define U8 unsigned char
#define BUFFER_SIZE 1518
////////////////////////////////////
U8 macaddr[6] = {0xAA,0xAD,0xBE,0xEF,0xFE,0xED}; // change it or left as is
U8 Mini210s_ip[4] = {192,168,1,17}; // IP address of your board - change according your Network
U8 server_ip[4] = {192,168,1,5}; // the Server IP address
U8 buf [BUFFER_SIZE+1]; // Data buffer
U8 request[] = "GET /index.html HTTP/1.0\n"
               "Host:192.168.1.5\n"
               "User-Agent: startos\n"
               "Accept: text/html\n"
               "Keep-Alive: 300\n"
               "Connection: keep-alive\n\n" ;

void Print_Page (int x0, int y0, U8 *txt);

////////////////////////////////////
void Main (void)
{
    int res,i; // the result

    res = NIC_Init (macaddr); // init NIC
    if ( res == 0 ) { Print_String(0,50,"No NIC/Link"); Buzz (8000,10); }
    else { Print_String ( 0, 30, "Connected at: " );
          if ( res&1 ) Print_String(112,30," 10M h/dup");
          if ( res&2 ) Print_String(112,30," 10M f/dup");
          if ( res&4 ) Print_String(112,30,"100M h/dup");
          if ( res&8 ) Print_String(112,30,"100M f/dup");
        }

start:
    res = Client ( Mini210s_ip, server_ip, request, buf ); // Mini210s_ip is the Client IP
    if ( res ) // We got proper Page
    {
        Print_Page (0,0, buf+0x36 ); // Data begins at 0x36 in the buffer
        // or you can send the Page to COM Port
        // Uart_SendByte ( '\n' );
        // for ( i=0x36; i < res; i++) Uart_SendByte ( buff[i] );
    }
    else goto start;

//      if ( !(*(int *)0x5800000c) & 0x8000 ) Exit(); // if pen is pressing the screen

    Delay (1000);
    goto start; // cycle repeats
}
////////////////////////////////////

//      the LCD width is small, so print the Page with a small Font 4x6

void Print_Page (int x0, int y0, U8 *txt) // txt - pointer on char
{
    extern char font4x6[];
    unsigned short *LCD_buffer;
    int LCD_X;
    int x,y,m,i;
    char t,s;
    y0=y0+6; // move to the foot of symbol

```

```

Fill_Screen(10);
// getting the Screen buffer parameters
LCD_buffer = (unsigned short *)((unsigned *)0x4d000014) << 1);
LCD_X      = (*(unsigned *)0x4d00001c);

for      ( i = 0; txt[i] != 0; i++)
{
    s = txt[i] & 0x7F;                                // get the Symbol code, valid code 0...127
    if      ( s == '\n' )                               // Carriage return (CR)
    {
        x0 = 0;
        goto do_nothing;
    }
    if      ( s == '\r' )                               // Line feed (LF)
    {
        if ( y0 < 230 ) y0 += 10;
        else y0 = 230;
        goto do_nothing;
    }
    m = y0 * LCD_X + x0;                                // starting point

    for(y=0; y<3; y++)
    {
        m += LCD_X << 1;                                // next Y
        t = font4x6[s*3+y];
        for(x=0; x<4; x++)
        {
            if      ( t & (0x80 >> x))                *(LCD_buffer+m) = 0xFFFF; // Draw Foreground (White)
            else                                           *(LCD_buffer+m) = 10;    // Draw Background (Dark Blue)
            if      ( t & (8 >> x))                      *(LCD_buffer+m+LCD_X) = 0xFFFF; // Draw Foreground
            else                                           *(LCD_buffer+m+LCD_X) = 10; // Draw Background (Dark b.)
            m++;                                           // next X
        }
        m = m - 4;                                        // restore X
    }
    x0 += 4;                                              // next Character place in the String
do_nothing: ;                                           // skip not drawing symbols CR&LF
}

////////////////////////////////////

```