Third Prize

Digital Oscillograph

Institution: **Department of Microwave Engineering Air Force Radar Academy**

Participants: **Hui Wu, Zhi-Xiong Deng, and Li-Hua Guo**

Instructor: **Yao-Jun Chen**

Design Introduction

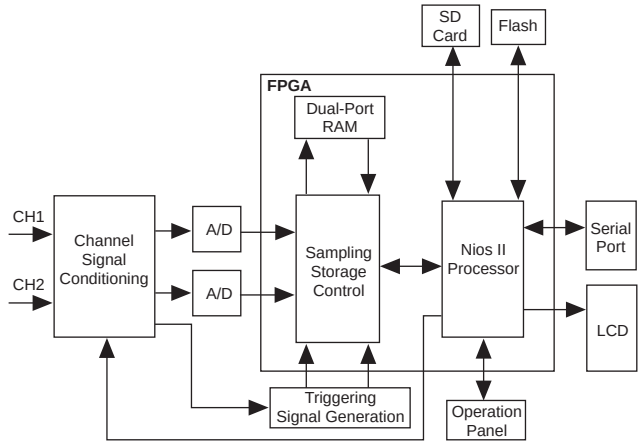
The digital storage oscillograph uses a microprocessor for control and data proccesing. It performs a variety of functions that simulation oscillographs cannot, such as leading triggering, combined triggering, burr capture, wave processing, hard-copy export, soft-disk recording, and long-term waveform storage. Typically, the digital storage oscillograph's bandwidth exceeds 1 GHz and the performance is higher than simulation oscillographs in many aspects. With high performance and a large application scope, developing a good digital storage oscillograph is important.

We implemented a digital oscillograph design using the Nios® II processor and an FPGA (a combination of software and hardware), operating on the Development and Education (DE2) board. The system has three parts:

- *External expanded circuit*—Converts from simulation signals to digital signals.
- *On-chip programmable part*—Controls the signal sampling storage, measures the frequency, and generates a self-checking signal.
- *Embedded processor*—Provides system and interface control.

Figure 1 shows the overall system structure. The input signal voltage is adjusted based on the collection scope of the condition circuit sampling. An analog-to-digital (A/D) device converts the signal from analog to digital. The system, running on the FPGA, stores the signal into the board's dual-port RAM. The Nios II processor reads data from the RAM and sends it to the LCD to display the resulting waveform.

Figure 1. System Structure



Function Description

Our design has the following functionality:

- Implements high-speed data sampling. The sampling signals can be obtained by dividing the internal system's frequency. The frequency can be changed by adjusting the frequency division coefficient, which changes the display's horizontal sensitivity.
- Supports programmable attenuation and amplification of the input signal, allowing the system to change the display's vertical sensitivity.
- Saves waveform data in SDRAM or flash for the system to show at a later time.
- Implements hardware-level equal precision frequency measurement.
- Uses the fast Fourier transform (FFT) algorithm with a time domain extraction method base 2 to analyze the tested signals' frequency spectrum.
- Supports communication between the DE2 board's serial communication interface and a computer so that the data can be sent to the computer and displayed after it is processed.
- Writes data to a secure digital (SD) card using the SD card interface on the DE2 board, which protects the data and allows it to be displayed in other regions.
- Generates unformed or edited waveform signals using the direct digital synthesis (DDS) frequency synthesizer method. These signals can replace those generated by the signal source and checks whether the system is functioning if there is no signal source.

Performance Parameters

Our design has the following performance:

- *Display*—Dual-channel waveform display.
- *Bandwidth*—About 500 kHz.
- *Maximum real-time sampling frequency*—16 MHz.

- *Storage depth*—512 points.
- *Testing voltage scope*—0 to +15 V.
- *Testing frequency scope*—7 Hz to 500 KHz.
- *Vertical sensitivity*—The minimum vertical sensitivity is up to 10 mV/div and the vertical sensitivity scope is 10 mV/div to 2.5 V/div.
- *Horizontal sensitivity*—The minimum horizontal sensitivity is up to 1 μs/div and the horizontal sensitivity scope is 1 μs/div to 500 ms/div.
- *Available trigger methods*—Edge and level triggering.

Design Architecture

This section describes the hardware and software design architecture.

Hardware Design

The following sections provide a detailed description of the hardware design.

Conditioning Circuit

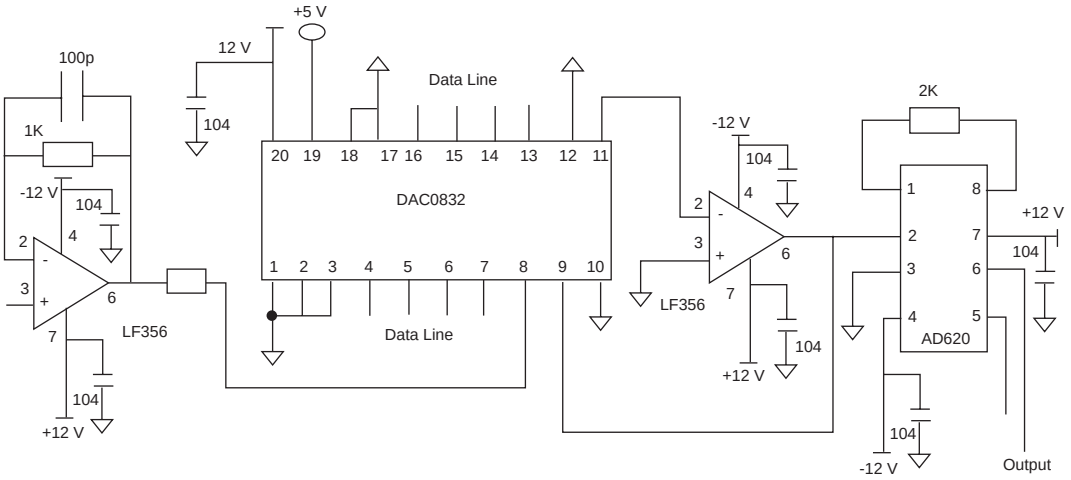
The conditioning circuit modulates the input signal to the scope, making it suitable for A/D conversion. The A/D reference is 2 V, and 8 grid lines on the oscilloscope equate to 2 V, or 0.25 V/div. Table 1 shows the amplification factor of the vertical scanning sensitivity per gear. The amplification factor refers to the percentage of the conditioning circuit's output voltage and input voltage.

Table 1. Relationship between Vertical Sensitivity and Amplification Factor

Sensitivity or Amplification	Relationship					
Vertical sensitivity mV/div	10	20	30	40	50	60
Amplification factor	25	12.5	8.33	6.25	5	4.17
Vertical sensitivity mV/div	80	100	150	200	300	400
Amplification factor	3.125	2.5	1.67	1.25	0.83	0.625
Vertical sensitivity mV/div	500	600	800	1,250	2,500	
Amplification factor	0.5	0.42	0.31	0.2	0.1	

As shown in Figure 2, the conditioning circuit is structured from attenuation to amplification.

Figure 2. Conditioning Circuit Diagram



The attenuation part consists of the 8-bit digital-to-analog (D/A) converter (DAC) DAC0832, and the D/A output voltage is:

$V_0 = V_{REF} \times D_{IN} / 256$

where V_{REF} is the input voltage and D_{IN} is the input from the Nios II processor. As D_{IN} varies, the system changes the attenuation factor, which affects the amplification factor of the conditioning circuit and the vertical sensitivity.

The amplification part consists of the meter amplifier, AD620. The amplification circuit's gain is:

$G = 1 + 49.4\ k\Omega / Rg = 25.7$

where $Rg = 2\ k\Omega$ is the fixed feedback resistor's value. The AD620 device's gain bandwidth product is 12 MHz and the gain bandwidth is 466.9 kHz; therefore, the system's testing signal frequency cannot exceed 466.9 kHz. If it exceeds this threshold, the AD620 device can still display the waveform and frequency, but it cannot display the correct amplitude value due to the signal amplitude attenuation after passing through the AD620 device. To solve this problem, we could substitute another high-speed component for the AD620 device.

Sampling Circuit

We used a 240 x 128 LCD as the external display. After it is divided into a grid using 10 horizontal and 8 vertical lines, the LCD has 16 x 16 pixel display sections.

The analog signal sampling circuit has an analog to digital (A/D) converter (ADC), the TLC5510 device. The maximum sampling frequency of this device is 20 MHz and it has a maximum real-time sampling frequency of 16 MHz. The TLC5510 is an 8-bit A/D chip with a conversion scope of 0x00 to 0xFF that corresponds to 128 points. We deal with the data from the TLC5510 device using a divide by 2, i.e., we pick the higher 7 bits from the TLC5510 device. Because each bit has 16 points, we can use the following formula to obtain the sampling frequency:

$fs = 16/T_{CELL}$

T_{CELL} demonstrates the time on the LCD's horizontal grid, as it relates to the time gap (horizontal sensitivity). The horizontal sensitivity is designed as 1 µs/div to about 500 ms/div, or 18 gears

altogether. According to the above formula, we can determine the sampling frequency. Table 2 shows a portion of the sampling frequency.

Table 2. Relationship between Horizontal Sensitivity and Sampling Frequency

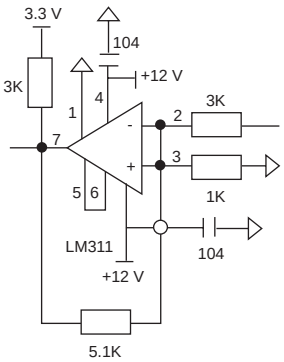
Sampling	Measurement						
T_{CELL}	1 µs	2 µs	5 µs	10 µs	20 µs	50 µs	100 µs
Sampling fs/kHz	16,000	8,000	3,200	1,600	800	320	160

A CPU provides the TLC5510 device's sampling signal, and the sampling frequency is obtained by the programmable frequency divider that divides the system's reference frequency. The divider has preset values that change the frequency division times, which converts the sampling signal to different frequencies. Various sampling signal frequencies lead to various times represented by each horizontal grid line, which in turn varies the time gap (horizontal sensitivity).

Waveform Shaping Circuit

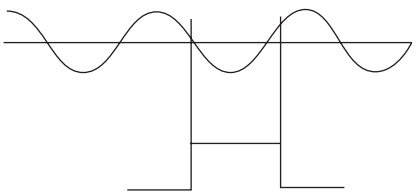
To generate the edge trigger's rising edge while controlling the sampling signal storage, the system must generate a stable rectangular wave. For this operation, the system uses a hysteresis loop comparer provided by the LM311 device. Figure 3 shows the waveform shaping circuit diagram.

Figure 3. Shaping Circuit Diagram



The inverting input terminal inputs the signal. When it is larger than zero, the output is at a low level; when the input passes through the zero point from high to low, the output immediately bounces to a high level. When the input passes through the zero point from low to high, the output remains at a high level because the LM311 device's inverting input terminal voltage is lower than that of the non-inverting input terminal. The output is not shifted to a low level until the inverting input terminal voltage is higher than the non-inverting input terminal. Thus, the output signal's rising edge remains on as soon as the input signal passes through the zero point from high to low. Figure 4 shows this process. The system uses this signal for storage control and edge triggering.

Figure 4. Hysteresis Loop Comparer



Storage Circuit

The sampling data is saved into dual-port RAM on the DE2 board. The address counter is synchronized by the sampling clock when the data is saved. The counter provides a write address for the dual-port RAM. When the address changes, the data is saved to the dual-port RAM in order. After 512 bytes of data are saved, the address counter returns to zero and stops counting. When the next trigger signal comes, it begins to count from the beginning and the data is saved from the beginning. With dual-port RAM, the system can read and write simultaneously. The Nios II processor generates the read address and has no connection to the address counter used by the writing address, which separates the read and write addresses and avoids errors.

Equal Precision Frequency Measurement

Traditionally, there are two methods for measurement:

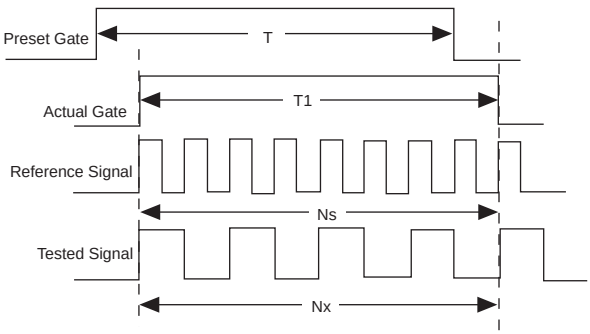
- *Frequency measurement*—This method records the tested signal's change cycles (or pulse numbers) N_X within the specified gate time T_W. The frequency of the tested signal is $f_X = N_X/T_W$.

- *Cycle measurement*—Assuming that the standard signal's frequency is fs, this method records the standard frequency's cycles N_S in a cycle T_X of the tested signal. The frequency of the tested signal is $f_X = f_S/N_S$.

The results of the two methods are different by ±1 word, and the testing accuracy is related to the counter's record N_X (number of pulses) or N_S (number of frequency cycles). To ensure testing accuracy, the low-frequency signal usually uses cycle measurement and the high-frequency signal uses frequency measurement. In contrast, our design uses the equal precision frequency measurement to facilitate the tests.

The gate time of the equal precision frequency measurement is not a fixed value. Instead, it is integer multiples of the tested signal cycles that are synchronous to the tested signal. In this way, the measurement eliminates the ±1 word error that results from counting the tested signal and achieves the equal precision measurement for the whole testing frequency band. See Figure 5.

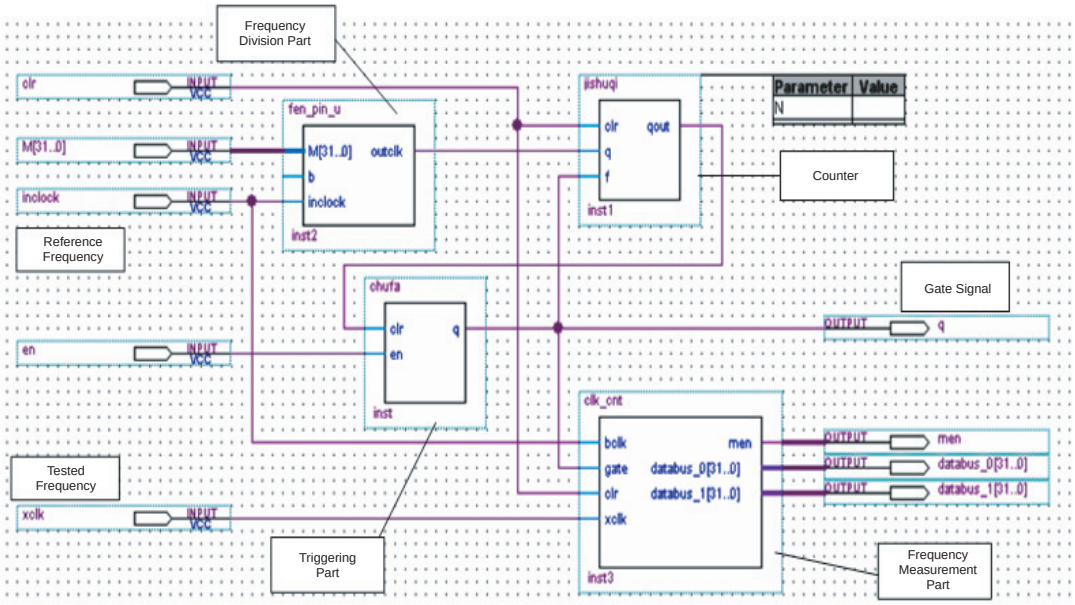
Figure 5. Equal Precision Frequency Measurement



During measurement, two counters work simultaneously to count the standard signal and the tested signal. When the gate starting signal is given (i.e., the rising edge of the preset gate), the counter does not count until the rising edge of the tested signal. When the preset gate closing signal (falling edge) arrives, the counter stops counting at the rising edge of the tested signal to complete the measurement. The actual gate time, T, is not strictly equal to the preset gate time T1; the difference is no more than a cycle of the tested signal. Provided in actual gate time T, the counter's record of the tested signal is N_X, the record of the standard signal is N_S, and the standard signal's frequency is fs. Thus, the tested signal's frequency, fs, is $f_S = N_X \times f_S/N_S$.

This formula shows that the N_X error is zero, because the gate time is integer multiples of the tested signal's cycle. That is, the comparative error of the frequency measurement has no connection with the frequency of the tested signal. Rather, it is only relevant to the gate time and standard signal's frequency. The longer the gate time, the higher the standard frequency and the smaller the comparative error of the frequency measurement. Figure 6 shows the block diagram of the equal precision frequency measurement.

Figure 6. Equal Precision Frequency Measurement Hardware Block Diagram



The block diagram has four modules:

- `fen_pin_u` divides the frequency of the reference clock `inclock`, and gets the square wave signal `outclock` with a 50% duty cycle (in which the `b` input stands for enable, low order, may not receive).

- `jishuqi` counts the `outclock` coming out of `fen_pin_u`. If the record is greater than or equal to the preset value `M[31..0]`, then `jishuqi` sends a signal to `chufa` while the counter stops counting and waits for the `clr` and `en` signals to begin the next counting cycle.

- The `chufa` module's `clr` signal receives the `qout` output generated by `jishuqi` to determine whether the `chufa` block's output `q` is high or low, generating the gate signal. The gate signal is used as the gate signal of `clk_cnt`.

- The `clk_cnt` block tests the frequency. The system delivers the `clr` signal to let the two 32-bit counters perform zero clearing. When the gate is high, the `BZ_ENA` and `DC_ENA` enables are 1s simultaneously (`BZ_ENA` controls `DC_ENA` to make `DC_ENA` consistent with it). Next, the block starts `BZ_Counter` and `DC_Counter` to make the system enter the count permission period. At this moment, `BZ_Counter` and `DC_Counter` independently count the tested signal and the standard frequency signal (`inclock` signal). After T_C seconds of the gate time width, the Nios II system sets the preset gate signal low. The two 32-bit counters are not shut off by the D trigger until the following tested signal's rising edge. At this time, the error caused by changing the gate time width T_C is, at most, a clock cycle of the reference clock `inclock` signal. Because `inclock`

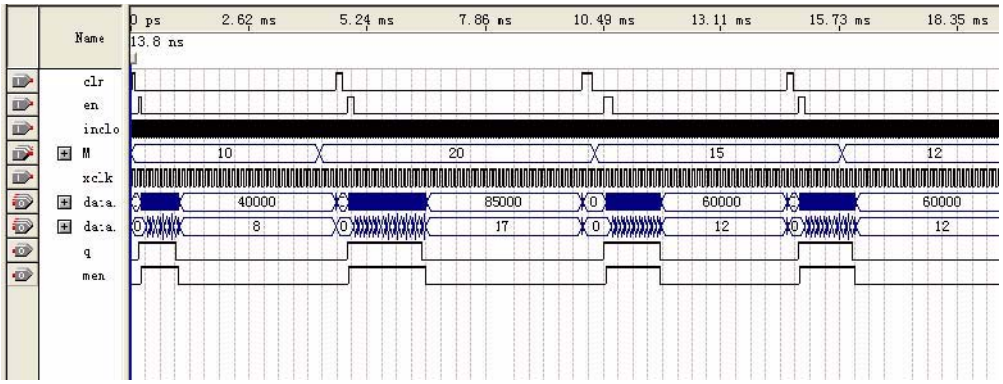
is sent by the highly stable 50-MHz crystal oscillator, the absolute measurement error at any moment is only 1/50000000 seconds, which is also the main error of the system.

Assuming that within a preset gate time T_C, the count value of the tested signal is N_X, the value of the standard frequency signal is N_B, and the frequency of the tested signal is X_{CLK}. According to the equal precision frequency measurement, the frequency of the tested signal is:

$X_{CLK} = N_B \times bclk/N_X$

Figure 7 shows the timing simulation.

Figure 7. Equal Precision Frequency Measurement Timing Simulation

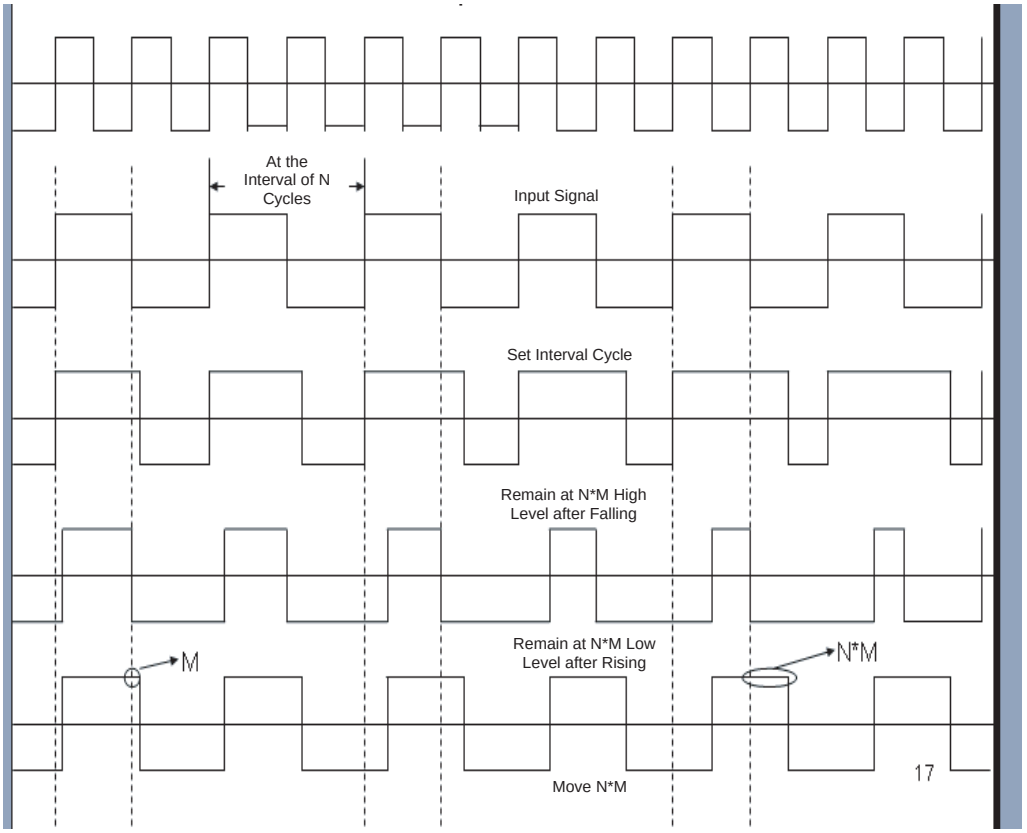


Equal Time Effectiveness Measurement

When the frequency of the tested signal is less than 200 kHz, the TLC5510 ADC sampling frequency can be up to 3.2 MHz at most. At this frequency, it does not severely interfere with the other components. However, if the sampling frequency is too high (more than 3.2 MHz), it causes very severe interference with the components and affects the waveform data's collection and display. Therefore, the system's testing frequency cannot be more than 200 kHz. To increase the system's tested frequency, we use the equal time effectiveness measurement.

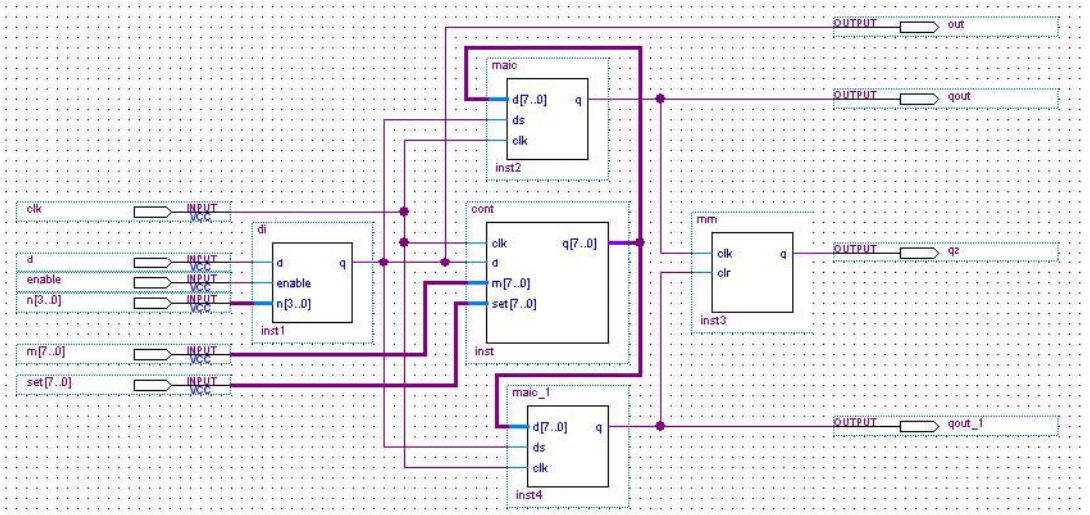
The equal time effectiveness measurement measures once every N cycles, and the time delay is TX/M, where TX is the cycle of the tested signal and M is the points collected in a cycle. Thus, when the tested signal's frequency is very high, the data can still be collected correctly. Figure 8 shows the schematic.

Figure 8. Equal Time Effectiveness Measurement Schematic



As shown in Figure 8, the input signal is first converted into a square wave signal. Later, a pulse signal is generated every N cycles, and these signals move backward by a small variable, which generates a low-frequency signal that is sent to the TLC5510 device as the sampling signal. The low frequency of this signal avoids the interference caused by a high sampling frequency. Figure 9 shows the hardware implementation.

Figure 9. Equal Time Effectiveness Measurement Block Diagram



The block diagram has five modules: `di`, `cont`, `maic`, `maic_1`, and `mm`. `d` is the tested signal, `enable` is the enabling signal, and `n[3..0]` is the cycle interval, which can be 4 or a number larger than 4 (which will be explained later in this paper). When `enable` is low (efficient), the counter inside the `di` module begins to count. When it counts to N, the output signal `q` outputs a pulse as wide as two cycles of the input signal `d`. The signal `q` is given to the `cont` module. In the `cont` module, `set[7..0]` are the points to collect, `clk` is the reference clock, and `m[7..0]` is the collection offset `m`. `m[7..0]` can be calculated as `m[7..0]` = TX x 50000000/`set[7..0]`.

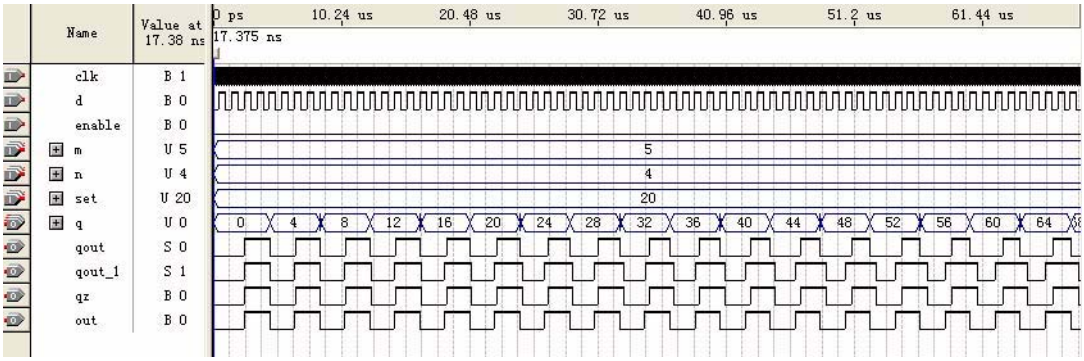
TX (the tested signal cycle) can be obtained as follows:

At the `cont` block's rising edge of `d`, the system initiates the internal counters `count` and `count1`. The counters begin counting, with `count` performing an add 1 operation, and `count1` adding m. That is, when a pulse arrives at `d`, `count` adds 1 and `count1` adds m until `count` reaches the collected points `set[7..0]`. After counting, `count` and `count1` are set to zero so they are ready for the next counting cycle. We make the `count1` value the output and send `q[7..0]` to `maic` and `maic_1`. `maic` postpones the rising edge of the `di` output `q`. The `cont` output, `q[7..0]`, references the pulse time with the falling edge remaining unchanged.

When a rising edge arrives at the `maic` input, `ds`, it initiates the internal counter and its output, `q`, is at low level. When the count reaches `q[7..0]`, the output is high and `ds` is high in the next cycle. The next work cycle begins after the rising edge of the next `ds`. `maic_1` operates like `maic`, except it postpones the falling edge of `ds` until the `q[7..0]` reference clock cycle. The rising edge remains unchanged and the operation is the same as `maic`, except the counter starts on the falling edge of `ds`. `mm` integrates the `maic` and `maic_1` signals. The `maic` output becomes the clock signal for `mm` and the `maic_1` output becomes the clear signal for `mm`. When `clr` is low, the output is zero; at the `clk` rising edge, if `clr` is high, the output is `clr`. According to the previously shown formula for `m[7..0]`, the final pulses of the `di_1` signal's output `q` signal are very narrow. To be detected by the system and while ensuring the hat the output signal's duty cycle is 50%, the `q` signal's pulse is widened by occupying two cycles of the tested signal. Therefore, N is a number equal to or bigger than 4.

Figure 10 shows the simulation waveform. `qz` moves away from `out` bit by bit, and each move has the same size as the previous move.

Figure 10. Equal Time Effectiveness Measurement Simulation Waveform



Edge and Level Triggering

If there is no trigger signal, the TLC5510 device samples the input signal. Because it can only sample 512 points at a time, it samples the signal again after 512 points. The TLC5510 sampling frequency is not an integral multiple of the input signal, which changes the sampling start position and makes the waveform display unstable. To avoid this problem, we need to synchronize the process by generating a trigger signal. Before the input signal reaches a set value, the trigger circuit generates a trigger signal, which tells the TLC5510 device to begin sampling. After sampling 512 points, TLC5510 stops sampling and waits for the next trigger signal.

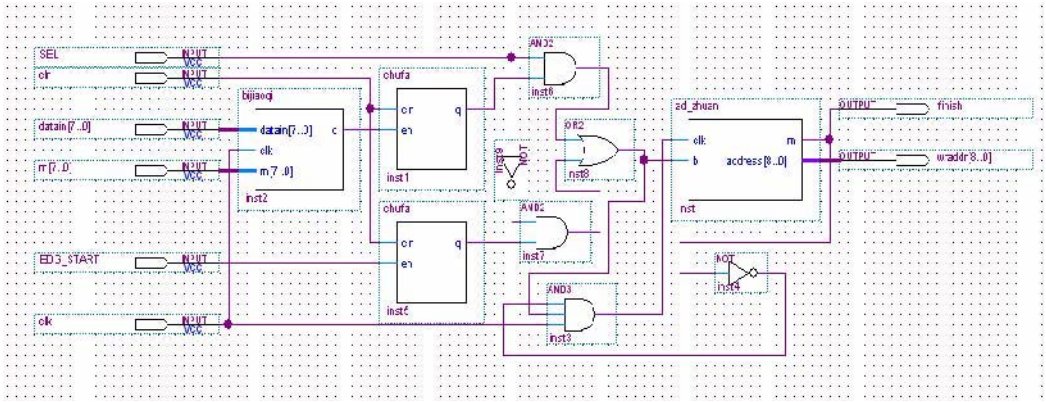
The design has two trigger modes:

- *Edge triggering*—The system uses a rectangular wave from the shaping circuit as the trigger signal. The positive edge is effective and stable when exceeding the zero point position. Therefore, it makes an effective trigger signal but cannot be controlled flexibly.

- *Level triggering*—With level triggering, the Nios II processor sets a level. If the input signal is higher than the set level, the system is triggered to store data. Because we can set the trigger level using `stsrem`, this process is more flexible than edge triggering.

Figure 11 shows the hardware diagram for the two triggering modes.

Figure 11. Level and Edge Triggering



This module has the following input signals.

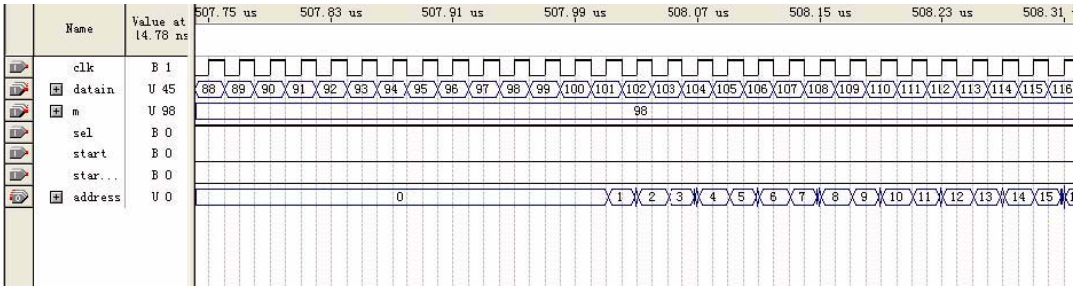
- SEL—Selects the trigger mode.
- start—Used for level triggering.
- EDG_START—Used for edge triggering mode.
- datain[7..0]—Provides the tested signal data obtained by sampling.
- m[7..0]—Provides the self-defined triggering level.
- clk—Clock.

When datain[7..0] is greater than or equal to m[7..0], the q output of the bijiaoqi block is high, otherwise it is low. Using a complementer, SEL is changed into the channel-1 signal, which is opposite the original signal. The module performs AND operations with the chufa q output signal above and below, and performs an OR on the result to perform selection.

The ad_zhuan counter stores the result. The Nios II system reads the result and displays it on the LCD according to the address. AND3 generates a clock that serves as the ad_zhuan module's system clock. When the OR2 output from the ad_zhuan block's b signal is low, m is low and the address[8..0] output is 0. When b is high and the addr counter reaches a set value, addr does not change and m is high. In the reverse process, m performs an AND operation with the clk signal and the OR2 output. The result is low, which stops ad_zhuan operation. If the counter does not reach the set value, m is 0 and the addr interval counter automatically adds a 1 when it receives a clock cycle.

Figure 12 shows the level and edge triggering emulation.

Figure 12. Level and Edge Triggering Emulation



The datain signal may exceed the defined 98 value because datain[1..0] in bijiaoqi is evaluated as 0 in two bits to avoid sampling data jittering and waveform reversion.

Control Panel

The system sets multiple keys. When the user presses a key, the Nios II processor performs the requested operation, such as changing the time gap (horizontal sensitivity), changing the voltage gap (vertical resolution), data storage, or playback. Because the keys on the DE2 board cannot meet our system's control requirements, we designed our system to use an additional 15 keys. See Figure 13 and Table 3.

Figure 13. Control Keys

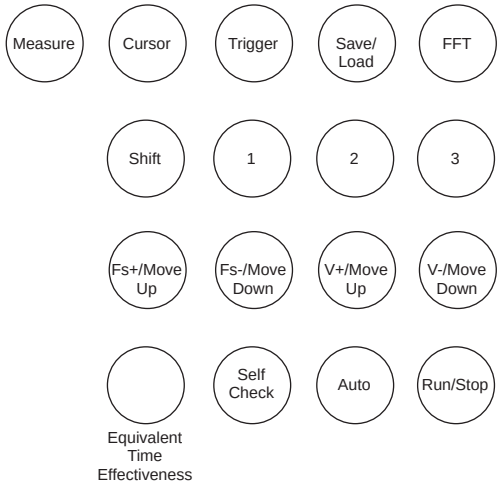


Table 3. Digital Oscillograph Control Keys

Key	Description
Measure	Displays the measurement menu.
Cursor	Displays the cursor.
Trigger	Displays the triggering mode menu.
Save/load	Displays the save and load menu.
FFT	Displays the frequency spectrum graphics menu.
Shift	Allows the user to select multiple functions.
fs+/move-up	Improves the sampling frequency or moves the displayed waveform up.
fs-/move-down	Lowers the sampling frequency or moves the displayed waveform down.
V+/move-up	Improves the vertical sensitivity or moves the displayed waveform left.
V-/move-down	Lowers vertical sensitivity or moves the displayed waveform right.
Run/stop	Run or stops, which determines whether to upgrade the control data.
Auto	Runs the system automatically.
Self check	Generates a self-check signal with DDS.
Equivalent time-effectiveness	Improves the system’s frequency scope operation.

The 1, 2, and 3 keys have a variety of functions, as shown in Table 4.

Table 4. 1, 2, and 3 Control Key Functions

Mode	Function
Measure	When Shift = 0, the 1, 2, and 3 keys provide channel selection (CH1 or CH2). When Shift = 1, the 1, 2, and 3 keys select the parameter type (f, T, etc.).
Cursor	1: Selects voltage or time increments. 2: When Shift = 0, the cursor moves up 1, when shift = 1, the cursor moves up 2. 3: When Shift = 0, the cursor moves down 1, when shift = 1, the cursor moves down 2.
Trigger	1: Signal source selection (CH1, CH2, or CH1 and CH2). 2: Selects continous or single triggering. 3: Selects edge or level triggering. When the 3 key is set for level triggering and Shift = 0, the 1 and 2 keys are the plus-minus for the triggering level.
Save/recall	1: Memory selection (flash01 through flash10 or SD card01 through SD card10). 2: Data storage. 3: Data playback.
FFT	Pressing the Shift key provides channel selection (CH1 or CH2). 1, 2: Unused. 3: Confirm or cancel.
Self-check	Pressing the Shift key selects the preset frequency (10 Hz, 20 Hz, 50 Hz, etc.). 1: Selects the preset phase (-180, -135, -90, etc.). 2: Selects between sine wave, square wave, triangular wave, and waveform editing. 3: Confirm or cancel.

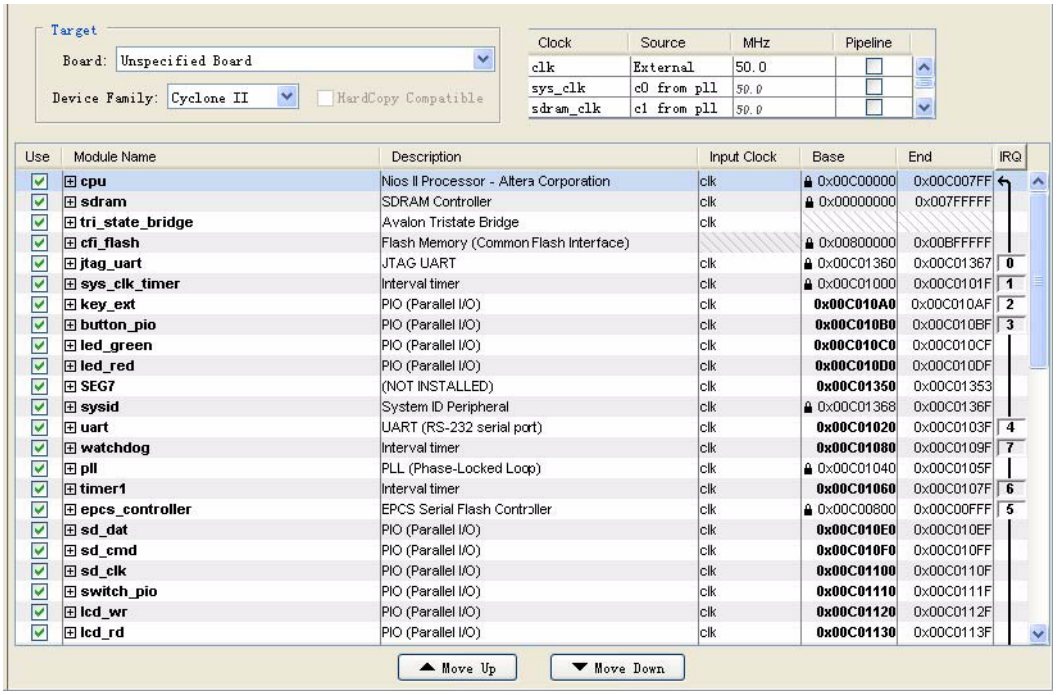
Software Design

This section describes the software design.

SOPC Resource Allocation

The Nios II processor provides excellent performance and made it easy for us to meet our design goals. Figure 14 shows our project in SOPC Builder.

Figure 14. SOPC Builder

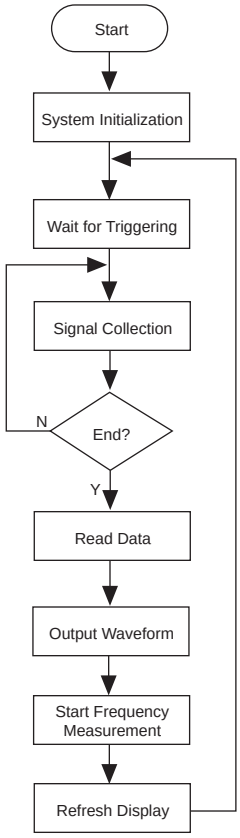


The CPU is a Nios II /f processor. The design has 4 Mbytes flash memory and 8 Mbytes SDRAM, which are connected to the Nios II processor via the Avalon bridge. We can run and debug the Nios II processor using a timer, JTAG_UART, and other modules. We added programmable I/O (PIO) peripherals (lcd_db, lcd_rd, lcd_wr, lcd_cs, and lcd_cd), which drive the external emulation LCD and control the display. A 160-MHz phase-locked loop (PLL) provides the equivalent time-effectiveness measurement's norm frequency.

System Design Flow

Figure 15 shows the system's software flow.

Figure 15. Software Flow



Interrupt System

The entire system uses interrupts. Figures 16 and 17 show the system flow for the serial port and key interrupts.

Figure 16. Serial Interrupt Flow

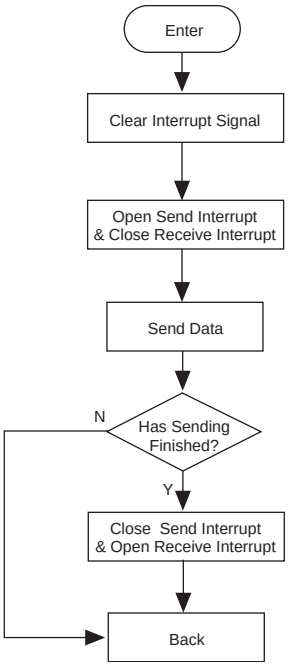
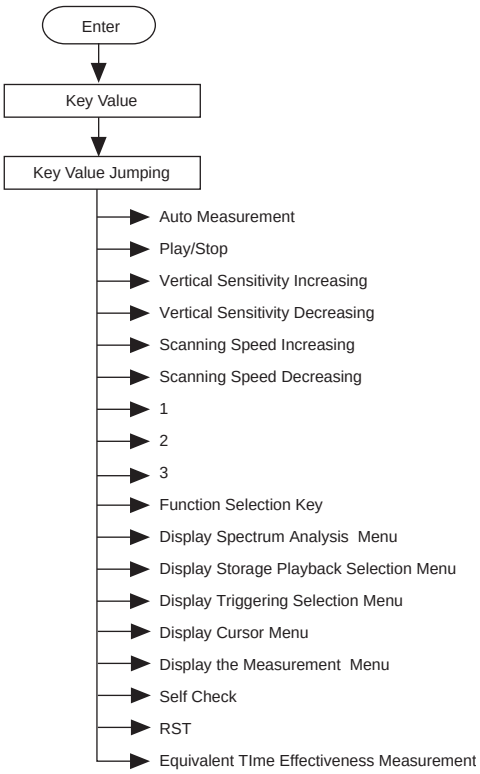


Figure 17. Key Interrupt Flow



Spectrum Analysis

The design's spectrum analysis uses the common time domain extraction, base 2 FFT. If the length of X(n) is n, divide X(n) into X1 and X2 according to the parity of n as shown in the following equations:

$X1(r) = X(2r)$ for r = 0, 1,…..N/2 - 1

$X2(r) = X(2r+1)$ for r = 0, 1,…..N/2 - 1

The sequence has 2 parts:

- Calculate the discrete Fourier transform (DFT) and add the result together.
- Obtain the DFT of the original sequence.

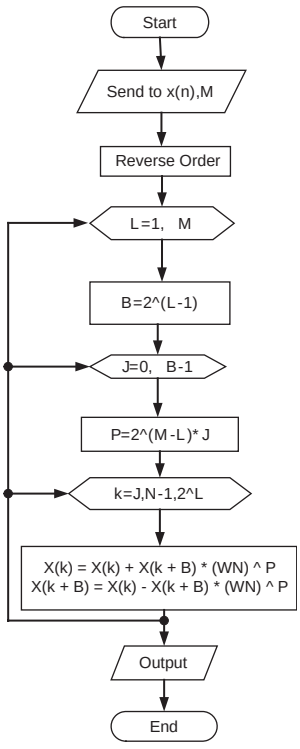
With this principle, divide the sequence by parity until it is the N/2 2-point DFT operation. This operation reduces the computing quantity. Additionally, due to the twiddle factor symmetry (see the following equations) during FFT programming, we use a circulation method to simplify the design.

$X(k) = X1(k) + W^{N}_{K} X2(k)$ for k = 0, 1,…,N/2

$X(k + N/2) = X1(k) -- W^{N}_{K} X2(k)$ for k = 0, 1,…,N/2

Because the time domain extraction method requires input data in reverse order, we have to reverse the data order before the FFT operation. To reverse it, we process the input data using a binary system and reverse the numbers in binary and then convert them into decimal to get the required input order. Figure 18 shows the FFT algorithm.

Figure 18. FFT Algorithm



Additional Functions

This section describes some additional functions our project contains, including serial port communication, DDS self-detection signals, and data storage, playback, and display.

Serial Port Communication

The DE2 board has an integrated RS-232 asynchronous serial communications port, which the design uses to establish the communication protocol, display data on the computer monitor, and obtain data. Our design only uses one computer. Therefore, establishing the serial port communication is simple: we just need to define the data delivery format and order according to the protocol. Our design uses a communication bit rate of 115,200 bits/second, no parity, 8 data bits, and 1 stop bit.

The sender sends data when the receiver sends it a signal. The sender only sends the data once, first using channel 1 and then channel 2. The channel data is in the following order: waveform data, signal frequency of the corresponding channel, cycle, peak-to-peak value, virtual value, and mean value. After channel data is sent, the sender sends a 0 as the final character. The data of the two channels cannot exceed 1,024 bytes.

Figure 19 shows the data send order. There are 8 data types (frequency, peak-to-peak value, virtual value, and mean value), each of which is 32 bits and occupies 32 bytes. The 0s occupy 8 bytes, so all of the data equals 40 bytes.

Figure 19. Data Send Order

0 471 472	Waveform data
	Frequency
	Cycle
	Peak-to-peak value
	Virtual value
	Mean value
	Maximum value
	Minimum value
511	Sampling points
512	0
  983 984	Waveform data
	Frequency
	Cycle
	Peak-to-peak value
	Virtual value
	Mean value
	Maximum value
1023	Minimum value
	Sampling points
	0

We wrote a Visual Basic (VB) program that displays the received data on the computer display. The program sends requests, receives communications data, and displays and adjusts the waveform.

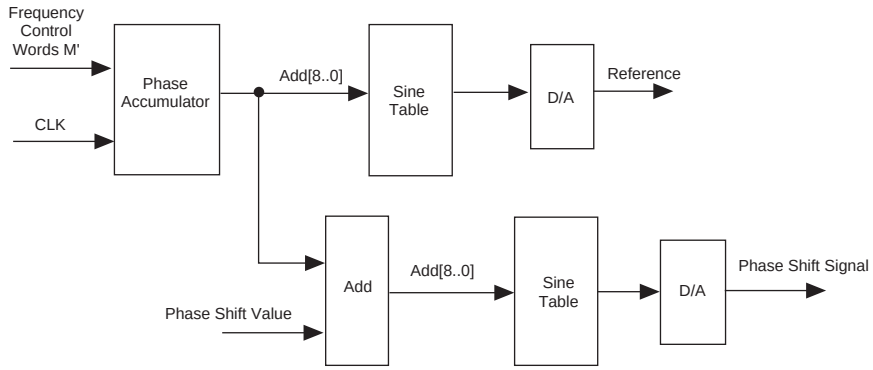
DDS-Generated Self-Detection Signal

To generate the self-detection signal, the system uses a signal generator that generates a sine wave, square wave, or triangular wave, and edits the waveform. The waveform's frequency, amplitude, and phase are adjustable. When building our design, we had two options for creating the self-detection signal:

- *Solution 1*—Use two special DDS devices (AD9850) to implement the signal generator. The AD9850 device can generate sine signals and alter the frequency and phase as long as the device is connected to an elaborate outside clock source. The AD9850 interface is easy to control, enabling the input of control data such as the frequency and phase from 8-bit parallel or serial ports. The output frequency is up to 125 MHz, and the circuit is easy and stable. However, this method only has 5-bit digital phase modulation, which varies the output signal phase between 180°, 90°, 45°, 22.5°, and 11.25° (or by a random combination of these values), which cannot achieve the accuracy of 0.1° that our design requires.

- *Solution 2*—Use an FPGA to implement the digital phase shift signal generator. The DDS adds the phase increment as required by the phase, takes the accumulation phase value as the address code to read the waveform data stored in the memory, and goes through D/A conversion and filtering to get the desired waveform. A cycle signal is divided into 512 shares and written to ROM as 512-point data. We can vary the phase and address increments using input to the Nios II processor to control the output frequency and phase. We can also generate other cycle signals using this method. Figure 20 shows a block diagram of this design.

Figure 20. Digital Phase Shift Signal Generation



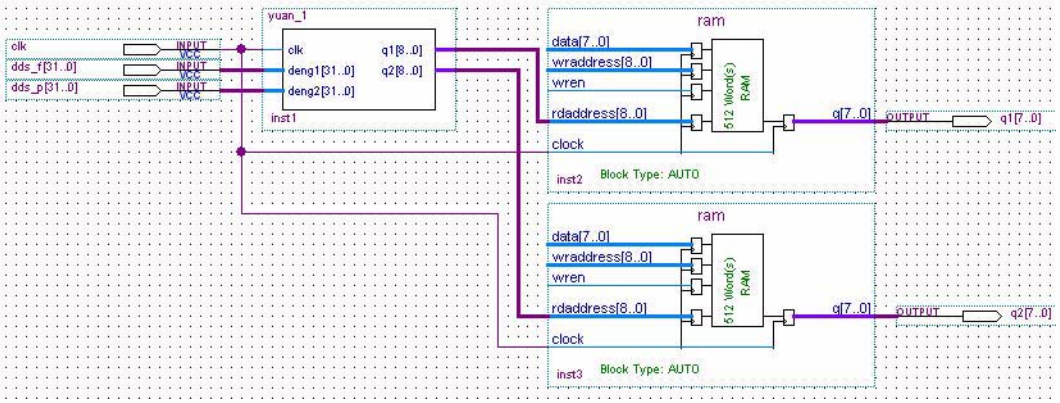
Using an FPGA to implement the digital phase shift signal source has the following advantages:

- The system can flexibly control the frequency and phase.

- Phase difference generation is very accurate and controllable.

Because of these advantages, we chose solution 2. Figure 21 shows the block diagram of this implementation for the case in which only one waveform is generated.

Figure 21. Signal Generator Hardware Block Diagram



As shown in Figure 21, `yuan_1` has three input signals:

- *dds_f[31..0]*—Presets the frequency, `Din`.
- *dds_p[31..0]*—Presets the phase.
- *clk*—Reference clock.

When the system starts, it first writes data into the two 512-bit dual-port RAM (`RAM1` and `RAM2`), which saves storage space and allows the waveform to be modified at any time to obtain the desired waveform. Next, the system presets the frequency and phase. At every clock cycle, the preset number adds to itself. The results are placed in the internal counter `count`, and the higher 9 bits of `count` are the `RAM1` read address. The system adds the phase preset number `dds_p[31..0]` to the higher 9 bits of `count` and uses the result as the `RAM2` read address, which offsets the waveform phase.

`count` has 32 bits. It automatically overflows when it is full because the RAM's data capacity is 512 bits. This design ensures that the data and address match. When count is full, it outputs a cycle of the required waveform signals. The signal is digital. The system can convert the signal from analog to digital using the ADI7123 device on the board and then output it through the VGA port.

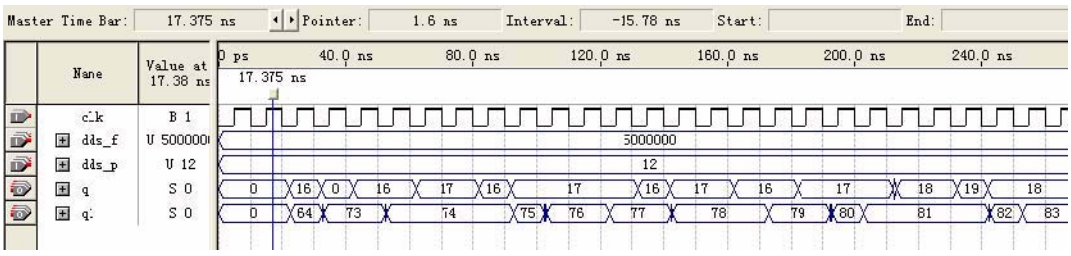
The relationship between the output signal frequency and the preset frequency, `Din`, is:

`fout` = `Din` x `fclk`/(2^N)

Changing the `Din` preset frequency changes the output signal frequency.

Figure 22 shows the hardware simulation that directly displays the waveform data.

Figure 22. Digital Frequency and Phase Modulation Function Simulation



Using the Nios II processor, the system generates varied waveform data and saves it to the dual-port RAM. The system can modify the generated waveform by setting keys that edit the DDS waveform parameter on the control panel. For example, by pressing different keys, the user can change the generated signal's waveform, frequency, peak-to-peak value, phase, and the harmonic component in the frequency spectrum. This action allows the user to display the waveform, measure parameters, and analyze the frequency spectrum.

Data Storage and Playback

The waveform can be stored in flash or SDRAM while it is displayed. Then, the user can use the playback function to re-display the data on the LCD.

Data Display in Other Regions

The portable SD card allows the data to be displayed in other regions. The user can save the data into the SD card using the control panel. When reading and writing to the SD card, the key problem is timing. The program must adhere to strict read/write timing to read and write data to/from the SD card. Figures 23 and 24 show the read/write timing schematics.

Figure 23. Read Timing

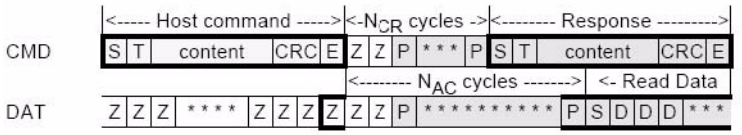


Figure 24. Write Timing

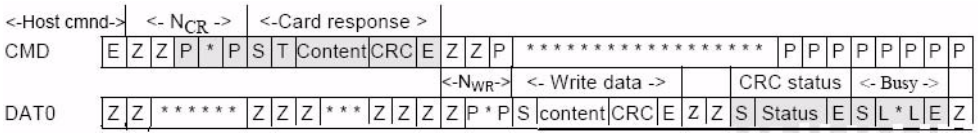


Table 5 describes the terms used in the figures.

Table 5. Timing Codes

Code	Description
S	Start bit (= 0)
T	Transmitter bit (Host = 1; Card = 0)
P	One-cycle pull-up (= 1)
E	End bit (= 1)
Z	High impedance state (-> = 1)
D	Data bits
X	Don't care data bits (from SD card)
*	Repetition
CRC	Cyclic redundancy code bits (7 bits)
	Card active
	Host active

System Test

We used a variety of tests to verify the system's operation to determine whether:

- The waveform is accurately displayed in LCD.
- There are glitches.
- There are errors in the frequency test and the degree of the error(s).
- The communication between the system and computer is successful.

To achieve the testing objectives, we used the following devices:

- TDS 210 mixed-signal oscilloscope
- YB 1605 function signal generator
- Computer
- Standard serial line
- SD card (1 Mbyte)
- Positive and negative 12-V power source

Waveform Test

For this test, we connected the system, let the signal source emit a signal, input the signal to the system, connected the oscilloscope with the signal source, and looked to see if the waveform displayed on the oscilloscope was as the same as that displayed on the system LCD. We used sine wave, triangle wave, and square wave signals as test input.

Our test result showed that the waveforms on the oscilloscope and LCD are almost the same and the waveform display functions correctly.

Frequency Test

For this test, the input signal's waveform and voltage remain unchanged while the frequency changes from 10 Hz to 500 kHz. We checked the frequency column on the LCD to see the result. We used two different signal routes to test the two channel frequencies. To avoid the impact of an unstable signal source on the signal, the design uses the TDS210 oscilloscope's value as the standard.

Table 6 shows the channel 1 testing result.

Table 6. Channel 1 Frequency Test Result

Signal Frequency (Hz)	8	100	1.100 k	9.99 k	99.96 k	300.27 k	500.25 k
Testing value (Hz)	7	99	1.099 k	9.979 k	99.94 k	300.2 k	495.9 k
TDS210 measurement	7.072	99.11	1.100 k	9.980 k	100.0 k	300.8 k	501.3 k
Error	1.02%	0.11%	0.09%	0.01%	0.06%	0.2%	1.08%

Table 7 shows the channel 2 testing result.

Table 7. Channel 2 Frequency Test Result

Signal Frequency (Hz)	8	100	1.100 k	9.99 k	99.95 k	300.24 k	500.22 k
Testing value	7	99	1.098 k	9.975 k	99.96	300.2 k	500.1 k
TDS210 Measurement	7.062	99.21	1.099 k	9.960 k	100.5 k	299.9 k	499.2 k
Error	0.88%	0.21%	0.09%	0.15%	0.54%	0.1%	0.18%

Voltage Test

In this test, we left the input signal frequency unchanged while changing its peak-to-peak value. We recorded the test value obtained in the system and calculated the error. Like the frequency test, we use two signal routes to test the two channels while taking the TDS210's value as the standard. The frequencies of the two input signal routes are 10 KHz.

Table 8 shows the channel 1 testing result.

Table 8. Channel 1 Voltage Test Result

Signal Peak-to-Peak Value	0.2 V	1.4 V	2.5 V	5.0 V	7.0 V	10.0 V	15.0 V
Test value	0.221 V	1.38 V	2.54 V	4.98 V	7.23 V	10.5 V	16.0 V
TDS210 measurement	0.215 V	1.33 V	2.48 V	4.96 V	6.84 V	10.1 V	15.8 V
Error	2.71%	3.76%	2.42%	0.40%	5.70%	3.96%	1.27%

Table 9 shows the channel 2 testing result.

Table 9. Channel 2 Voltage Test Result

Signal Peak-to-Peak Value	0.2 V	1.4 V	2.5 V	5.0 V	7.0 V	10.0 V	15.0 V
Testing value	0.211 V	1.34 V	2.46 V	4.87 V	7.0 V	10.2 V	15.8 V
TDS210 measurement	0.218 V	1.32 V	2.44 V	4.96 V	6.96 V	10.0 V	15.8 V
Error	3.21%	1.51%	0.82%	1.81%	0.57%	2%	0

Serial Communication Test

In this test, we connected the asynchronous serial communication interfaces of the system and computer with a serial cable. Then we started the program and watched the waveform and data displayed on the computer screen.

According to the test, the design correctly displays the waveform while changing the system's input signal. If the receiver (computer) sends an inquiry, the waveform displayed on the computer screen is also changed. The display correctly displays data such as the testing signal's frequency.

DDS Self-Detection

According to this test, the self-detection waveform is generated correctly.

Test Summary

Our tests proved that the system can perform all functions specified by the system design. The equal precision frequency measurement and equal time-effectiveness sampling methods improve testing accuracy and efficiency. Additionally, the system has little delay and error, and can meet general demands.

Conclusion

The four-month project increased our understanding of FPGAs, from the theory to hands-on experience. During the project, we were impressed by the powerful Quartus® II software; specifically, the many commonly used intellectual property (IP) cores that are available in SOPC Builder. We could modify them to fit our design needs, which made the design process easy and convenient. Additionally, the system permitted us to add our own custom IP cores, allowing us to meet customer demands while making the design flexible. The Quartus II software provides all of the functions that are required for the design process from beginning to end. The graphics and user interface helped us use the software more easily. The Quartus II software contains the Nios II Integrated Development Environment (IDE), so we could use that graphic interface for software programming, compiling, and design modification. We could even use the interface for system downloading and operation.

By participating in this competition, we obtained new knowledge, applied our existing knowledge, and mastered the skills required to create systematic, orderly designs. The competition also helped us develop our overall viewpoint. For example, when discovering a design fault, we learned not to just modify the code, but to consider the overall situation.

We also learned the importance of teamwork. As the saying goes, “let the expert deal with his specialty.” Each member of the design team should work on the area that he/she is best at, ensuring that the design is completed in time and that problems are found and solved as soon as possible.

Finally, we want to thank our instructors and all the people who supported us during the design process.

References

Collection of Work of the Fifth National Undergraduate Electronic Design Contest Winners (2001). XiDian University Press, 2006.

Hongwei, Li and Yuan Sihua. *Quartus II-based FPGA/CPLD Design*. Electronic Industry Press, 2006.

Yumei, Ding and Gao Xiquan. *Digital Signal Processing* (2nd edition). XiDian University Press, 2004.

Hubei Undergraduate Electronic Design Contest. 2006.

Wang Dong. *Visual Basic Program Design* (2nd edition). Tsinghua University Press, 2002.

Haoqiang, Tan, Zhang Jiwen, and Tang Yongyan. *A discourse of C Programming*. Higher Education Press, 2003.

Xie Jiakui. *Electronic Circuit (Linearity)*. Higher Education Press, 2003.

Shujun, Guo, Wang Yuhua, and Ge Renqiu. *Principle and Application of Embedded Processor-Nios II System Design and C Programming*. Tsinghua University Press, 2004.

SD Group. *SD Memory Card Specifications*. 2000.

Cheng, Wang, Wu Jihua, and Fan Lizhen. *Altera FPGA/CPLD Design*. 2005.

Screen Shots

Figures 25 through 31 provide screen shots of our system.

Figure 25. Waveform Data Storage and Invoking Interface

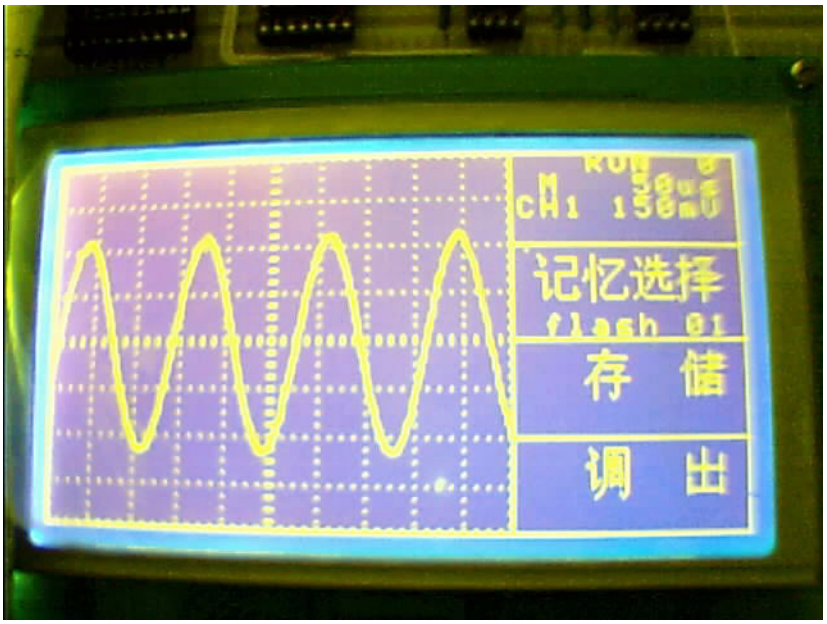


Figure 26. Trigger Selection Interface

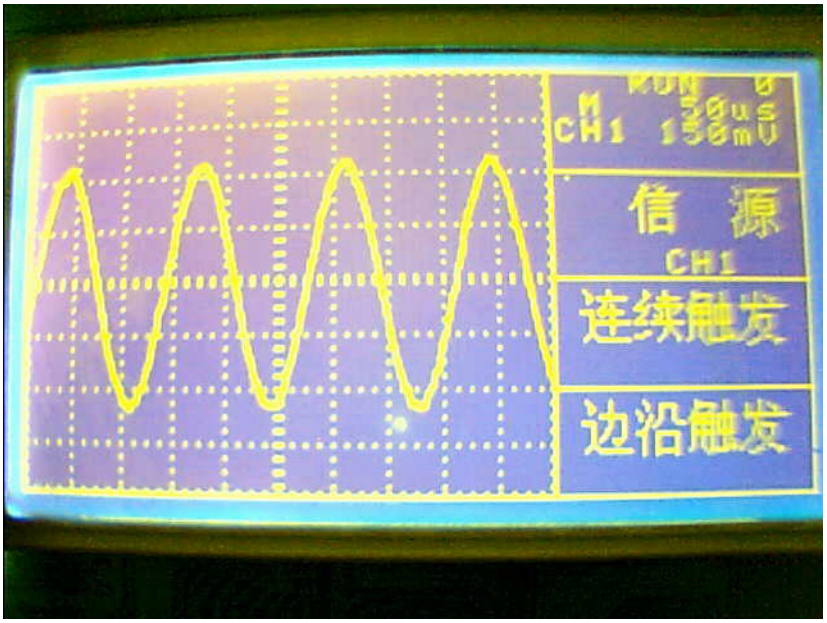


Figure 27. Square Wave Spectrum Analysis

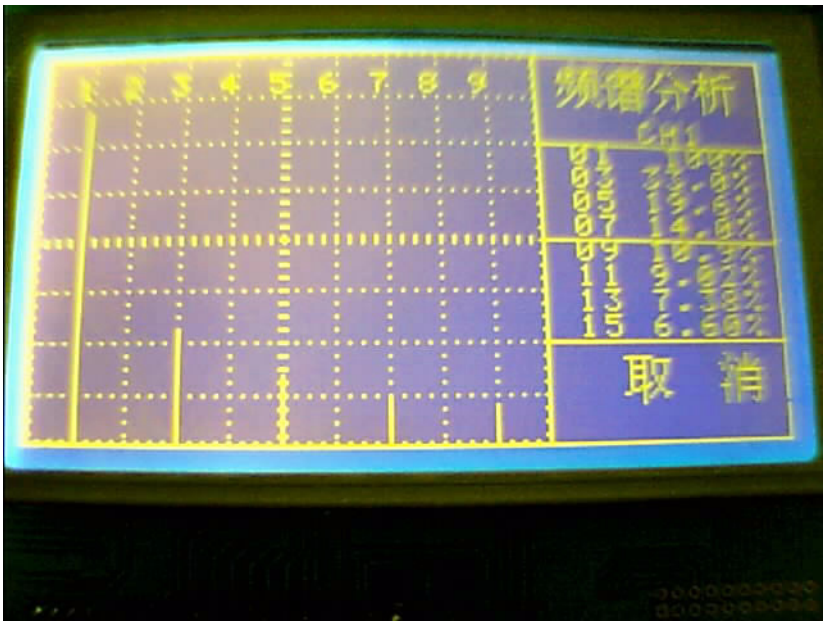


Figure 28. Waveform Editing

Figure 29. Voltage Measurement Between Any Two Points

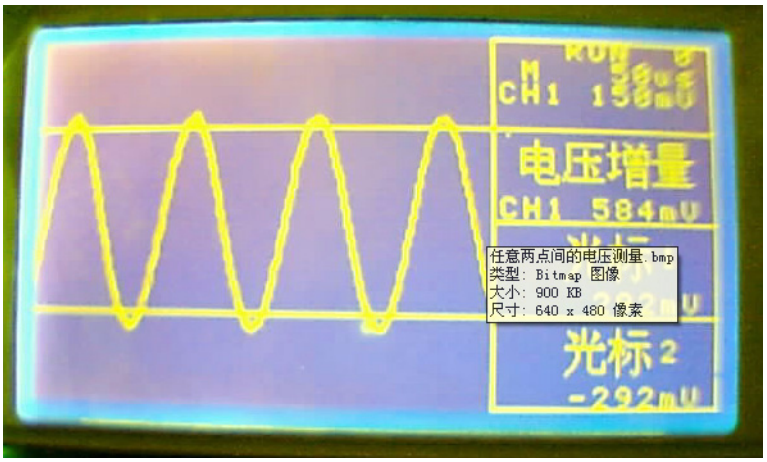
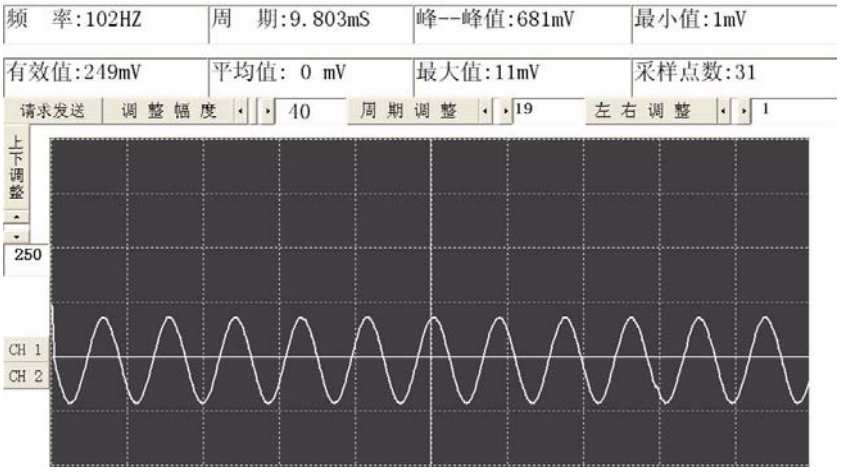


Figure 30. Self-Measured Signal Frequency Phase Setting

Figure 31. Computer Screen Display for Serial Port Communications

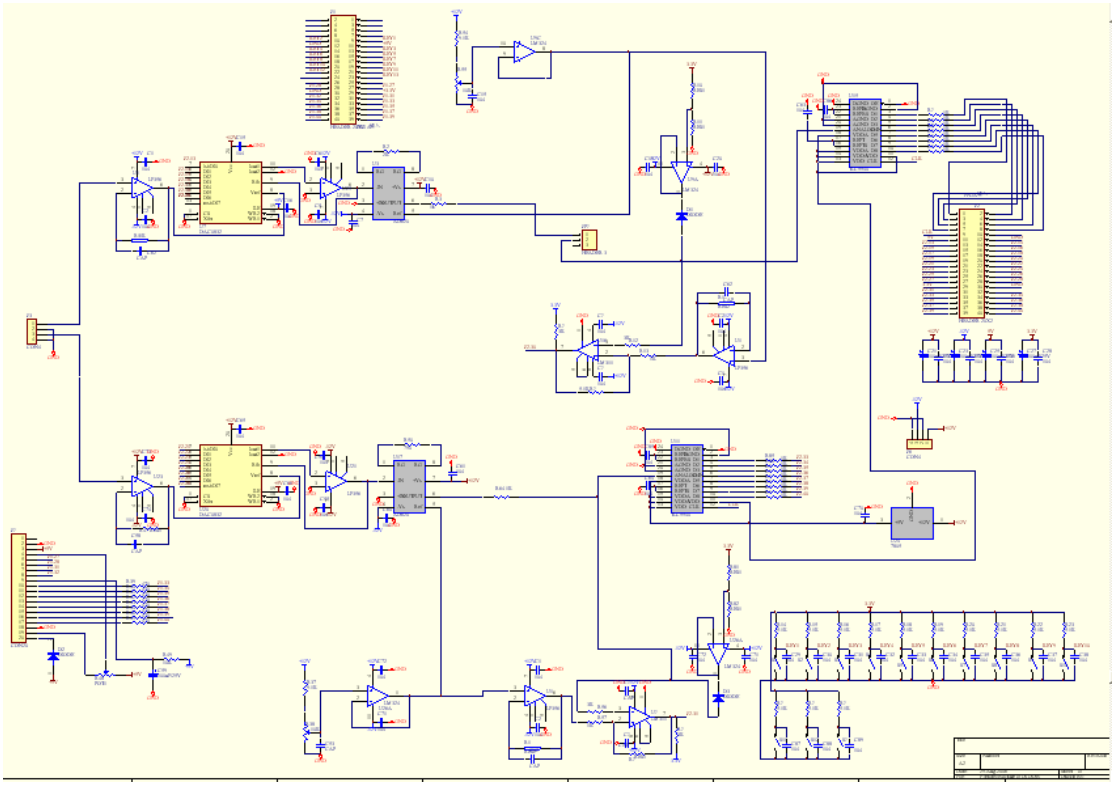


Appendix: Code

Refer to the PDF of this paper on the Altera web site at http://www.altera.com for code that was created for this project.

Appendix: Code Examples

Extended Circuit



VHDL source program measured by the equivalent precision frequency:

1) fen_pin_u module source program

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity fen_pin_u is
port( M: in integer;
      b:in std_logic;
     inclock:in std_logic ;
 outclk: out std_logic );
end fen_pin_u;
architecture ma of fen_pin_u is
signal count: integer;
signal clk1: std_logic ;
signal N: integer;
begin
a1:
process (M,b,inclock)
begin
N<=M;
if b='0'  then
if (inclock'event and inclock='1') then
if (count>=N-1) then
count<=0;
else
count<=count+1;
if count <(integer(N/2)) then
clk1 <='0';
else
```

```
clk1<='1';
end if ;
end if;
end if;
else
clk1<='0';
end if;
end process a1;
outclk<=clk1;
end ma;
```

2) jishuqi module

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity jishuqi is
generic( N:integer:=5000);
port(clr,q,f:in  std_logic;
        qout:out std_logic);
end jishuqi;
architecture mm of jishuqi is
signal clk :std_logic;
signal   k :integer;
begin
clk<=q and f;
process(clk,clr)
begin
if clr='1' then
k<=0;
qout<='0';
else
if k>=N then
k<=0;
qout<='1';
elsif clk'event and clk='1' then
k<=k+1;
qout<='0';
end if;
end if;
end process;
end mm;
```

3) chufa module source program

```
library  ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity chufa is
port(clr,en:in std_logic;
    q:out std_logic);
end chufa;
architecture mmm of chufa is
begin
process(clr,en)
begin
if clr='1' then
q<='0';
elsif en'event and en='1' then
q<='1';
end if;
end process;
end mmm;
```

4) clk_cnt module source program

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity clk_cnt is
```

```
port(bclk,gate,clr,xclk:in std_logic;
    men: out std_logic;
    databus_0:out std_logic_vector(31 downto 0);
    databus_1:out std_logic_vector(31 downto 0));
end clk_cnt;
architecture counter of clk_cnt is
signal bz_count,dc_count:std_logic_vector(31 downto 0);
signal bz_ena,dc_ena:std_logic;
begin
databus_0<=bz_count(31 downto 0);
databus_1<=dc_count(31 downto 0);
bzcounter:
process(bclk,clr,gate)
begin
if clr='1' then
bz_count<=(others=>'0');
elsif bclk'event and bclk='1' then
if bz_ena='1' then
bz_count<=bz_count+1;
end if;
end if;
end process;
dccounter:
process(xclk,clr,gate)
begin
if clr='1' then
dc_count<=(others=>'0');
elsif xclk'event and xclk='1' then
if dc_ena='1' then
 dc_count<=dc_count+1;
end if;
end if;
end process;
DQ:
process(xclk,clr)
begin
if clr='1' then
dc_ena<='0';
elsif xclk'event and xclk='1' then
dc_ena<=gate;
end if;
end process;
bz_ena<=dc_ena;
men<=dc_ena;
end counter;
```

2 Source program of equivalent time-effectiveness measurement

1) di module source program

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity  di is
port
( d,enable: in std_logic;
      n:in std_logic_vector(3 downto 0);
  q: out std_logic);
end entity;
architecture mm of di is
signal count2: std_logic_vector(3 downto 0):="0000";
signal count3: std_logic_vector(3 downto 0);
begin
count3<=n-"0001";
process(enable,d,count3)
begin
if enable='1' then
q<='0';
else
if d'event and d='1' then
```

```
if count2=(count3-"0001") then
   q<='1';
   count2<=count2+"0001";
elsif count2>=count3 then
 q<='1';
 count2 <="0000";
else
 count2 <=count2+"0001";
 q<='0';
end if;
end if;
end if;
end process;
end mm;
```

2) cont module source program

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity cont is
port(clk, d : in std_logic;
     q :out std_logic_vector(7 downto 0);
     m,set :in std_logic_vector(7 downto 0));
end entity;
architecture mm of cont is
signal count : std_logic_vector(7 downto 0):="00000000";
signal count1: std_logic_vector(7 downto 0):="00000000";
signal count2: std_logic_vector(7 downto 0):="00000001";
begin
process(d,m,set)
begin
if d'event and d='1' then
if count2<set then
count<=count+m;
count2<=count2+"00000001";
else
count<="00000000";
count2<="00000001";
end if ;
end if;
if d'event and d='0' then
 q<=count;
end if;
end process;
end mm;
```

3) maic module source program

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity maic is
port(d :in  std_logic_vector(7 downto 0);
     ds,clk :in  std_logic;
      q:out std_logic);
end entity ;
architecture ma of maic is
signal count1,count2:std_logic_vector(7 downto 0):="00000000";
begin
process(ds,clk)
begin
if clk'event and clk='0' then
count1<=d;
end if;
if clk'event and clk='1' then
if ds='1' then
if count2>count1 then
q<='1';
 count2<=count2;
```

```
else
q<='0';
count2<=count2+"00000001";
end if ;
else
q<='0';
count2<="00000000";
end if;
end if;
end process;
end ma;
```

4) maic_1 module source program

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity maic_1 is
port(d :in  std_logic_vector(7 downto 0);
    ds,clk :in  std_logic;
     q:out std_logic);
end entity ;
architecture ma of maic_1 is
signal count1,count2:std_logic_vector(7 downto 0):="00000000";
begin
process(ds,clk)
begin
if ds'event and ds='0' then
count1<=d;
end if;
if clk'event and clk='1' then
if ds='0' then
if count2>count1 then
 q<='0';
 count2<=count2;
else
q<='1';
count2<=count2+"00000001";
end if ;
else
q<='1';
count2<="00000000";
end if;
end if;
end process;
end ma;
```

5) mm module source program

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity mm is
port(clk,clr: in std_logic;
    q: out std_logic);
end entity;
architecture nn of mm is
 begin
process(clr,clk)
begin
if clr='0' then
q<='0';
else
if clk'event and clk='1' then
q<='1';
end if ;
end if;
end process;
end nn;
```

3 Triggering source program

Bijiaoqi module source program

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity bijiaoqi is
port(datain :in std_logic_vector(7 downto 0);
   clk    :in std_logic;
    m     :in std_logic_vector(7 downto 0);
    q     :out std_logic;
    q2    :out std_logic_vector(7 downto 0));
end entity;
architecture mm of bijiaoqi is
signal bit1:std_logic_vector(7 downto 0);
begin
bit1(7 downto 2)<=datain(7 downto 2);
bit1 (1 downto 0)<=null;
process(bit1,clk)
begin
if clk'event and clk='1' then
if bit1 > m then
q<= '1';
else
q<='0';
end if ;
end if;
end process;
q2<=bit1;
end mm;
```

4 Source program of different waveform signal generated by DDS

yuan_1 module source program

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity yuan_1 is
port(  clk:in std_logic;
      deng1,deng2:in std_logic_vector(31 downto 0);
          q1,q2:out std_logic_vector(8 downto 0));
end entity;
architecture mm of yuan_1 is
signal count1,count2,count3 :std_logic_vector(31 downto 0):="00000000000000000000000000000000";
signal count4 :std_logic_vector(8 downto 0);
begin
count1<=deng1;
count2<=deng2;
q1<=count3(31 downto 23);
q2<=count4;
process(clk,count1,count2)
begin
if clk'event and clk='1' then
count3<=count3+count1;
count4<=(count3(31 downto 23)+count2(8 downto 0));
end if;
end process;
end mm;
```

5 VB major program and communications

```
Private Sub form_load()
Text1.Text = "frequency"
Text2.Text = "cycle"
Text3.Text = "peak—peak value"
Text4.Text = "virtual value"
Text5.Text = "mean value"
Text6.Text = "max. value"
```

```
Text7.Text = “min. value”
Text8.Text = “sampling points”
Text11.Text = Str$(HScroll4.Value)
Text10.Text = Str$(HScroll2.Value)
Text9.Text = Str$(HScroll3.Value)
Text12.Text = Str$(VScroll1.Value)
MSComm1.CommPort = 1
MSComm1.PortOpen = True
MSComm1.Settings = “115200,n,8,1”
MSComm1.InputLen = 0
MSComm1.RThreshold = 1024
MSComm1.InBufferCount = 0
MSComm1.InputMode = comInputModeBinary
MSComm1.NullDiscard = False
End Sub

Private Sub MSComm1_OnComm()
Picture1.Cls
c = 0
pin (void)
On Error Resume Next
    With MSComm1
    Select Case MSComm1.CommEvent
    Case comEvReceive
    bytData = .Input
    ReDim s(UBound(bytData)) As Byte
    For I = 0 To UBound(bytData)
    s(I) = bytData(I)
    Next I
    For I = 0 To 471
    f(I) = s(I)
    Next I
    Do While I <= 511
    f(I - c * 3) = zh(I, s())
    I = I + 4
    c = c + 1
    Loop
    For I = 512 To 983
    f(I - 30) = s(I)
    Next I
    Do While I <= 1023
    f(I - c * 3 - 30) = zh(I + 0, s())
    I = I + 4
    c = c + 1
    Loop
    tin f()
    For I = 472 To 481
    b(I - 472) = f(I)
    Next I
    For I = 954 To 963
    b(I - 944) = f(I)
    Next I
    try2 Text1, b(0)
    try3 Text2, b(1)
    Text3.Text = “peak—peak value:” & b(2) \ 10 & “mV”
    Text4.Text = “virtual value:” & b(3) \ 10 & “mV”
    If b(4) < 0 Or b(4) > 1000000 Then
    Text5.Text = “mean value: 0 mV”
    Else: Text5.Text = “mean value:” & b(4) \ 100 & “mV”
    End If
    Text6.Text = “max. value:” & b(5) \ 10 & “mV”
    Text7.Text = “min. value:” & b(6) \ 10 & “mV”
    Text8.Text = “sampling points:” & b(7)
    End Select
    End With
End Sub
```

6 C program of soft-core

1.main.c source program

```
#include <stdio.h>
#include <unistd.h>
#include <math.h>
#include <string.h>
#include “altera_avalon_uart_regs.h”
#include “altera_avalon_uart.h”
#include “altera_avalon_uart_regs.h”
#include “altera_avalon_pio_regs.h”
#include “sys/alt_flash.h”
#include “sys/alt_irq.h”
#include “system.h”
#include “CHARBANK.H”
#include “SD_Card.h”
#include “MAIN.H”
#include “LCD.H”
#include “fft.h”

/********************* main function ********************/
int main ( void )
{int i=0;
 lcd_initial();
 lcd_clear_ram();
 Init_CGRAM();
 usleep(1000);
 Draw_Grid();
 sys_clk_timer_init();
 key_init();
 button_init();
 uart_init();
 IOWR(DSX_ENA1_BASE,0,1);
 IOWR(DAC0832_DB1_BASE,0,DIN[CH1_V_Index]);
 IOWR(DAC0832_DB2_BASE,0,DIN[CH2_V_Index]);
 IOWR(FP_FS_BASE,0,FS_FP[T_Index]);
 IOWR(TRIGER1_LEVEL_BASE,0,trigger_level);
 IOWR(TRIGER1_SEL_BASE,0,1);
 IOWR(TRIGER2_LEVEL_BASE,0,trigger_level);
 IOWR(TRIGER2_SEL_BASE,0,0);
 IOWR(FX1_CLR_BASE,0,0);
 IOWR(FX1_START_BASE,0,0);
 IOWR(FP_FX1_BASE,0,5000000);
 IOWR(FX2_CLR_BASE,0,0);
 IOWR(FX2_START_BASE,0,0);
 IOWR(FP_FX2_BASE,0,5000000);
 cepin_start();
 continue_trigger();
 measure();
 autoset();
 Display_Wave();
 while(1)
 {
  if(status2_ttype_flag)
     continue_trigger();
  else
     single_trigger();
  for(i=0;i<472;i++)
    {uart_data[i]=buff1_data[i];}
  for(;i<482;i++)
    {uart_data[i]=buff1_data[i+40];}
  for(;i<954;i++)
    {uart_data[i]=buff2_data[i-482];}
  for(;i<964;i++)
    {uart_data[i]=buff2_data[i-482+40];}
 if(key_fft==0)
  {
   if(key_zice==0)
      Display_Wave();
   if(IORD(FX1_GATE_BASE,0)&&run_stop_flag)
    {
    measure();
    printf(“CH1_f=%dHz\n”,CH1_F);
```

```
      printf("CH2_f=%dHz\n",CH2_F);
      cepin_start();
      if(key_dsx)
        {
         IOWR(DSX_N1_BASE,0,3);
         IOWR(DSX_SET1_BASE,0,32);
         i=(160000000>>5)/CH1_F;
         IOWR(DSX_M1_BASE,0,i);
        }
      }
   Refresh();
   }
   if(IORD(SWITCH_PIO_BASE,0)&0x01)
     {IOWR(DSX_ENA1_BASE,0,0);key_dsx=1;}
   else
     { IOWR(DSX_ENA1_BASE,0,1);key_dsx=0;}
  }
}

void Refresh(void)
{int i=0,j=0;

  if(key_measure)
    {measure_Refresh(1);measure_Refresh(2);measure_Refresh(3);}
  if(key_cursor)
    cursor_Refresh();
 if((key_cursor==0)&&(key_zice==0))
    {
     for(i=0;i<20;i++)
      for(j=0;j<16;j++)
        lcd_disp_text(i,j,0x00);
     grid_Refresh();
    }
  if(key_save)
    save_Refresh();
  if(key_trigger)
    trigger_Refresh();
  if(key_fft)
    {fft_Refresh();}
  if(key_fft==0)
    {
          lcd_disp_string(20,4,"          ",10);
          lcd_disp_string(20,6,"          ",10);
          lcd_disp_string(20,8,"          ",10);
          lcd_disp_string(20,10,"          ",10);
    }
  if(key_fft==0&&key_zice==0)
    {
      lcd_disp_string(20,0,"          ",10);
      lcd_disp_text(28,0,(shift_flag+0x10));
      if(run_stop_flag)
        lcd_disp_string(23,0,"RUN  ",5);
      else
        lcd_disp_string(23,0,"STOP ",5);
      lcd_disp_string(20,1," M  ",4);
      lcd_disp_string(24,1,TDIV[T_Index],5);

      if(ch_flag==1)
        {
         lcd_disp_string(20,2,"CH1 ",4);
         lcd_disp_string(24,2,VDIV[CH1_V_Index],5);
         lcd_disp_string(20,3,"          ",10);
        }
      if(ch_flag==2)
        {
         lcd_disp_string(20,3,"CH2 ",4);
         lcd_disp_string(24,3,VDIV[CH2_V_Index],5);
         lcd_disp_string(20,2,"          ",10);
        }
      if(ch_flag==3)
```

```
      {
       lcd_disp_string(20,2,"CH1 ",4);
       lcd_disp_string(24,2,VDIV[CH1_V_Index],5);
       lcd_disp_string(20,3,"          ",10);
       lcd_disp_string(20,3,"CH2 ",4);
       lcd_disp_string(24,3,VDIV[CH2_V_Index],5);
      }
   }
   if(key_zice)
   {
   zice_Refresh();
   }
}

void measure_Refresh(unsigned char status)
{
 unsigned char status_mtype_flag,status_mch_flag,y1,y2;

 switch(status)
   {case 1:status_mtype_flag=status1_mtype_flag;
        status_mch_flag=status1_mch_flag;
        y1=36;
        y2=7;
        break;
    case 2:status_mtype_flag=status2_mtype_flag;
        status_mch_flag=status2_mch_flag;
        y1=68;
        y2=11;
        break;
    case 3:status_mtype_flag=status3_mtype_flag;
        status_mch_flag=status3_mch_flag;
        y1=100;
        y2=15;
        break;
   }

    if(status_mch_flag)
      {lcd_disp_string(20,y2-2,"CH2       ",10);
       lcd_disp_string(20,y2,"          ",10);
       if(ch_flag==1)
         lcd_disp_hanzi(21,y1,guanbi,4);
       else
       switch(status_mtype_flag)
         {case 0:lcd_disp_hanzi(21,y1,pinlv,4);
                  ch_f_disp(2,y2);
                  break;
          case 1:lcd_disp_hanzi(21,y1,zhouqi,4);
                 ch_t_disp(2,y2);
                break;
          case 2:lcd_disp_hanzi(21,y1,fengfengzhi,4);
                 ch_pv_disp(2,y2);
                 break;
          case 3:lcd_disp_hanzi(21,y1,youxiaozhi,4);
                 ch_vv_disp(2,y2);
                 break;
          case 4:lcd_disp_hanzi(21,y1,pingjunzhi,4);
                 ch_av_disp(2,y2);
                 break;
          case 5:lcd_disp_hanzi(21,y1,wu,4);
                 lcd_disp_string(20,y2,"          ",10);
                 break;
         }
       }
     else
       {lcd_disp_string(20,y2-2,"CH1       ",10);
       lcd_disp_string(20,y2,"          ",10);
       if(ch_flag==2)
         lcd_disp_hanzi(21,y1,guanbi,4);
       else
       switch(status_mtype_flag)
```

```
          {case 0:lcd_disp_hanzi(21,y1,pinlv,4);
                  ch_f_disp(1,y2);
                  break;
           case 1:lcd_disp_hanzi(21,y1,zhouqi,4);
                  ch_t_disp(1,y2);
                 break;
           case 2:lcd_disp_hanzi(21,y1,fengfengzhi,4);
                  ch_pv_disp(1,y2);
                  break;
           case 3:lcd_disp_hanzi(21,y1,youxiaozhi,4);
                  ch_vv_disp(1,y2);
                  break;
           case 4:lcd_disp_hanzi(21,y1,pingjunzhi,4);
                  ch_av_disp(1,y2);
                  break;
           case 5:lcd_disp_hanzi(21,y1,wu,4);
                  lcd_disp_string(20,y2,"          ",10);
                  break;
          }
      }
}

void cursor_Refresh(void)
{ int i=0,j=0;
  lcd_write(CGRAMSA>>11);lcd_write(0x00);lcd_ctrl(0x22);
    for(i=0;i<20;i++)
     for(j=0;j<16;j++)
       lcd_disp_text(i,j,0x00);
 if(status1_ctype_flag)
   {
   if(status1_ctype_flag==1)
     lcd_disp_string(20,7,"CH1",3);
   else if(status1_ctype_flag==2)
        lcd_disp_string(20,7,"CH2",3);
    lcd_disp_hanzi(21,36,dianya,2);
    lcd_disp_hanzi(25,36,zhengliang,2);
    V_disp(1);
    lcd_disp_hanzi(23,68,guangbiao,2);
    lcd_disp_text(27,9,0x11);
    V_disp(2);
    lcd_disp_hanzi(23,100,guangbiao,2);
    lcd_disp_text(27,13,0x12);
    V_disp(3);
    for(i=0;i<20;i++)
       {
       lcd_disp_text(i,line_V1>>3,(line_V1%8)|0xf0);
       lcd_disp_text(i,line_V2>>3,(line_V2%8)|0xf0);
       }
   }
 else
   {
    lcd_disp_hanzi(21,36,shijian,2);
    lcd_disp_hanzi(25,36,zhengliang,2);
    T_disp(1);
    lcd_disp_hanzi(23,68,guangbiao,2);
    lcd_disp_text(27,9,0x11);
    T_disp(2);
    lcd_disp_hanzi(23,100,guangbiao,2);
    lcd_disp_text(27,13,0x12);
    T_disp(3);
    for(i=0;i<16;i++)
       {
       lcd_disp_text(line_T2>>3,i,(line_T2%8)|0xf8);
       lcd_disp_text(line_T1>>3,i,(line_T1%8)|0xf8);
       }
   }
}

void save_Refresh(void)
{
```

```
 lcd_disp_hanzi(21,36,jiyixuanzhe,4);
 lcd_disp_hanzi(21,68,chunchu,4);
 lcd_disp_string(20,11,"          ",10);
 lcd_disp_hanzi(21,100,diaochu,4);
 lcd_disp_string(20,15,"          ",10);
  if(status1_stype_flag<=10)
     {
     lcd_disp_string(20,7," flash    ",10);
     lcd_disp_text(27,7,(status1_stype_flag/10+0x10));
     lcd_disp_text(28,7,(status1_stype_flag%10+0x10));
     }
  else
     {
     lcd_disp_string(20,7,"SDcard    ",10);
     lcd_disp_text(27,7,(status1_stype_flag/10-1+0x10));
     lcd_disp_text(28,7,(status1_stype_flag%10+0x10));
     }
  lcd_disp_string(20,15,"          ",10);
}

void trigger_Refresh(void)
{
 if(ch_flag==1)
   lcd_disp_string(20,7,"    CH1   ",10);
 else if(ch_flag==2)
        lcd_disp_string(20,7,"    CH2   ",10);
      else if(ch_flag==3)
             lcd_disp_string(20,7," CH1  CH2 ",10);

 if(status2_ttype_flag)
    lcd_disp_hanzi(21,68,lianxu,2);
 else
    lcd_disp_hanzi(21,68,danci,2);

 if(!status3_ttype_flag)
    {
    lcd_disp_hanzi(21,100,dianping,2);
    lcd_disp_string(23,15,CHUFA[trigger_level/20],3);
    }
 else
    {
    lcd_disp_hanzi(21,100,bianyan,2);
    lcd_disp_string(20,15,"          ",10);
    }
}

void fft_Refresh(void)
{
 int i=0,j=0;
  float temp_data[128],max;
  lcd_disp_string(20,0,"          ",10);
  lcd_disp_string(20,1,"          ",10);
  lcd_disp_string(20,2,"          ",10);
  lcd_disp_string(20,3,"          ",10);
  lcd_disp_hanzi(21,4,pinpufenxi,4);
  if(shift_flag)
     lcd_disp_string(20,3,"    CH2   ",10);
  else
     lcd_disp_string(20,3,"    CH1   ",10);
   if(status3_fft_flag)
   {
    max=1.2*fft_data[1];
    for(i=0;i<128;i++)
      temp_data[i]=127*(1-fft_data[i]/max);
    lcd_disp_hanzi(21,100,quxiao,4);
    for(i=0;i<20;i+=2)
      for(j=temp_data[i>>1];j<128;j++)
      {
        lcd_set_graph(i,j);
        lcd_ctrl(0xff);
```

```
    }
  lcd_disp_text(1,1,0x11);
  lcd_disp_text(3,1,0x12);
  lcd_disp_text(5,1,0x13);
  lcd_disp_text(7,1,0x14);
  lcd_disp_text(9,1,0x15);
  lcd_disp_text(11,1,0x16);
  lcd_disp_text(13,1,0x17);
  lcd_disp_text(15,1,0x18);
  lcd_disp_text(17,1,0x19);
 }
 else
  {
  lcd_disp_hanzi(21,100,queren,4);
  lcd_disp_text(1,1,0x00);
  lcd_disp_text(3,1,0x00);
  lcd_disp_text(5,1,0x00);
  lcd_disp_text(7,1,0x00);
  lcd_disp_text(9,1,0x00);
  lcd_disp_text(11,1,0x00);
  lcd_disp_text(13,1,0x00);
  lcd_disp_text(15,1,0x00);
  lcd_disp_text(17,1,0x00);
  lcd_disp_string(20,4,"          ",10);
  lcd_disp_string(20,5,"          ",10);
  lcd_disp_string(20,6,"          ",10);
  lcd_disp_string(20,7,"          ",10);
  lcd_disp_string(20,8,"          ",10);
  lcd_disp_string(20,9,"          ",10);
  lcd_disp_string(20,10,"          ",10);
  lcd_disp_string(20,11,"          ",10);
  //Display_Wave();
  }
}



void grid_Refresh(void)
{
  unsigned int i,j;
  lcd_write(CGRAMSA>>11);lcd_write(0x00);lcd_ctrl(0x22);
  lcd_disp_text(0,0,0xc0);
  for(i=1;i<18;)
  {
    lcd_disp_text(i++,0,0xc1);
    lcd_disp_text(i++,0,0xc2);
  }
  lcd_disp_text(19,0,0xc3);
  for(j=1;j<15;j+=2)
  {
    lcd_disp_text(0,j,0xc5);
    for(i=2;i<19;i+=2)
      lcd_disp_text(i,j,0xc6);
    lcd_disp_text(19,j,0xc7);
  }
  for(j=2;j<15;j+=2)
  {
    lcd_disp_text(0,j,0xc8);
    for(i=1;i<19;)
    {
      lcd_disp_text(i++,j,0xc9);
      lcd_disp_text(i++,j,0xca);
    }
    lcd_disp_text(19,j,0xcb);
  }
  lcd_disp_text(0,15,0xcc);
  for(i=1;i<19;)
  {
    lcd_disp_text(i++,15,0xcd);
    lcd_disp_text(i++,15,0xce);
```

```
  }
  lcd_disp_text(19,15,0xcf);

  for(i=1;i<19;)
    {
     lcd_disp_text(i,7,0xd2);
     lcd_disp_text(i++,8,0xd5);
     lcd_disp_text(i,7,0xd1);
     lcd_disp_text(i++,8,0xd6);
    }
  lcd_disp_text(0,7,0xd3);
  lcd_disp_text(19,7,0xd4);
  lcd_disp_text(0,8,0xd7);
  lcd_disp_text(19,8,0xd8);

  for(j=1;j<15;)
    {
     lcd_disp_text(9,j,0xd9);
     lcd_disp_text(10,j++,0xdd);
     lcd_disp_text(9,j,0xda);
     lcd_disp_text(10,j++,0xde);
    }
  lcd_disp_text(9,0,0xdb);
  lcd_disp_text(9,15,0xdc);
  lcd_disp_text(10,0,0xdf);
  lcd_disp_text(10,15,0xe0);
  lcd_disp_text(9,7,0xe1);
  lcd_disp_text(9,8,0xe2);
  lcd_disp_text(10,7,0xe3);
  lcd_disp_text(10,8,0xe4);

}
```

2.main.h source program

```
void uart_init(void)
{
 IOWR(UART_BASE,2,0);
 IOWR(UART_BASE,3,0x80);
 alt_irq_register(UART_IRQ, NULL, uart_irq);
}

void uart_irq(void *context,alt_u32 interrupt)
{
 static int i=0,j=0,k=0;
 if(IORD(UART_BASE,0)-0x30)
 {
  IOWR(UART_BASE,2,0);
  IOWR(UART_BASE,3,0x40);
 if(k<472)
   {IOWR(UART_BASE,1,uart_data[i++]&0xff);k++;}
 else if(k<512)
        switch(j++%4)
        {
         case 0:i++;IOWR(UART_BASE,1,uart_data[i]&0xff);k++;break;
         case 1:IOWR(UART_BASE,1,(uart_data[i]>>8)&0xff);k++;break;
         case 2:IOWR(UART_BASE,1,(uart_data[i]>>16)&0xff);k++;break;
         case 3:IOWR(UART_BASE,1,(uart_data[i]>>24)&0xff);k++;break;
        }
       else if(k<984)
            {IOWR(UART_BASE,1,uart_data[i++]&0xff);k++;}
           else
             switch(j++%4)
             {
              case 0:i++;IOWR(UART_BASE,1,uart_data[i]&0xff);k++;break;
              case 1:IOWR(UART_BASE,1,(uart_data[i]>>8)&0xff);k++;break;
              case 2:IOWR(UART_BASE,1,(uart_data[i]>>16)&0xff);k++;break;
              case 3:IOWR(UART_BASE,1,(uart_data[i]>>24)&0xff);k++;break;
             }
 if(i==964&&j%4==0)
```

```
  { i=0;
    j=0;
    k=0;
    IOWR(UART_BASE,3,0x80);
  }
 }
 IOWR(UART_BASE,2,0);
}

void button_init()
{
  IOWR(BUTTON_PIO_BASE,2,0xff);
  IOWR(BUTTON_PIO_BASE,3,0x00);
  alt_irq_register(BUTTON_PIO_IRQ,NULL,button_irq);
}

void button_irq(void* context,alt_u32 id)
{
  switch(IORD(BUTTON_PIO_BASE,3))
  {
  case 4:printf("key 2 have been press! \n");
           key_measure=0;
           key_cursor=0;
           key_save=0;
           key_trigger=0;
           key_fft=0;
           key_zice=1;
           clear();
           zice_disp();break;
  }
    usleep(20000);
  switch(IORD(BUTTON_PIO_BASE,0))
  {
    case 0xe:printf("key 0 have been press! \n");
         if((key_fft==0)&&(key_zice==0))
           {
           run_stop_flag=~run_stop_flag;
           if(run_stop_flag)
             {
              IOWR(RUN_STOP_BASE,0,1);
              lcd_disp_string(20,0,"   RUN  ",8);
             }
           else
             {
              IOWR(RUN_STOP_BASE,0,0);
              lcd_disp_string(20,0,"   STOP ",8);
             }
           }
         break;
    case 0xd:printf("key 1 have been press! \n");
         if((key_fft==0)&&(key_zice==0))
           {
           lcd_disp_string(23,0,"AUTO",4);
           autoset();
           if(run_stop_flag)
             {
              IOWR(RUN_STOP_BASE,0,1);
              lcd_disp_string(20,0,"   RUN  ",8);
             }
           else
             {
              IOWR(RUN_STOP_BASE,0,0);
              lcd_disp_string(20,0,"   STOP ",8);
             }
           }
         break;
  }
  IOWR(BUTTON_PIO_BASE,3,0);
}
```

```
void sys_clk_timer_init()
{
  IOWR(SYS_CLK_TIMER_BASE,2,0x4240);
  IOWR(SYS_CLK_TIMER_BASE,3,0x000f);
  IOWR(SYS_CLK_TIMER_BASE,1,1);
  IOWR(SYS_CLK_TIMER_BASE,0,0);
  alt_irq_register(SYS_CLK_TIMER_IRQ,NULL,sys_clk_timer_int);
  IOWR(SYS_CLK_TIMER_BASE,1,5);
}

void sys_clk_timer_int (void* context ,alt_u32 id)
{
 static int flag=0,led=1;
    if(led&0x81)
      flag=led^1;
    if(flag)
      led=led>>1;
    else
      led=led<<1;
  IOWR(LED_GREEN_BASE,0,led);
  IOWR(SYS_CLK_TIMER_BASE,0,0);
  IOWR(SYS_CLK_TIMER_BASE,1,5);
}

void timer1_init()
{
  IOWR(TIMER1_BASE,2,0x4240);
  IOWR(TIMER1_BASE,3,0x000f);
  IOWR(TIMER1_BASE,1,1);
  IOWR(TIMER1_BASE,0,0);
  alt_irq_register(TIMER1_IRQ,NULL,timer1_int);
  IOWR(TIMER1_BASE,1,5);
}

void timer1_int (void* context ,alt_u32 id)
{
  IOWR(WATCHDOG_BASE,3,0x1234);
  IOWR(TIMER1_BASE,0,0);
  IOWR(TIMER1_BASE,1,5);
}

void key_init()
{
  IOWR(KEY_EXT_BASE,2,0x1fff);
  IOWR(KEY_EXT_BASE,3,0x000);
  if(alt_irq_register(KEY_EXT_IRQ,NULL,key_int))
    printf("key error");
}

void key_int (void* context,alt_u32 id)
{int i=0,j=0;
 switch(IORD(KEY_EXT_BASE,3))
 {
 case 1:printf("key 0 have been press! \n");
            key_measure=1;
            key_cursor=0;
            key_save=0;
            key_trigger=0;
            key_fft=0;
            key_zice=0;
            clear();
            measure_disp();
        break;
 case 2:printf("key 1 have been press! \n");
            key_measure=0;
            key_cursor=1;
            key_save=0;
            key_trigger=0;
            key_fft=0;
            key_zice=0;
```

```
            clear();
            for(i=0;i<20;i++)
              for(j=0;j<16;j++)
                 lcd_disp_text(i,j,0x00);
            cursor_disp();
       break;
case 4:printf("key 2 have been press! \n");
          key_save=0;
          key_measure=0;
          key_cursor=0;
          key_fft=0;
          key_trigger=1;
          key_zice=0;
           clear();
          lcd_disp_hanzi(21,36,xinyuan,4);
          lcd_disp_hanzi(25,68,chufa,2);
          lcd_disp_string(20,11,"          ",10);
          lcd_disp_hanzi(25,100,chufa,2);
          lcd_disp_string(20,15,"          ",10);
       break;
case 8:printf("key 3 have been press! \n");
          key_save=1;
          key_measure=0;
          key_cursor=0;
          key_trigger=0;
          key_fft=0;
          key_zice=0;
          clear();
          save_disp();
       break;
case 16:printf("key 4 have been press! \n");
          key_fft=1;
          key_save=0;
          key_measure=0;
          key_cursor=0;
          key_trigger=0;
          key_zice=0;
          clear();
          fft0_disp();
       break;
}
usleep(20000);
 switch(IORD(KEY_EXT_BASE,0))
 {
  case 0x1fdf:printf("key 5 have been press! \n");
          if(key_zice==1)
           zice0_disp();
          else
           {
            if(shift_flag==1)
             shift_flag=0;
            else
             shift_flag++;
            if(key_fft)
              {
              if(shift_flag)
                 lcd_disp_string(20,3,"    CH2   ",10);
              else
                 lcd_disp_string(20,3,"    CH1   ",10);
              }
             else
              lcd_disp_text(28,0,(shift_flag+0x10));
           }
            break;
  case 0x1fbf:printf("key 6 have been press! \n");
            if(key_measure)
              measure123_disp(1);
            if(key_cursor)
              cursor1_disp();
             if(key_trigger)
```

```
            trigger1_disp();
          if(key_save)
            save1_disp();
          if(key_zice)
            zice1_disp();
          break;
case 0x1f7f:printf("key 7 have been press! \n");
          if(key_measure)
            measure123_disp(2);
          if(key_cursor)
            cursor2_disp();
          if(key_trigger)
            trigger2_disp();
          if(key_save)
            save2_disp();
          if(key_zice)
            zice2_disp();
          break;
case 0x1eff:printf("key 8 have been press! \n");
          if(key_measure)
            measure123_disp(3);
          if(key_cursor)
            cursor3_disp();
          if(key_trigger)
            trigger3_disp();
          if(key_save)
            save3_disp();
          if(key_fft)
            fft3_disp();
          if(key_zice)
            zice3_disp();
          break;
case 0x1dff:printf("key 9 have been press! \n");
          if(key_zice==0)
          {
          if(key_fft==0)
          {
           if(shift_flag==0&&T_Index>0)
            {
              T_Index--;
              IOWR(FP_FS_BASE,0,FS_FP[T_Index]);
              lcd_disp_string(24,1,TDIV[T_Index],5);
            }
           else if(shift_flag==1&&T_addr<520)
               T_addr+=8;
          }
          }
          else if(status2_zice_flag==3)
               {
                if(status4_zice_flag==8)
                  status4_zice_flag=0;
                else
                  status4_zice_flag++;
                switch(status4_zice_flag)
                 {
                 case 0:lcd_ctrl(0x9c);
                        break;
                 case 1:lcd_ctrl(0x9f);
                       lcd_write(0x09);lcd_write(0x05);lcd_ctrl(0x21);
                        break;
                 case 2:lcd_ctrl(0x9f);
                       lcd_write(0x13);lcd_write(0x05);lcd_ctrl(0x21);
                        break;
                 case 3:lcd_ctrl(0x9f);
                       lcd_write(0x09);lcd_write(0x08);lcd_ctrl(0x21);
                        break;
                 case 4:lcd_ctrl(0x9f);
                       lcd_write(0x13);lcd_write(0x08);lcd_ctrl(0x21);
                        break;
                 case 5:lcd_ctrl(0x9f);
```

```
                    lcd_write(0x09);lcd_write(0x0b);lcd_ctrl(0x21);
                     break;
                case 6:lcd_ctrl(0x9f);
                    lcd_write(0x13);lcd_write(0x0b);lcd_ctrl(0x21);
                     break;
                case 7:lcd_ctrl(0x9f);
                    lcd_write(0x09);lcd_write(0x0e);lcd_ctrl(0x21);
                     break;
                case 8:lcd_ctrl(0x9f);
                    lcd_write(0x13);lcd_write(0x0e);lcd_ctrl(0x21);
                     break;
                }
               }
               else
                {
                if(zice_pv1==10)
                  zice_pv1=0;
                else
                  zice_pv1++;
                lcd_disp_string(9,5,zice_pv[zice_pv1],4);
                }
         break;
case 0x1bff:printf("key 10 have been press! \n");
        if(key_zice==0)
         {
          if(key_fft==0)
          {
          if(shift_flag==0&&T_Index<17)
          {
            T_Index++;
            IOWR(FP_FS_BASE,0,FS_FP[T_Index]);
            lcd_disp_string(24,1,TDIV[T_Index],5);
          }
          else if(shift_flag==1&&T_addr>-168)
             T_addr-=8;
          }
         }
         else if(status2_zice_flag==3)
              {
               switch(status4_zice_flag)
                {
                case 0:
                      break;
                case 1:if(zice_pv2==10)
                         zice_pv2=0;
                       else
                         zice_pv2++;
                       lcd_disp_string(6,5,zice_pv[zice_pv2],4);
                       break;
                case 2:if(zice_pv3==10)
                         zice_pv3=0;
                       else
                         zice_pv3++;
                       lcd_disp_string(16,5,zice_pv[zice_pv3],4);
                       break;
                case 3:if(zice_pv4==10)
                         zice_pv4=0;
                       else
                         zice_pv4++;
                       lcd_disp_string(6,8,zice_pv[zice_pv4],4);
                       break;
                case 4:if(zice_pv5==10)
                         zice_pv5=0;
                       else
                         zice_pv5++;
                       lcd_disp_string(16,8,zice_pv[zice_pv5],4);
                       break;
                case 5:if(zice_pv6==10)
                         zice_pv6=0;
                       else
```

```
                    zice_pv6++;
                  lcd_disp_string(6,11,zice_pv[zice_pv6],4);
                  break;
            case 6:if(zice_pv7==10)
                    zice_pv7=0;
                  else
                    zice_pv7++;
                  lcd_disp_string(16,11,zice_pv[zice_pv7],4);
                  break;
            case 7:if(zice_pv8==10)
                    zice_pv8=0;
                  else
                    zice_pv8++;
                  lcd_disp_string(6,14,zice_pv[zice_pv8],4);
                  break;
            case 8:if(zice_pv9==10)
                    zice_pv9=0;
                  else
                    zice_pv9++;
                  lcd_disp_string(16,14,zice_pv[zice_pv9],4);
                  break;
             }
            }
           else
            {
            if(zice_pv2==10)
              zice_pv2=0;
            else
              zice_pv2++;
            lcd_disp_string(9,13,zice_pv[zice_pv2],4);
            }
         break;
case 0x17ff:printf("key 11 have been press! \n");
        if((key_fft==0)&&(key_zice==0))
        {
        if(shift_flag==0)
         {
          if(((ch_flag==1)||(ch_flag==3))&&(CH1_V_Index>0))
          {
            CH1_V_Index--;
            lcd_disp_string(24,2,VDIV[CH1_V_Index],5);
            IOWR(DAC0832_DB1_BASE,0,DIN[CH1_V_Index]);
          }
          if(((ch_flag==2)||(ch_flag==3))&&(CH2_V_Index>0))
          {
            CH2_V_Index--;
            lcd_disp_string(24,3,VDIV[CH2_V_Index],5);
            IOWR(DAC0832_DB2_BASE,0,DIN[CH2_V_Index]);
          }
         }
        else if(shift_flag==1)
            {
             if(((ch_flag==1)||(ch_flag==3))&&(V_data1<120))
               V_data1+=8;
             if(((ch_flag==2)||(ch_flag==3))&&(V_data2<120))
               V_data2+=8;
            }
         }
         break;
case 0x0fff:printf("key 12 have been press! \n");
        if((key_fft==0)&&(key_zice==0))
        {
        if(shift_flag==0)
         {
          if(((ch_flag==1)||(ch_flag==3))&&(CH1_V_Index<16))
          {
            CH1_V_Index++;
            lcd_disp_string(24,2,VDIV[CH1_V_Index],5);
            IOWR(DAC0832_DB1_BASE,0,DIN[CH1_V_Index]);
          }
```

```
            if(((ch_flag==2)||(ch_flag==3))&&(CH2_V_Index<16))
            {
              CH2_V_Index++;
              lcd_disp_string(24,3,VDIV[CH2_V_Index],5);
              IOWR(DAC0832_DB2_BASE,0,DIN[CH2_V_Index]);
            }
           }
          else if(shift_flag==1)
              {
               if(((ch_flag==1)||(ch_flag==3))&&(V_data1>-120))
                  V_data1-=8;
               if(((ch_flag==2)||(ch_flag==3))&&(V_data2>-120))
                  V_data2-=8;
              }
          }
           break;
 }
 IOWR(KEY_EXT_BASE,3,0);
}

void cepin_start()
{
 IOWR(FX1_CLR_BASE,0,1);
 usleep(1);
 IOWR(FX1_CLR_BASE,0,0);
 usleep(1);
 IOWR(FX1_START_BASE,0,1);
 usleep(20000);
 IOWR(FX1_START_BASE,0,0);

 IOWR(FX2_CLR_BASE,0,1);
 usleep(1);
 IOWR(FX2_CLR_BASE,0,0);
 usleep(1);
 IOWR(FX2_START_BASE,0,1);
 usleep(20000);
 IOWR(FX2_START_BASE,0,0);
}

void autoset()
{int i=0;
 if(ch_flag==1||ch_flag==3)
 {
 if(CH1_F)
  {
   while((CH1_TNUM>64&&T_Index<17)||(CH1_TNUM<20&&T_Index>0))
   {
    if(CH1_TNUM>64)
      T_Index++;
    if(CH1_TNUM<30)
      T_Index--;
   IOWR(FP_FS_BASE,0,FS_FP[T_Index]);
   usleep(1);

   IOWR(BUFF1_START_BASE,0,1);
   usleep(1);
   IOWR(BUFF1_START_BASE,0,0);
   usleep(1);
   while(!IORD(BUFF1_FINISH_BASE,0));
   for( i=1;i<512;i++)
    {
     IOWR(BUFF1_RDADDR_BASE,0,i);
     buff1_data[i]=127-(IORD(BUFF1_DATA_BASE,0)>>1);
    }
    measure();
  }
   while((CH1_MAX-CH1_MIN<64&&CH1_V_Index>0)||(CH1_MAX-CH1_MIN>=120&&CH1_V_Index<17))
      {
       if(CH1_MAX-CH1_MIN<64)
          CH1_V_Index--;
```

```
        if(CH1_MAX-CH1_MIN>=120)
          CH1_V_Index++;
   IOWR(DAC0832_DB1_BASE,0,DIN[CH1_V_Index]);
   usleep(1);
   IOWR(BUFF1_START_BASE,0,1);
   usleep(1);
   IOWR(BUFF1_START_BASE,0,0);
   usleep(1);
   while(!IORD(BUFF1_FINISH_BASE,0));
   for( i=1;i<512;i++)
    {
     IOWR(BUFF1_RDADDR_BASE,0,i);
     buff1_data[i]=127-(IORD(BUFF1_DATA_BASE,0)>>1);
    }
    measure();
    }
   if((key_zice==0)&&(key_fft==0))
   {
   lcd_disp_string(24,1,TDIV[T_Index],5);
   lcd_disp_string(24,2,VDIV[CH1_V_Index],5);
   }
  }
 }
 if(ch_flag==2)
 {
if(CH2_F)
  {
    while((CH2_TNUM>64&&T_Index<17)||(CH2_TNUM<20&&T_Index>0))
    {
     if(CH2_TNUM>64)
       T_Index++;
     if(CH2_TNUM<30)
       T_Index--;
   IOWR(FP_FS_BASE,0,FS_FP[T_Index]);
   usleep(1);
   IOWR(BUFF2_START_BASE,0,1);
   usleep(1);
   IOWR(BUFF2_START_BASE,0,0);
   usleep(1);
   while(!IORD(BUFF2_FINISH_BASE,0));
   for( i=1;i<512;i++)
    {
     IOWR(BUFF2_RDADDR_BASE,0,i);
     buff2_data[i]=127-(IORD(BUFF2_DATA_BASE,0)>>1);
    }
    measure();
  }
   while((CH2_MAX-CH2_MIN<64&&CH2_V_Index>0)||(CH2_MAX-CH2_MIN>=120&&CH2_V_Index<17))
      {
        if(CH2_MAX-CH2_MIN<64)
          CH2_V_Index--;
        if(CH2_MAX-CH2_MIN>=120)
          CH2_V_Index++;
   IOWR(DAC0832_DB2_BASE,0,DIN[CH2_V_Index]);
   usleep(1);
   IOWR(BUFF2_START_BASE,0,1);
   usleep(1);
   IOWR(BUFF2_START_BASE,0,0);
   usleep(1);
   while(!IORD(BUFF2_FINISH_BASE,0));
   for( i=1;i<512;i++)
    {
     IOWR(BUFF2_RDADDR_BASE,0,i);
     buff2_data[i]=127-(IORD(BUFF2_DATA_BASE,0)>>1);
    }
    measure();

      }
   if((key_zice==0)&&(key_fft==0))
   {
```

```
    lcd_disp_string(24,1,TDIV[T_Index],5);
    lcd_disp_string(24,2,VDIV[CH2_V_Index],5);
    }
  }
 }
}

void clear(void)
{
  lcd_ctrl(0x9c);
  //lcd_disp_string(20,4,"          ",10);
  lcd_disp_string(20,5,"          ",10);
  //lcd_disp_string(20,6,"          ",10);
  lcd_disp_string(20,7,"          ",10);
  //lcd_disp_string(20,8,"          ",10);
  lcd_disp_string(20,9,"          ",10);
  //lcd_disp_string(20,10,"          ",10);
  lcd_disp_string(20,11,"          ",10);
  //lcd_disp_string(20,12,"          ",10);
  lcd_disp_string(20,13,"          ",10);
  //lcd_disp_string(20,14,"          ",10);
  lcd_disp_string(20,15,"          ",10);
  lcd_disp_hanzi(21,4,hz_clear,4);
  lcd_disp_hanzi(21,36,hz_clear,4);
  lcd_disp_hanzi(21,68,hz_clear,4);
  lcd_disp_hanzi(21,100,hz_clear,4);
}

void ft_measure(unsigned char ch)
{
  int f=0,t=0,i=0,j=0;
  if(ch==1||ch==3)
    {i=IORD(FX1_COUNT_B_BASE,0);
     j=IORD(FX1_COUNT_X_BASE,0);
      if(i)
        f=(float)j/i*50000000;
      else
        f=0;
      if(f)
        t=1000000/f;
      else
        t=0;
   CH1_F=f;CH1_T=t;
   }
 if(ch==2||ch==3)
   {i=IORD(FX2_COUNT_B_BASE,0);
    j=IORD(FX2_COUNT_X_BASE,0);
    if(i)
      f=(float)j/i*50000000;
    else
      f=0;
    if(f)
      t=1000000/f;
    else
      t=0;
    CH2_F=f;CH2_T=t;
       }
}

void pv_measure(unsigned char ch)
{
  int i=1,max=0,min=0,v_index,data[521];
  if(ch==1)
    {for(i=0;i<521;i++)
      data[i]=buff1_data[i];
      v_index=CH1_V_Index;
    }
  else if(ch==2)
    {for(i=0;i<521;i++)
      data[i]=buff2_data[i];
```

```
    v_index=CH2_V_Index;
   }
  max=data[1];
  min=data[1];
  for(i=2;i<175;i++)
   {if(data[i]<min)
    min=data[i];
   if(data[i]>max)
    max=data[i];
   }
  i=(10*(max-min)*(V_cell[v_index]))>>4;
  if(ch==1)
   {CH1_MAX=max;CH1_MIN=min;CH1_PV=i;}
  else if(ch==2)
   {CH2_MAX=max;CH2_MIN=min;CH2_PV=i;}
}

void vv_measure(unsigned char ch)
{
  int i=0,f=0,num=0,vv=64,data[521],v_index;
  if(ch==1)
   {for(i=0;i<521;i++)
    data[i]=buff1_data[i];
    f=CH1_F;
    v_index=CH1_V_Index;
    }
  else if(ch==2)
   {for(i=0;i<521;i++)
    data[i]=buff2_data[i];
    f=CH2_F;
    v_index=CH2_V_Index;
    }
  if(f)
   {
    num=FS[T_Index]/f;
    vv=0;
    for(i=1;i<=num;i++)
      { if(i<512)
          vv=vv+(63-data[i])*(63-data[i]);
      }
    vv=63-sqrt((float)vv/num);
   }
  vv=40000*(64-vv)/(25*DIN[v_index]);
  if(ch==1)
   {CH1_TNUM=num;CH1_VV=vv;}
  else if(ch==2)
   {CH2_TNUM=num;CH2_VV=vv;}
}

void av_measure(unsigned char ch)
{
  int i=1,av=64,f,data[521],v_index,num;
  if(ch==1)
   {for(i=0;i<521;i++)
    data[i]=buff1_data[i];
    f=CH1_F;
    v_index=CH1_V_Index;
    num=CH1_TNUM;
    }
  else if(ch==2)
   {for(i=0;i<521;i++)
    data[i]=buff2_data[i];
    f=CH2_F;
    v_index=CH2_V_Index;
    num=CH2_TNUM;
    }
  if(f)
   {for(i=1;i<=num;i++)
     {if(i<512)
        av+=data[i];
```

```
      }
    av/=num;
   }
  av=400000*(64-av)/(25*DIN[v_index]);
  if(ch==1)
    CH1_AV=av;
  else if(ch==2)
    CH2_AV=av;
}

void measure(void)
{
  ft_measure(1);
  pv_measure(1);
  vv_measure(1);
  av_measure(1);
  buff1_data[513]=CH1_F;
  buff1_data[514]=CH1_T;
  buff1_data[515]=CH1_PV;
  buff1_data[516]=CH1_VV;
  buff1_data[517]=CH1_AV;
  buff1_data[518]=CH1_MAX;
  buff1_data[519]=CH1_MIN;
  buff1_data[520]=CH1_TNUM;
  ft_measure(2);
  pv_measure(2);
  vv_measure(2);
  av_measure(2);
  buff2_data[513]=CH2_F;
  buff2_data[514]=CH2_T;
  buff2_data[515]=CH2_PV;
  buff2_data[516]=CH2_VV;
  buff2_data[517]=CH2_AV;
  buff2_data[518]=CH2_MAX;
  buff2_data[519]=CH2_MIN;
  buff2_data[520]=CH2_TNUM;
}

void measure_disp(void)
{
 measure_Refresh(1);
 measure_Refresh(2);
 measure_Refresh(3);
}

void measure123_disp(unsigned char status)
{
  unsigned char status_mtype_flag,status_mch_flag,y1,y2;
  lcd_disp_hanzi(21,4,hz_clear,4);
  switch(status)
  {case 1:status_mtype_flag=status1_mtype_flag;
          status_mch_flag=status1_mch_flag;
          y1=36;
          y2=7;
          break;
   case 2:status_mtype_flag=status2_mtype_flag;
          status_mch_flag=status2_mch_flag;
          y1=68;
          y2=11;
          break;
   case 3:status_mtype_flag=status3_mtype_flag;
          status_mch_flag=status3_mch_flag;
          y1=100;
          y2=15;
          break;
  }
      if(shift_flag==1)
        {
          if(status_mtype_flag==5)
            status_mtype_flag=0;
```

```
        else
          status_mtype_flag++;
        if(status_mch_flag)
          { lcd_disp_string(20,y2-2,"CH2       ",10);
           lcd_disp_string(20,y2,"          ",10);
           if(ch_flag==1)
             lcd_disp_hanzi(21,y1,guanbi,4);
           else
             switch(status_mtype_flag)
               {case 0:lcd_disp_hanzi(21,y1,pinlv,4);
                  ch_f_disp(2,y2);
                  break;
                case 1:lcd_disp_hanzi(21,y1,zhouqi,4);
                 ch_t_disp(2,y2);
                   break;
                case 2:lcd_disp_hanzi(21,y1,fengfengzhi,4);
                  ch_pv_disp(2,y2);
                  break;
               case 3:lcd_disp_hanzi(21,y1,youxiaozhi,4);
                 ch_vv_disp(2,y2);
                   break;
               case 4:lcd_disp_hanzi(21,y1,pingjunzhi,4);
                 ch_av_disp(2,y2);
                   break;
               case 5:lcd_disp_hanzi(21,y1,wu,4);
                 lcd_disp_string(20,y2,"          ",10);
                   break;
               }
          }
        else
          {lcd_disp_string(20,y2-2,"CH1       ",10);
           lcd_disp_string(20,y2,"          ",10);
           if(ch_flag==2)
            lcd_disp_hanzi(21,y1,guanbi,4);
           else
             switch(status_mtype_flag)
               {case 0:lcd_disp_hanzi(21,y1,pinlv,4);
                     ch_f_disp(1,y2);
                     break;
               case 1:lcd_disp_hanzi(21,y1,zhouqi,4);
                      ch_t_disp(1,y2);
                     break;
               case 2:lcd_disp_hanzi(21,y1,fengfengzhi,4);
                      ch_pv_disp(1,y2);
                     break;
               case 3:lcd_disp_hanzi(21,y1,youxiaozhi,4);
                      ch_vv_disp(1,y2);
                     break;
               case 4:lcd_disp_hanzi(21,y1,pingjunzhi,4);
                      ch_av_disp(1,y2);
                     break;
               case 5:lcd_disp_hanzi(21,y1,wu,4);
                      lcd_disp_string(20,y2,"          ",10);
                     break;
               }
          }
      }
    else if(shift_flag==0)
        {
          status_mch_flag=~status_mch_flag;
          if(status_mch_flag)
            {lcd_disp_string(20,y2-2,"CH2       ",10);
             lcd_disp_string(20,y2,"          ",10);
             }
           else
             {lcd_disp_string(20,y2-2,"CH1       ",10);
              lcd_disp_string(20,y2,"          ",10);
              }
        }
  switch(status)
```

```
  {case 1:status1_mtype_flag=status_mtype_flag;
         status1_mch_flag=status_mch_flag;
         break;
   case 2:status2_mtype_flag=status_mtype_flag;
         status2_mch_flag=status_mch_flag;
         break;
   case 3:status3_mtype_flag=status_mtype_flag;
         status3_mch_flag=status_mch_flag;
         break;
  }
}

void cursor_disp(void)
{
  if(status1_ctype_flag)
   {
    lcd_disp_hanzi(21,36,dianya,2);
    lcd_disp_hanzi(25,36,zhengliang,2);
     V_disp(1);
     V_disp(2);
     V_disp(3);
   }
  else
  {
    lcd_disp_hanzi(21,36,shijian,2);
    lcd_disp_hanzi(25,36,zhengliang,2);
    T_disp(1);
    T_disp(2);
    T_disp(3);
  }
  lcd_disp_hanzi(23,68,guangbiao,2);
  lcd_disp_text(27,9,0x11);
  lcd_disp_hanzi(23,100,guangbiao,2);
  lcd_disp_text(27,13,0x12);
}

void cursor1_disp(void)
{int i=0;
  if(status1_ctype_flag==2)
    status1_ctype_flag=0;
  else
    status1_ctype_flag++;
  if(status1_ctype_flag==1||status1_ctype_flag==2)
    {
     lcd_disp_hanzi(21,36,dianya,2);
     lcd_disp_hanzi(25,36,zhengliang,2);
     for(i=0;i<16;i++)
        {
        lcd_disp_text(line_T1>>3,i,0x00);
        lcd_disp_text(line_T2>>3,i,0x00);
        }
     for(i=0;i<20;i++)
        {
        lcd_disp_text(i,line_V1>>3,(line_V1%8)|0xf0);
        lcd_disp_text(i,line_V2>>3,(line_V2%8)|0xf0);
        }
    }
  else
    {
     lcd_disp_hanzi(21,36,shijian,2);
     lcd_disp_hanzi(25,36,zhengliang,2);
     for(i=0;i<20;i++)
        {
        lcd_disp_text(i,line_V1>>3,0x00);
        lcd_disp_text(i,line_V2>>3,0x00);
        }
     for(i=0;i<16;i++)
        {
        lcd_disp_text(line_T1>>3,i,(line_T1%8)|0xf8);
        lcd_disp_text(line_T2>>3,i,(line_T2%8)|0xf8);
```

```
      }
   }
}

void cursor2_disp(void)
{int i=0;
 lcd_write(CGRAMSA>>11);lcd_write(0x00);lcd_ctrl(0x22);
 if(status1_ctype_flag)
   {
    if(shift_flag==0&&line_V1>1)
      {
       line_V1-=2;
       if((line_V1>>3)!=0)
         for(i=0;i<20;i++)
           {
            lcd_disp_text(i,line_V1>>3,(line_V1%8)|0xf0);
           }
      }
    else if(shift_flag==1&&line_V2>1)
          {
           line_V2-=2;
           if((line_V2>>3)!=0)
             for(i=0;i<20;i++)
               {
                lcd_disp_text(i,line_V2>>3,(line_V2%8)|0xf0);
               }
          }
   }
 else
   {
    if(shift_flag==0&&line_T1>1)
      {
       line_T1-=2;
       if((line_T1>>3)!=0)
         for(i=0;i<16;i++)
           {
            lcd_disp_text(line_T1>>3,i,(line_T1%8)|0xf8);
           }
      }
    else if(shift_flag==1&&line_T2>1)
         {
          line_T2-=2;
          if((line_T2>>3)!=0)
          for(i=0;i<16;i++)
           {
            lcd_disp_text(line_T2>>3,i,(line_T2%8)|0xf8);
           }
         }
   }
}

void cursor3_disp(void)
{int i=0;
 lcd_write(CGRAMSA>>11);lcd_write(0x00);lcd_ctrl(0x22);
 if(status1_ctype_flag)
   {
    if(shift_flag==0&&line_V1<126)
      {
       line_V1+=2;
       if((line_V1>>3)!=15)
         for(i=0;i<20;i++)
           {
            lcd_disp_text(i,line_V1>>3,(line_V1%8)|0xf0);
           }
      }
    else if(shift_flag==1&&line_V2<126)
          {
           line_V2+=2;
           if((line_V2>>3)!=15)
             for(i=0;i<20;i++)
```

```
                {
                 lcd_disp_text(i,line_V2>>3,(line_V2%8)|0xf0);
                }
           }
   }
  else
   {
    if(shift_flag==0&&line_T1<158)
      {
       line_T1+=2;
       if((line_T1>>3)!=19)
         for(i=0;i<15;i++)
           {
            lcd_disp_text(line_T1>>3,i,(line_T1%8)|0xf8);
           }
      }
    else if(shift_flag==1&&line_T2<158)
          {
           line_T2+=2;
           if((line_T1>>3)!=19)
           for(i=0;i<15;i++)
            {
             lcd_disp_text(line_T2>>3,i,(line_T2%8)|0xf8);
            }
          }
   }
}


void save_disp(void)
{SD_card_init();
 lcd_disp_hanzi(21,36,jiyixuanzhe,4);
 lcd_disp_string(20,7," flash 01",9);
 lcd_disp_hanzi(21,68,chunchu,4);
 lcd_disp_string(20,11,"          ",10);
 lcd_disp_hanzi(21,100,diaochu,4);
 lcd_disp_string(20,15,"          ",10);
}

void save1_disp(void)
{
  if(status1_stype_flag==20)
     status1_stype_flag=1;
  else
     status1_stype_flag++;
  if(status1_stype_flag<=10)
     {
     lcd_disp_string(20,7," flash ",7);
     lcd_disp_text(27,7,(status1_stype_flag/10+0x10));
     lcd_disp_text(28,7,(status1_stype_flag%10+0x10));
     }
  else
     {
     lcd_disp_string(20,7,"SDcard ",7);
     lcd_disp_text(27,7,(status1_stype_flag/10-1+0x10));
     lcd_disp_text(28,7,(status1_stype_flag%10+0x10));
     }
}

void save2_disp(void)
{
  alt_flash_fd* fd;
  flash_region* regions;
  int number_of_regions,i=0,j=0;
  int data[1042],data1[521]={0},data2[521]={0};
  unsigned char sd_buff1[512],sd_buff2[512];
  if((ch_flag==1)||(ch_flag==3))
    {
     for(i=0;i<521;i++)
       data1[i]=buff1_data[i];
```

```
    }
  if((ch_flag==2)||(ch_flag==3))
    {
    for(i=0;i<521;i++)
      data2[i]=buff2_data[i];
    }
  for(i=0;i<521;i++)
    data[i]=data1[i];
  for(;i<1042;i++)
    data[i]=data2[i-521];

  if(status1_stype_flag<=10)
  {
    fd = alt_flash_open_dev(CFI_FLASH_NAME);
    alt_get_flash_info(fd, &regions, &number_of_regions);
    alt_write_flash(fd,flash_addr[status1_stype_flag],data,4096);
    alt_flash_close_dev(fd);
  }
  else
  {
    for(i=0;i<472;i++)
      sd_buff1[i]=buff1_data[i]&0xff;
    for(j=512;j<522;j++)
      {
        sd_buff1[i++]=buff1_data[j]&0xff;
        sd_buff1[i++]=buff1_data[j]>>8&0xff;
        sd_buff1[i++]=buff1_data[j]>>16&0xff;
        sd_buff1[i++]=buff1_data[j]>>24&0xff;
      }

    for(i=0;i<472;i++)
      sd_buff2[i]=buff2_data[i]&0xff;
    for(j=512;j<522;j++)
      {
        sd_buff2[i++]=buff2_data[j]&0xff;
        sd_buff2[i++]=buff2_data[j]>>8&0xff;
        sd_buff2[i++]=buff2_data[j]>>16&0xff;
        sd_buff2[i++]=buff2_data[j]>>24&0xff;
      }
    SD_write_lba(sd_buff1,sd_addr[status1_stype_flag-11],1);
    SD_write_lba(sd_buff2,sd_addr[status1_stype_flag-11]+1,1);
  }
  run_stop_flag=0;
  IOWR(RUN_STOP_BASE,0,0);
  lcd_disp_string(20,1,"   STOP  ",8);
}

void save3_disp(void)
{
  alt_flash_fd* fd;
  flash_region* regions;
  int number_of_regions,i=0,j=0;
  unsigned int data[1042];
  unsigned char sd_buff1[512]={0},sd_buff2[512]={0};
  run_stop_flag=0;
  IOWR(RUN_STOP_BASE,0,0);
  lcd_disp_string(20,1,"   STOP  ",8);
  if(status1_stype_flag<=10)
  {
  fd = alt_flash_open_dev(CFI_FLASH_NAME);
  alt_get_flash_info(fd, &regions, &number_of_regions);
  alt_read_flash(fd,flash_addr[status1_stype_flag],data,4096);
  alt_flash_close_dev(fd);
    for(i=0;i<521;i++)
      buff1_data[i]=data[i];

    for(;i<1042;i++)
        buff2_data[i-521]=data[i];
  }
  else
```

```
  {
  SD_read_lba(sd_buff1,sd_addr[status1_stype_flag-11],1);
  SD_read_lba(sd_buff2,sd_addr[status1_stype_flag-11]+1,1);
  for(i=0;i<472;i++)
    {
      buff1_data[i]=sd_buff1[i]&0x0ff;
      buff2_data[i]=sd_buff2[i]&0x0ff;
    }
  for(j=512;j<522;j++)
    {
      buff1_data[j]=sd_buff1[i]|((sd_buff1[i+1]<<8)&0x0ff00)|((sd_buff1[i+2]<<16)&0x0ff0000)|((sd_buff1[i+3]<<24)&0xff000000);
      buff2_data[j]=sd_buff2[i]|((sd_buff2[i+1]<<8)&0x0ff00)|((sd_buff2[i+2]<<16)&0x0ff0000)|((sd_buff2[i+3]<<24)&0xff000000);
      i+=4;
    }
  }
    CH1_F=buff1_data[513];
    CH1_T=buff1_data[514];
    CH1_PV=buff1_data[515];
    CH1_VV=buff1_data[516];
    CH1_AV=buff1_data[517];
    CH1_MAX=buff1_data[518];
    CH1_MIN=buff1_data[519];
    CH1_TNUM=buff1_data[520];
    CH2_F=buff2_data[513];
    CH2_T=buff2_data[514];
    CH2_PV=buff2_data[515];
    CH2_VV=buff2_data[516];
    CH2_AV=buff2_data[517];
    CH2_MAX=buff2_data[518];
    CH2_MIN=buff2_data[519];
    CH2_TNUM=buff2_data[520];
  Display_Wave();
}

void single_trigger(void)
{int j=0;
    if(run_stop_flag)
      {
       IOWR(BUFF1_START_BASE,0,1);
       usleep(1);
       IOWR(BUFF1_START_BASE,0,0);
       usleep(1);
       IOWR(BUFF2_START_BASE,0,1);
       usleep(1);
       IOWR(BUFF2_START_BASE,0,0);
       usleep(2000);
        for( j=1;j<512;j++)
          {
            IOWR(BUFF1_RDADDR_BASE,0,j);
            buff1_data[j]=127-(IORD(BUFF1_DATA_BASE,0)>>1);
          }
        for( j=1;j<512;j++)
          {
            IOWR(BUFF2_RDADDR_BASE,0,j);
            buff2_data[j]=127-(IORD(BUFF2_DATA_BASE,0)>>1);
          }
      }
      run_stop_flag=0;
      IOWR(RUN_STOP_BASE,0,0);
      lcd_disp_string(20,0,"   STOP ",8);
}

void continue_trigger(void)
{int j=0;
    if(run_stop_flag)
      {
       IOWR(BUFF1_START_BASE,0,1);
       usleep(1);
       IOWR(BUFF1_START_BASE,0,0);
       usleep(1);
```

```
       for( j=1;j<512;j++)
         {
          IOWR(BUFF1_RDADDR_BASE,0,j);
          buff1_data[j]=127-(IORD(BUFF1_DATA_BASE,0)>>1);
         }
     }
        if(run_stop_flag)
     {
      IOWR(BUFF2_START_BASE,0,1);
      usleep(1);
      IOWR(BUFF2_START_BASE,0,0);
      usleep(1);
       for( j=1;j<512;j++)
         {
          IOWR(BUFF2_RDADDR_BASE,0,j);
          buff2_data[j]=127-(IORD(BUFF2_DATA_BASE,0)>>1);
         }
     }
}

void trigger1_disp(void)
{
   if((shift_flag!=0)||(status3_ttype_flag))
   {
    if(ch_flag==3)
       ch_flag=1;
    else
       ch_flag++;
    if(ch_flag==1)
       lcd_disp_string(20,7,"    CH1   ",10);
    else if(ch_flag==2)
          lcd_disp_string(20,7,"    CH2   ",10);
        else if(ch_flag==3)
             lcd_disp_string(20,7," CH1  CH2 ",10);
   }
   else
     {
      if(trigger_level>30)
        trigger_level-=20;
     IOWR(TRIGER1_LEVEL_BASE,0,trigger_level);
     lcd_disp_string(23,15,CHUFA[trigger_level/20],3);
     }
}

void trigger2_disp(void)
{
   if((shift_flag!=0)||(status3_ttype_flag))
   {
    status2_ttype_flag=~status2_ttype_flag;
    if(status2_ttype_flag)
      lcd_disp_hanzi(21,68,lianxu,2);
    else
      lcd_disp_hanzi(21,68,danci,2);
   }
   else
     {
     if(trigger_level<220)
       trigger_level+=20;
     IOWR(TRIGER1_LEVEL_BASE,0,trigger_level);
     IOWR(TRIGER2_LEVEL_BASE,0,trigger_level);
     lcd_disp_string(23,15,CHUFA[trigger_level/20],3);
     }
}

void trigger3_disp(void)
{
    status3_ttype_flag=~status3_ttype_flag;
    if(status3_ttype_flag)
      {
      lcd_disp_hanzi(21,100,bianyan,2);
```

```
      IOWR(TRIGER1_SEL_BASE,0,0);
      IOWR(TRIGER2_SEL_BASE,0,0);
      }
    else
      {
       lcd_disp_hanzi(21,100,dianping,2);
       IOWR(TRIGER1_SEL_BASE,0,1);
       IOWR(TRIGER2_SEL_BASE,0,1);
      }
}

void fft0_disp(void)
{
         lcd_disp_string(20,0,"         ",10);
         lcd_disp_string(20,1,"         ",10);
         lcd_disp_string(20,2,"         ",10);
         lcd_disp_string(20,3,"         ",10);
         lcd_disp_hanzi(21,4,pinpufenxi,4);
         if(shift_flag)
           lcd_disp_string(20,3,"    CH2   ",10);
         else
           lcd_disp_string(20,3,"    CH1   ",10);
         if(status3_fft_flag)
           lcd_disp_hanzi(21,100,quxiao,4);
         else
           lcd_disp_hanzi(21,100,queren,4);
}

void fft3_disp(void)
{
 if(status3_fft_flag==1)
   status3_fft_flag=0;
 else
   status3_fft_flag++;
 if(status3_fft_flag)
  {
   lcd_disp_hanzi(21,100,quxiao,4);
   fft();
   fft_disp();
  }
 else
  {
   lcd_disp_hanzi(21,100,queren,4);
   lcd_disp_text(1,1,0x00);
   lcd_disp_text(3,1,0x00);
   lcd_disp_text(5,1,0x00);
   lcd_disp_text(7,1,0x00);
   lcd_disp_text(9,1,0x00);
   lcd_disp_text(11,1,0x00);
   lcd_disp_text(13,1,0x00);
   lcd_disp_text(15,1,0x00);
   lcd_disp_text(17,1,0x00);
   lcd_disp_string(20,4,"         ",10);
   lcd_disp_string(20,5,"         ",10);
   lcd_disp_string(20,6,"         ",10);
   lcd_disp_string(20,7,"         ",10);
   lcd_disp_string(20,8,"         ",10);
   lcd_disp_string(20,9,"         ",10);
   lcd_disp_string(20,10,"         ",10);
   lcd_disp_string(20,11,"         ",10);
   Display_Wave();
  }
}

void zice_disp(void)
{
  int i=0,j=0;
   for(i=0;i<20;i++)
     for(j=0;j<16;j++)
       lcd_disp_text(i,j,0x00);
```

```
  for( i=0;i<20;i++)
   for(j=0;j<128;j++)
     {
      lcd_set_graph(i,j);
      lcd_disp8(0x00);
     }
lcd_disp_string(20,0,"          ",10);
lcd_disp_string(20,1,"          ",10);
lcd_disp_string(20,2,"          ",10);
lcd_disp_string(20,3,"          ",10);
 Draw_hline(0,0,20);
 Draw_hline(0,127,20);
 Draw_sline(0,0,127);
 lcd_disp_hanzi(21,4,yuzhipinlv,4);
 lcd_disp_string(22,3,zice_f[status0_zice_flag],6);
 lcd_disp_hanzi(21,36,yuzhixiangwei,4);
 lcd_disp_string(22,7,zice_x[status1_zice_flag],4);
 if(status2_zice_flag==0)
  lcd_disp_hanzi(21,68,zhengxuanbo,4);
 else if(status2_zice_flag==1)
       lcd_disp_hanzi(21,68,fangbo,4);
      else if(status2_zice_flag==2)
            lcd_disp_hanzi(21,68,sanjiaobo,4);
           else if(status2_zice_flag==3)
                 lcd_disp_hanzi(21,68,bianjiboxing,4);
 if(status2_zice_flag==3)
   {
   Draw_hline(0,32,20);
   lcd_disp_hanzi(1,8,jibo,2);
   lcd_disp_string(5,2,":1",2);
   lcd_disp_hanzi(1,36,xiebo,2);
   lcd_disp_text(5,5,0x12);
   lcd_disp_string(6,5,zice_pv[zice_pv2],4);
   lcd_disp_hanzi(11,36,xiebo,2);
   lcd_disp_text(15,5,0x13);
   lcd_disp_string(16,5,zice_pv[zice_pv3],4);
   lcd_disp_hanzi(1,60,xiebo,2);
   lcd_disp_text(5,8,0x14);
   lcd_disp_string(6,8,zice_pv[zice_pv4],4);
   lcd_disp_hanzi(11,60,xiebo,2);
   lcd_disp_text(15,8,0x15);
   lcd_disp_string(16,8,zice_pv[zice_pv5],4);
   lcd_disp_hanzi(1,84,xiebo,2);
   lcd_disp_text(5,11,0x16);
   lcd_disp_string(6,11,zice_pv[zice_pv6],4);
   lcd_disp_hanzi(11,84,xiebo,2);
   lcd_disp_text(15,11,0x17);
   lcd_disp_string(16,11,zice_pv[zice_pv7],4);
   lcd_disp_hanzi(1,108,xiebo,2);
   lcd_disp_text(5,14,0x18);
   lcd_disp_string(6,14,zice_pv[zice_pv8],4);
   lcd_disp_hanzi(11,108,xiebo,2);
   lcd_disp_text(15,14,0x19);
   lcd_disp_string(16,14,zice_pv[zice_pv9],4);
   }
   else
   {
   Draw_hline(0,64,20);
   lcd_disp_hanzi(1,8,tongdao1,3);
   lcd_disp_hanzi(1,32,fengfengzhi,4);
   lcd_disp_string(9,5,zice_pv[zice_pv1],4);
   lcd_disp_hanzi(1,72,tongdao2,3);
   lcd_disp_hanzi(1,96,fengfengzhi,4);
   lcd_disp_string(9,13,zice_pv[zice_pv2],4);
   }
 if(status3_zice_flag==1)
   lcd_disp_hanzi(21,100,queren,4);
 else
   lcd_disp_hanzi(21,100,quxiao,4);
}
```

```
void zice0_disp(void)
{
 if(status0_zice_flag==14)
    status0_zice_flag=0;
 else
    status0_zice_flag++;
 lcd_disp_string(22,3,zice_f[status0_zice_flag],6);
}

void zice1_disp(void)
{
 if(status1_zice_flag==8)
    status1_zice_flag=0;
 else
    status1_zice_flag++;
 lcd_disp_string(22,7,zice_x[status1_zice_flag],4);
}

void zice2_disp(void)
{int i=0,j=0;

   for(i=0;i<20;i++)
     for(j=0;j<16;j++)
        lcd_disp_text(i,j,0x00);
   for( i=0;i<20;i++)
    for(j=0;j<128;j++)
      {
       lcd_set_graph(i,j);
       lcd_disp8(0x00);
      }
  if(status2_zice_flag==3)
    status2_zice_flag=0;
  else
    status2_zice_flag++;
  if(status2_zice_flag==0)
   lcd_disp_hanzi(21,68,zhengxuanbo,4);
  else if(status2_zice_flag==1)
        lcd_disp_hanzi(21,68,fangbo,4);
       else if(status2_zice_flag==2)
             lcd_disp_hanzi(21,68,sanjiaobo,4);
            else if(status2_zice_flag==3)
              lcd_disp_hanzi(21,68,bianjiboxing,4);
  Draw_hline(0,0,20);
  Draw_hline(0,127,20);
  Draw_sline(0,0,127);
  if(status2_zice_flag==3)
    {
    Draw_hline(0,32,20);
    lcd_disp_hanzi(1,8,jibo,2);
    lcd_disp_string(5,2,":1",2);
    lcd_disp_hanzi(1,36,xiebo,2);
    lcd_disp_text(5,5,0x12);
    lcd_disp_string(6,5,zice_pv[zice_pv2],4);
    lcd_disp_hanzi(11,36,xiebo,2);
    lcd_disp_text(15,5,0x13);
    lcd_disp_string(16,5,zice_pv[zice_pv3],4);
    lcd_disp_hanzi(1,60,xiebo,2);
    lcd_disp_text(5,8,0x14);
    lcd_disp_string(6,8,zice_pv[zice_pv4],4);
    lcd_disp_hanzi(11,60,xiebo,2);
    lcd_disp_text(15,8,0x15);
    lcd_disp_string(16,8,zice_pv[zice_pv5],4);
    lcd_disp_hanzi(1,84,xiebo,2);
    lcd_disp_text(5,11,0x16);
    lcd_disp_string(6,11,zice_pv[zice_pv6],4);
    lcd_disp_hanzi(11,84,xiebo,2);
    lcd_disp_text(15,11,0x17);
    lcd_disp_string(16,11,zice_pv[zice_pv7],4);
    lcd_disp_hanzi(1,108,xiebo,2);
```

```
   lcd_disp_text(5,14,0x18);
   lcd_disp_string(6,14,zice_pv[zice_pv8],4);
   lcd_disp_hanzi(11,108,xiebo,2);
   lcd_disp_text(15,14,0x19);
   lcd_disp_string(16,14,zice_pv[zice_pv9],4);
   }
  else
   {
   Draw_hline(0,64,20);
   lcd_disp_hanzi(1,8,tongdao1,3);
   lcd_disp_hanzi(1,32,fengfengzhi,4);
   lcd_disp_string(9,5,zice_pv[zice_pv1],4);
   lcd_disp_hanzi(1,72,tongdao2,3);
   lcd_disp_hanzi(1,96,fengfengzhi,4);
   lcd_disp_string(9,13,zice_pv[zice_pv2],4);
   }
}

void zice3_disp(void)
{int i,j;
  if(status3_zice_flag==1)
    status3_zice_flag=0;
  else
    status3_zice_flag++;
   lcd_ctrl(0x9c);
  if(status3_zice_flag==1)
    lcd_disp_hanzi(21,100,queren,4);
  else
    {
     lcd_disp_hanzi(21,100,quxiao,4);
     zice_m[0]=zice_pv_cell[zice_pv1];
     zice_m[1]=zice_pv_cell[zice_pv2];
     zice_m[2]=zice_pv_cell[zice_pv2];
     zice_m[3]=zice_pv_cell[zice_pv3];
     zice_m[4]=zice_pv_cell[zice_pv4];
     zice_m[5]=zice_pv_cell[zice_pv5];
     zice_m[6]=zice_pv_cell[zice_pv6];
     zice_m[7]=zice_pv_cell[zice_pv7];
     zice_m[8]=zice_pv_cell[zice_pv8];
     zice_m[9]=zice_pv_cell[zice_pv9];
      IOWR(DDS_F_BASE,0,DIN_zice_f[status0_zice_flag]);
      IOWR(DDS_P_BASE,0,DIN_zice_x[status1_zice_flag]);
  if(status2_zice_flag==0)
    {
     for(i=0;i<512;i++)
      {j=127*zice_m[0]*(sin(2*3.14*i/512)+1);
       IOWR(DDS_WRADD1_BASE,0,i);
       IOWR(DDS_DB1_BASE,0,j);
       j=127*zice_m[1]*(sin(2*3.14*i/512)+1);
       IOWR(DDS_WRADD2_BASE,0,i);
       IOWR(DDS_DB2_BASE,0,j);
      }
    }
  if(status2_zice_flag==1)
    {
     for(i=0;i<256;i++)
      {
       IOWR(DDS_WRADD1_BASE,0,i);
       IOWR(DDS_DB1_BASE,0,0);
       IOWR(DDS_WRADD2_BASE,0,i);
       IOWR(DDS_DB2_BASE,0,0);
      }
       j=255*zice_m[0];
     for(i=256;i<512;i++)
      {
       IOWR(DDS_WRADD1_BASE,0,i);
       IOWR(DDS_DB1_BASE,0,j);
      }
     j=255*zice_m[1];
     for(i=256;i<512;i++)
```

```
     {
      IOWR(DDS_WRADD2_BASE,0,i);
      IOWR(DDS_DB2_BASE,0,j);
     }
    }
  if(status2_zice_flag==2)
    {
     for(i=0;i<256;i++)
      {
       IOWR(DDS_WRADD1_BASE,0,i);
       IOWR(DDS_DB1_BASE,0,i*zice_m[0]);
       IOWR(DDS_WRADD2_BASE,0,i);
       IOWR(DDS_DB2_BASE,0,i*zice_m[1]);
      }
     IOWR(DDS_WRADD1_BASE,0,256);
     IOWR(DDS_DB1_BASE,0,255*zice_m[0]);
     IOWR(DDS_WRADD2_BASE,0,256);
     IOWR(DDS_DB2_BASE,0,255*zice_m[1]);
     for(i=257;i<512;i++)
       {
       IOWR(DDS_WRADD1_BASE,0,i);
       IOWR(DDS_DB1_BASE,0,(512-i)*zice_m[0]);
       IOWR(DDS_WRADD2_BASE,0,i);
       IOWR(DDS_DB2_BASE,0,(512-i)*zice_m[1]);
       }
    }
  if(status2_zice_flag==3)
    {
      for(i=0;i<512;i++)
      {

j=127*(1+sin(2*3.14*i/512)+zice_m[2]*sin(2*2*3.14*i/512)+zice_m[3]*sin(2*3*3.14*i/512)+zice_m[4]*sin(2*4*3.14*i/512)+zice_
m[5]*sin(5*2*3.14*i/512)+zice_m[6]*sin(6*2*3.14*i/512)+zice_m[7]*sin(7*2*3.14*i/512)+zice_m[8]*sin(8*2*3.14*i/512)+zice_m
[9]*sin(9*2*3.14*i/512));
       IOWR(DDS_WRADD1_BASE,0,i);
       IOWR(DDS_DB1_BASE,0,j);
       IOWR(DDS_WRADD2_BASE,0,i);
       IOWR(DDS_DB2_BASE,0,j);
      }
    }
  }
}
```

3、Lcd□□□□□:

```
int lcd_busy(void)
{
  int lcd_control=0;
  IOWR(LCD_DB_BASE,1,0);
  IOWR(LCD_CD_BASE,0,1);
  IOWR(LCD_CS_BASE,0,0);
  IOWR(LCD_RD_BASE,0,0);
  lcd_control=IORD(LCD_DB_BASE,0);
  IOWR(LCD_RD_BASE,0,1);
  IOWR(LCD_CS_BASE,0,1);
  return(lcd_control);
}

void lcd_st01(void)
{
  while((lcd_busy()&3)!=3){;}
}

void lcd_st3(void)
{
  while ((lcd_busy()&8)!=8){;}
}
```

```
void lcd_ctrl(int dat)
{
  lcd_st01();
  IOWR(LCD_DB_BASE,1,0xff);
  IOWR(LCD_CD_BASE,0,1);
  IOWR(LCD_CS_BASE,0,0);
  IOWR(LCD_WR_BASE,0,0);
  IOWR(LCD_DB_BASE,0,dat);
  IOWR(LCD_WR_BASE,0,1);
  IOWR(LCD_CS_BASE,0,1);
}

void lcd_write(int dat)
{
  lcd_st01();
  IOWR(LCD_DB_BASE,1,0xff);
  IOWR(LCD_CD_BASE,0,0);
  IOWR(LCD_CS_BASE,0,0);
  IOWR(LCD_WR_BASE,0,0);
  IOWR(LCD_DB_BASE,0,dat);
  IOWR(LCD_WR_BASE,0,1);
  IOWR(LCD_CS_BASE,0,1);
}


void lcd_initial()
{
  IOWR(LCD_CS_BASE,0,1);
  IOWR(LCD_RD_BASE,0,1);
  IOWR(LCD_WR_BASE,0,1);
  IOWR(LCD_CD_BASE,0,1);

  lcd_write(TASAL);lcd_write(TASAH);lcd_ctrl(0x40);
  lcd_write(ZIFU_NUM);lcd_write(0x00);lcd_ctrl(0x41);
  lcd_write(GASAL);lcd_write(GASAH);lcd_ctrl(0x42);
  lcd_write(ZIFU_NUM);lcd_write(0x00);lcd_ctrl(0x43);
  lcd_write(CGRAMSA>>11);lcd_write(0x00);lcd_ctrl(0x22);
  lcd_ctrl(0x80);
  lcd_ctrl(0x9C);
  lcd_ctrl(0xa0);
  lcd_clear_ram();
}

void lcd_clear_ram()
{
  int i;
  lcd_write(0x00);lcd_write(0x00);lcd_ctrl(0x24);
  lcd_ctrl(0xb0);
  for(i=0;i<8192;i++)
  {
     lcd_autowrite(0x00);
  }
  lcd_ctrl(0xb2);
}

void lcd_image()
{
  int i;
  lcd_write(0x00);lcd_write(0x00);lcd_ctrl(0x24);
  lcd_ctrl(0xb0);
  for(i=0;i<4096;i++)
  {
     lcd_autowrite(image[i]);
  }
  lcd_ctrl(0xb2);
}

void lcd_set_graph(unsigned char x,unsigned char y)
{
```

```
  unsigned int xy;
  xy=(GASAH<<8)+GASAL+y*ZIFU_NUM+x;lcd_write(xy&0xff);
  xy=xy>>8;
  lcd_write(xy);lcd_ctrl(0x24);
}


void lcd_disp8(int dat)
{ lcd_write(dat);
  lcd_ctrl(0xc0);
}

void lcd_disp1608(unsigned int x,unsigned int y,unsigned char n)
{
  unsigned char i;  unsigned int j;
  for(i=0;i<16;i++)
  {
     lcd_set_graph(x,y+i);
     j=n*16+i;
     lcd_disp8(asc8[j]);
  }
}

void lcd_set_text(unsigned char x,unsigned char y)
{
 unsigned int xy;
 xy=(TASAH<<8)+TASAL+y*ZIFU_NUM+x;lcd_write(xy&0xff);
 xy=xy>>8;lcd_write(xy);lcd_ctrl(0x24);
}

void lcd_disp_text(unsigned char x,unsigned char y,unsigned char ccode)
{
 lcd_set_text(x,y);
 lcd_write(ccode);lcd_ctrl(0xc4);
}
void lcd_disp_char(unsigned char x,unsigned char y,unsigned char ccode)
{
 lcd_set_text(x,y);
 lcd_write(ccode-0x20);lcd_ctrl(0xc4);
}

void lcd_disp_string(unsigned char x,unsigned char y,unsigned char *string,unsigned int len)
{
  int i;
  for(i=0;i<len;i++)
    lcd_disp_char(x+i,y,string[i]);
}

void lcd_disp_hanzi(unsigned char x,unsigned char y,unsigned char *string,unsigned int len)
{
  unsigned int i,j,m_Addr;
  for(i=0;i<len;i++)
    {
      m_Addr=(GASAH<<8)+GASAL+ZIFU_NUM*y+x+(i<<1);
      for(j=0;j<32;)
        {
          lcd_write(m_Addr&0xff);
          lcd_write(m_Addr>>8);
          lcd_ctrl(0x24);
          lcd_write(string[(i<<5)+j++]);
          lcd_ctrl(0xc0);
          lcd_write(string[(i<<5)+j++]);
          lcd_ctrl(0xc4);
          m_Addr+=ZIFU_NUM;
        }
    }

}

void Init_CGRAM(void)
```

```
{
  int i;
  lcd_write(CGRAMSA&0xff);
  lcd_write(CGRAMSA>>8);
  lcd_ctrl(0x24);
  lcd_ctrl(0xb0);
  for(i=0;i<512;i++)
    lcd_autowrite(mCGRAM[i]);
  lcd_ctrl(0xb2);
}

void lcd_autowrite(int dat)
{
  lcd_st3();
  IOWR(LCD_DB_BASE,1,0xff);
  IOWR(LCD_CD_BASE,0,0);
  IOWR(LCD_CS_BASE,0,0);
  IOWR(LCD_WR_BASE,0,0);
  IOWR(LCD_DB_BASE,0,dat);
  IOWR(LCD_WR_BASE,0,1);
  IOWR(LCD_CS_BASE,0,1);
}

void Draw_Grid(void)
{
   unsigned int i,j;
   lcd_disp_text(0,0,0xc0);
   for(i=1;i<18;)
   {
     lcd_disp_text(i++,0,0xc1);
     lcd_disp_text(i++,0,0xc2);
   }
   lcd_disp_text(19,0,0xc3);
   for(j=1;j<15;j+=2)
   {
     lcd_disp_text(0,j,0xc5);
     for(i=2;i<19;i+=2)
       lcd_disp_text(i,j,0xc6);
     lcd_disp_text(19,j,0xc7);
   }
   for(j=2;j<15;j+=2)
   {
     lcd_disp_text(0,j,0xc8);
     for(i=1;i<19;)
     {
       lcd_disp_text(i++,j,0xc9);
       lcd_disp_text(i++,j,0xca);
     }
     lcd_disp_text(19,j,0xcb);
   }
   lcd_disp_text(0,15,0xcc);
   for(i=1;i<19;)
   {
     lcd_disp_text(i++,15,0xcd);
     lcd_disp_text(i++,15,0xce);
   }
   lcd_disp_text(19,15,0xcf);
   for(i=1;i<19;)
     {
       lcd_disp_text(i,7,0xd2);
       lcd_disp_text(i++,8,0xd5);
       lcd_disp_text(i,7,0xd1);
       lcd_disp_text(i++,8,0xd6);
     }
   lcd_disp_text(0,7,0xd3);
   lcd_disp_text(19,7,0xd4);
   lcd_disp_text(0,8,0xd7);
   lcd_disp_text(19,8,0xd8);
   for(j=1;j<15;)
     {
```

```
      lcd_disp_text(9,j,0xd9);
      lcd_disp_text(10,j++,0xdd);
      lcd_disp_text(9,j,0xda);
      lcd_disp_text(10,j++,0xde);
     }
  lcd_disp_text(9,0,0xdb);
  lcd_disp_text(9,15,0xdc);
  lcd_disp_text(10,0,0xdf);
  lcd_disp_text(10,15,0xe0);
  lcd_disp_text(9,7,0xe1);
  lcd_disp_text(9,8,0xe2);
  lcd_disp_text(10,7,0xe3);
  lcd_disp_text(10,8,0xe4);
  Draw_hline(20,0,10);
  Draw_hline(20,32,10);
  Draw_hline(20,64,10);
  Draw_hline(20,96,10);
  Draw_hline(20,127,10);
  Draw_sline(160,0,128);
  Draw_sline(239,0,128);
 lcd_disp_text(28,0,(shift_flag+0x10));
 run_stop_flag=255;
 IOWR(RUN_STOP_BASE,0,1);
 lcd_disp_string(23,0,"RUN ",4);
 lcd_disp_string(20,1," M  ",4);
 lcd_disp_string(24,1,TDIV[T_Index],5);
 lcd_disp_string(20,2,"CH1 ",4);
 lcd_disp_string(24,2,VDIV[CH1_V_Index],5);
 lcd_disp_string(20,3,"CH2 ",4);
 lcd_disp_string(24,3,VDIV[CH2_V_Index],5);
 lcd_disp_string(20,5,"CH1",3);
 lcd_disp_hanzi(21,36,pinlv,4);
 lcd_disp_string(26,7,"0Hz",3);
 lcd_disp_string(20,9,"CH1",3);
 lcd_disp_hanzi(21,68,fengfengzhi,4);
 lcd_disp_string(26,11,"0mV",3);
 lcd_disp_string(20,13,"CH1",3);
 lcd_disp_hanzi(21,100,pingjunzhi,4);
 lcd_disp_string(26,15,"0mV",3);
}

void Display_Wave(void)
{
  unsigned int i,j,k;
  int m1=0,n1=0,m2=0,n2=0;
  unsigned int Temp1,Temp2;
  unsigned int m_Addr;
  unsigned int disp_data[128]={0},disp1_data[128]={0},disp2_data[128]={0};
 if((ch_flag==1)||(ch_flag==3))
   {
   if(T_addr<0)
     m1=0;
   else
     m1=buff1_data[1+T_addr]+V_data1;
   if(m1<0)
     m1=0;
   else if(m1>127)
         m1=127;
   }
 if((ch_flag==2)||(ch_flag==3))
   {
   if(T_addr<0)
     m2=0;
   else
     m2=buff2_data[1+T_addr]+V_data2;
   if(m2<0)
     m2=0;
   else if(m2>127)
         m2=127;
   }
```

```
 for(i=0;i<20;i++)
  {
   if((ch_flag==1)||(ch_flag==3))
   { Temp1= 0x80;
     for(k=0;k<128;k++)
        disp1_data[k]=0;
     for(j=0;j<8;j++)
     {
      if((j+2+8*i+T_addr)<0||(j+2+8*i+T_addr)>511)
       n1=0;
      else
       n1=buff1_data[j+2+8*i+T_addr]+V_data1;
      if(n1<0)
       n1=0;
      else if(n1>127)
            n1=127;
      for(k=m1;k<=n1;k++)
        disp1_data[k] |= Temp1;
      for(k=n1;k<=m1;k++)
        disp1_data[k] |= Temp1;
      m1=n1;
      Temp1>>= 1;
     }
   }
   if((ch_flag==2)||(ch_flag==3))
   { Temp2= 0x80;
     for(k=0;k<128;k++)
        disp2_data[k]=0;
     for(j=0;j<8;j++)
     {
      if((j+2+8*i+T_addr)<0||(j+2+8*i+T_addr)>511)
       n2=0;
      else
       n2=buff2_data[j+2+8*i+T_addr]+V_data2;
      if(n2<0)
        n2=0;
      else if(n2>127)
            n2=127;
      for(k=m2;k<=n2;k++)
         disp2_data[k] |= Temp2;
      for(k=n2;k<=m2;k++)
         disp2_data[k] |= Temp2;
      m2=n2;
      Temp2>>= 1;
     }
   }
     if(ch_flag==1)
       for(k=0;k<128;k++)
         disp_data[k]=disp1_data[k];
     else if(ch_flag==2)
           for(k=0;k<128;k++)
             disp_data[k]=disp2_data[k];
          else if(ch_flag==3)
                for(k=0;k<128;k++)
                  disp_data[k]=(disp1_data[k])|(disp2_data[k]);
     m_Addr=(GASAH<<8)+GASAL+i;
     for(k=0;k<128;k++)
     {
       lcd_write(m_Addr&0xff);
       lcd_write(m_Addr>>8);
       lcd_ctrl(0x24);
       lcd_write(disp_data[127-k]);
       lcd_ctrl(0xc4);
       m_Addr+=ZIFU_NUM;
     }
  }
}

void Draw_hline(unsigned char x,unsigned char y,unsigned char len)
{ int i;
```

```
  lcd_set_graph(x,y);
  for(i=0;i<len;i++)
    {
     lcd_disp8(0xff);
    }
}

void Draw_sline(unsigned char x,unsigned char y,unsigned char len)
{ int i;
  for(i=0;i<len;i++)
    { lcd_set_graph(x>>3,y+i);
      lcd_ctrl(0xf8|~(x%8));
    }
}

void ch_f_disp(unsigned char ch,unsigned char y)
{
  int data;
  if(ch==1)
    data=CH1_F;
  else if(ch==2)
       data=CH2_F;
   lcd_disp_string(20,y,"       Hz ",10);
if(data/10000000)
  {
   lcd_disp_text(22,y,(data/10000000+0x10));//
   lcd_disp_text(23,y,(data/1000000%10+0x10));//
   lcd_disp_text(24,y,0x0e);//
   lcd_disp_text(25,y,(data/100000%10+0x10));//
   lcd_disp_char(26,y,'M');//M
  }
else if(data/1000000)
     {
      lcd_disp_text(23,y,(data/1000000+0x10));//
      lcd_disp_text(24,y,0x0e);//.
      lcd_disp_text(25,y,(data/100000%10+0x10));//
      lcd_disp_text(26,y,'M');//M
      }
    else if(data/100000)
      {
       lcd_disp_text(21,y,(data/100000+0x10));
       lcd_disp_text(22,y,(data/10000%10+0x10));
       lcd_disp_text(23,y,(data/1000%10+0x10));
       lcd_disp_text(24,y,0x0e);
       lcd_disp_text(25,y,(data/100%10+0x10));
       lcd_disp_text(26,y,0x4b);//k
      }
    else if(data/10000)
         {
          lcd_disp_text(21,y,(data/10000+0x10));
          lcd_disp_text(22,y,(data/1000%10+0x10));
          lcd_disp_text(23,y,0x0e);
          lcd_disp_text(24,y,(data/100%10+0x10));
          lcd_disp_text(25,y,(data/10%10+0x10));
          lcd_disp_text(26,y,0x4b);//k
         }
        else if(data/1000)
             {
              lcd_disp_text(21,y,(data/1000+0x10));
              lcd_disp_text(22,y,0x0e);
              lcd_disp_text(23,y,(data/100%10+0x10));
              lcd_disp_text(24,y,(data/10%10+0x10));
              lcd_disp_text(25,y,(data%10+0x10));
              lcd_disp_text(26,y,0x4b);//k
             }
            else if(data/100)
                 {
                  lcd_disp_text(24,y,(data/100+0x10));
                  lcd_disp_text(25,y,(data/10%10+0x10));
                  lcd_disp_text(26,y,(data%10+0x10));
```

```
                }
               else if(data/10)
                    {
                    lcd_disp_text(25,y,(data/10+0x10));
                    lcd_disp_text(26,y,(data%10+0x10));
                    }
                  else
                    {
                    lcd_disp_text(26,y,(data+0x10));
                    }
}

void ch_t_disp(unsigned char ch,unsigned char y)
{
 unsigned int i;
 if(ch==1)
   i=CH1_T;
 else if(ch==2)
     i=CH2_T;

  lcd_disp_string(20,y,"       s ",10);
 if(i/1000000)
   {
    lcd_disp_text(24,y,(i/1000000+0x10));
    lcd_disp_text(25,y,0x0e);
    lcd_disp_text(26,y,(i/100000%10+0x10));
    lcd_disp_text(27,y,(i/10000%10+0x10));
   }
 else if(i/100000)
       {
        lcd_disp_text(24,y,(i/100000+0x10));
        lcd_disp_text(25,y,(i/10000%10+0x10));
        lcd_disp_text(26,y,(i/1000%10+0x10));
        lcd_disp_char(27,y,'m');
       }
     else if(i/10000)
          {
           lcd_disp_text(23,y,(i/10000+0x10));
           lcd_disp_text(24,y,(i/1000%10+0x10));
           lcd_disp_text(25,y,0x0e);
           lcd_disp_text(26,y,(i/100%10+0x10));
           lcd_disp_char(27,y,'m');
          }
        else if(i/1000)
             {
              lcd_disp_text(24,y,(i/1000+0x10));
              lcd_disp_text(25,y,0x0e);
              lcd_disp_text(26,y,(i/100%10+0x10));
              lcd_disp_char(27,y,'m');
             }
           else if(i/100)
                {
                 lcd_disp_text(24,y,(i/100+0x10));
                 lcd_disp_text(25,y,(i/10%10+0x10));
                 lcd_disp_text(26,y,(i%10+0x10));
                 lcd_disp_char(27,y,'u');
                }
              else if(i/10)
                   {
                    lcd_disp_text(25,y,(i/10+0x10));
                    lcd_disp_text(26,y,(i%10+0x10));
                    lcd_disp_char(27,y,'u');
                   }
                 else
                   {
                    lcd_disp_text(26,y,i+0x10);
                    lcd_disp_char(27,y,'u');
                   }
}
```

```
void ch_pv_disp(unsigned char ch,unsigned char y)
{
  unsigned int i=1;
  lcd_disp_string(20,y,"       V ",10);
  if(ch==1)
     i=CH1_PV;
  else if(ch==2)
        i=CH2_PV;
if(i/100000)
 {
  lcd_disp_text(24,y,(i/100000+0x10));
  lcd_disp_text(25,y,(i/10000%10+0x10));
  lcd_disp_text(26,y,0x0e);//.
  lcd_disp_text(27,y,(i/1000%10+0x10));
 }
else if(i/10000)
   {
    lcd_disp_text(24,y,(i/10000+0x10));
    lcd_disp_text(25,y,0x0e);//.
    lcd_disp_text(26,y,(i/1000%10+0x10));
    lcd_disp_text(27,y,(i/100%10+0x10));
   }
  else if(i/1000)
        {
         lcd_disp_text(24,y,(i/1000+0x10));
         lcd_disp_text(25,y,(i/100%10+0x10));
         lcd_disp_text(26,y,(i/10%10+0x10));
         lcd_disp_text(27,y,0x4d);//m
        }
      else if(i/100)
            {
             lcd_disp_text(23,y,(i/100+0x10));
             lcd_disp_text(24,y,(i/10%10+0x10));
             lcd_disp_text(25,y,0x0e);//.
             lcd_disp_text(26,y,(i%10+0x10));
             lcd_disp_text(27,y,0x4d);//m
             }
          else if(i/10)
               {
                lcd_disp_text(24,y,(i/10+0x10));
                lcd_disp_text(25,y,0x0e);//.
                lcd_disp_text(26,y,(i%10+0x10));
                lcd_disp_text(27,y,0x4d);//m
                }
              else
                {
                lcd_disp_text(24,y,0x10);
                lcd_disp_text(25,y,0x0e);//.
                lcd_disp_text(26,y,(i%10+0x10));
                lcd_disp_text(27,y,0x4d);//m
                }
}

void ch_vv_disp(unsigned char ch,unsigned char y)
{
  int i;
  if(ch==1)
    i=CH1_VV;
  else if(ch==2)
        i=CH2_VV;
   lcd_disp_string(20,y,"       V ",10);
   if(i/100000)
      {
      lcd_disp_text(24,y,(i/100000+0x10));
      lcd_disp_text(25,y,(i/10000%10+0x10));
      lcd_disp_text(26,y,0x0e);//.
      lcd_disp_text(27,y,(i/1000%10+0x10));
     }
   else if(i/10000)
        {
```

```
        lcd_disp_text(24,y,(i/10000+0x10));
        lcd_disp_text(25,y,0x0e);//.
        lcd_disp_text(26,y,(i/1000%10+0x10));
        lcd_disp_text(27,y,(i/100%10+0x10));
        }
      else if(i/1000)
            {
            lcd_disp_text(24,y,(i/1000+0x10));
            lcd_disp_text(25,y,(i/100%10+0x10));
            lcd_disp_text(26,y,(i/10%10+0x10));
            lcd_disp_text(27,y,0x4d);//m
            }
          else if(i/100)
               {
                lcd_disp_text(23,y,(i/100+0x10));
                lcd_disp_text(24,y,(i/10%10+0x10));
                lcd_disp_text(25,y,0x0e);
                lcd_disp_text(26,y,(i%10+0x10));
                lcd_disp_text(27,y,0x4d);//m
               }
              else if(i/10)
                   {
                   lcd_disp_text(24,y,(i/10+0x10));
                   lcd_disp_text(25,y,0x0e);
                   lcd_disp_text(26,y,(i%10+0x10));
                   lcd_disp_text(27,y,0x4d);//m
                   }
                  else
                   {lcd_disp_text(26,y,+0x10);
                    lcd_disp_text(27,y,0x4d);//m
                   }
}

void ch_av_disp(unsigned char ch,unsigned char y)
{int i=1;
 lcd_disp_string(20,y,"       V ",10);
  if(ch==1)
    i=CH1_AV;
  else if(ch==2)
      i=CH2_AV;
  if(i>=0)
   {
     if(i/100000)
       {
       lcd_disp_text(24,y,(i/100000+0x10));
       lcd_disp_text(25,y,0x0e);//.
       lcd_disp_text(26,y,(i/10000%10+0x10));
       lcd_disp_text(27,y,(i/1000%10+0x10));
       }
     else if(i/10000)
            {
            lcd_disp_text(24,y,(i/10000+0x10));
            lcd_disp_text(25,y,(i/1000%10+0x10));
            lcd_disp_text(26,y,(i/100%10+0x10));
            lcd_disp_text(27,y,0x4d);//m
            }
          else if(i/1000)
               {
                lcd_disp_text(23,y,(i/1000+0x10));
                lcd_disp_text(24,y,(i/100%10+0x10));
                lcd_disp_text(25,y,0x0e);//.
                lcd_disp_text(26,y,(i/10%10+0x10));
                lcd_disp_text(27,y,0x4d);//m
               }
              else if(i/100)
                   {
                   lcd_disp_text(24,y,(i/100+0x10));
                   lcd_disp_text(25,y,0x0e);//.
                   lcd_disp_text(26,y,(i/10%10+0x10));
                   lcd_disp_text(27,y,0x4d);//m
```

```
                }
               else if(i/10)
                    {
                    lcd_disp_text(24,y,(i/10+0x10));
                    lcd_disp_text(25,y,(i%10+0x10));
                    lcd_disp_text(26,y,0x10);
                    lcd_disp_text(27,y,0x55);//u
                    }
                   else
                    {
                    lcd_disp_text(25,y,(i+0x10));
                    lcd_disp_text(26,y,0x10);
                    lcd_disp_text(27,y,0x55);//u
                    }
  }
  else
  {i=-i;
   if(i/100000)
      {
      lcd_disp_text(23,y,0x0d);//-
      lcd_disp_text(24,y,(i/100000+0x10));
      lcd_disp_text(25,y,0x0e);//.
      lcd_disp_text(26,y,(i/10000%10+0x10));
      lcd_disp_text(27,y,(i/1000%10+0x10));
      }
     else if(i/10000)
          {
          lcd_disp_text(23,y,0x0d);//-
          lcd_disp_text(24,y,(i/10000+0x10));
          lcd_disp_text(25,y,(i/1000%10+0x10));
          lcd_disp_text(26,y,(i/100%10+0x10));
          lcd_disp_text(27,y,0x4d);//m
          }
        else if(i/1000)
             {
              lcd_disp_text(24,y,0x0d);//-
              lcd_disp_text(25,y,(i/1000+0x10));
              lcd_disp_text(26,y,(i/100%10+0x10));
              lcd_disp_text(27,y,0x4d);//m
             }
            else if(i/100)
                 {
                 lcd_disp_text(23,y,0x0d);//-
                 lcd_disp_text(24,y,(i/100+0x10));
                 lcd_disp_text(25,y,0x0e);//.
                 lcd_disp_text(26,y,(i/10%10+0x10));
                 lcd_disp_text(27,y,0x4d);//m
                 }
                else if(i/10)
                     {
                     lcd_disp_text(23,y,0x0d);//-
                     lcd_disp_text(24,y,(i/10+0x10));
                     lcd_disp_text(25,y,(i%10+0x10));
                     lcd_disp_text(26,y,0x10);
                     lcd_disp_text(27,y,0x55);//u
                     }
                    else
                     {
                      lcd_disp_text(24,y,0x0d);//-
                      lcd_disp_text(25,y,(i+0x10));
                      lcd_disp_text(26,y,0x10);
                      lcd_disp_text(27,y,0x55);//u
                     }
  }
}

void V_disp(unsigned char status)
{
  int i=0;
  unsigned char line_v=0,y=0,v_index;
```

```
if(status1_ctype_flag==1)
  v_index=CH1_V_Index;
else if(status1_ctype_flag==2)
      v_index=CH2_V_Index;
switch(status)
{
case 1:line_v=0;
       y=7;
      if(line_V1>line_V2)
        i=(V_cell[v_index]*(line_V1-line_V2))>>4;
      else
        i=(V_cell[v_index]*(line_V2-line_V1))>>4;
      break;
case 2:
      line_v=line_V1;
      y=11;
      if(line_v-64>=0)
        i=(V_cell[v_index]*(line_v-64))>>4;
      else
        i=(V_cell[v_index]*(64-line_v))>>4;
      break;
case 3:
      line_v=line_V2;
      y=15;
      if(line_v-64>=0)
        i=(V_cell[v_index]*(line_v-64))>>4;
      else
        i=(V_cell[v_index]*(64-line_v))>>4;
      break;
}
lcd_disp_string(23,y ,"     V ",7);
if(line_v-64<=0)
  {
    if(i/10000)
       {
       lcd_disp_text(24,y,(i/10000+0x10));
       lcd_disp_text(25,y,(i/1000%10+0x10));
       lcd_disp_text(26,y,0x0e);
       lcd_disp_text(27,y,(i/100%10+0x10));
       }
    else if(i/1000)
          {
           lcd_disp_text(24,y,(i/1000+0x10));
           lcd_disp_text(25,y,0x0e);
           lcd_disp_text(26,y,(i/100%10+0x10));
           lcd_disp_text(27,y,(i/10%10+0x10));
           }
         else if(i/100)
               {
                lcd_disp_text(24,y,(i/100+0x10));
                lcd_disp_text(25,y,(i/10%10+0x10));
                lcd_disp_text(26,y,(i%10+0x10));
                lcd_disp_char(27,y,'m');
               }
              else if(i/10)
                   {
                    lcd_disp_text(25,y,(i/10+0x10));
                    lcd_disp_text(26,y,(i%10+0x10));
                    lcd_disp_char(27,y,'m');
                   }
                  else
                    {
                     lcd_disp_text(26,y,(i+0x10));
                     lcd_disp_char(27,y,'m');
                    }
  }
else
  {
    if(i/10000)
       {
```

```
        lcd_disp_char(23,y,'-');
        lcd_disp_text(24,y,(i/10000+0x10));
        lcd_disp_text(25,y,(i/1000%10+0x10));
        lcd_disp_text(26,y,0x0e);
        lcd_disp_text(27,y,(i/100%10+0x10));
        }
     else if(i/1000)
          {
           lcd_disp_char(23,y,'-');
           lcd_disp_text(24,y,(i/1000+0x10));
           lcd_disp_text(25,y,0x0e);
           lcd_disp_text(26,y,(i/100%10+0x10));
           lcd_disp_text(27,y,(i/10%10+0x10));
           }
         else if(i/100)
              {
               lcd_disp_char(23,y,'-');
               lcd_disp_text(24,y,(i/100+0x10));
               lcd_disp_text(25,y,(i/10%10+0x10));
               lcd_disp_text(26,y,(i%10+0x10));
               lcd_disp_char(27,y,'m');
               }
             else if(i/10)
                  {
                   lcd_disp_char(24,y,'-');
                   lcd_disp_text(25,y,(i/10+0x10));
                   lcd_disp_text(26,y,(i%10+0x10));
                   lcd_disp_char(27,y,'m');
                  }
                 else
                  {
                   lcd_disp_char(25,y,'-');
                   lcd_disp_text(26,y,(i+0x10));
                   lcd_disp_char(27,y,'m');
                  }
   }
}

void T_disp(unsigned char status)
{
  int i=0;
  unsigned char line_t=0,y=0;
  switch(status)
  {
  case 1:line_t=160;
        y=7;
       if(line_T1>line_T2)
         i=(T_cell[T_Index]*(line_T1-line_T2))>>4;
       else
         i=(T_cell[T_Index]*(line_T2-line_T1))>>4;
       break;
  case 2:
       line_t=line_T1;
       y=11;
       if(line_t-80>=0)
         i=(T_cell[T_Index]*(line_t-80))>>4;
       else
         i=(T_cell[T_Index]*(80-line_t))>>4;
       break;
  case 3:
       line_t=line_T2;
       y=15;
       if(line_t-80>=0)
         i=(T_cell[T_Index]*(line_t-80))>>4;
       else
         i=(T_cell[T_Index]*(80-line_t))>>4;
       break;
  }
  lcd_disp_string(20,y ,"       s ",10);
  if(line_t-80>=0)
```

```
{
 if(i/1000000)
   {
   lcd_disp_text(24,y,(i/1000000+0x10));
   lcd_disp_text(25,y,0x0e);
   lcd_disp_text(26,y,(i/100000%10));
   lcd_disp_text(27,y,(i/10000%10+0x10));
   }
 else if(i/100000)
      {
       lcd_disp_text(24,y,(i/100000+0x10));
       lcd_disp_text(25,y,(i/10000%10+0x10));
       lcd_disp_text(26,y,(i/1000%10+0x10));
       lcd_disp_char(27,y,'m');
      }
     else if(i/10000)
          {
           lcd_disp_text(25,y,(i/10000+0x10));
           lcd_disp_text(26,y,(i/1000%10+0x10));
           lcd_disp_char(27,y,'m');
          }
         else if(i/1000)
              {
               lcd_disp_text(24,y,(i/1000+0x10));
               lcd_disp_text(25,y,0x0e);
               lcd_disp_text(26,y,(i/100%10+0x10));
               lcd_disp_char(27,y,'m');
              }
             else if(i/100)
                  {
                   lcd_disp_text(24,y,(i/100+0x10));
                   lcd_disp_text(25,y,(i/10%10+0x10));
                   lcd_disp_text(26,y,(i%10+0x10));
                   lcd_disp_char(27,y,'u');
                  }
                 else if(i/10)
                      {
                       lcd_disp_text(25,y,(i/10+0x10));
                       lcd_disp_text(26,y,(i%10+0x10));
                       lcd_disp_char(27,y,'u');
                       }
                      else
                       {
                       lcd_disp_text(26,y,(i+0x10));
                       lcd_disp_char(27,y,'u');
                       }
}
else
 {
  if(i/1000000)
    {
    lcd_disp_char(23,y,'-');
    lcd_disp_text(24,y,(i/1000000+0x10));
    lcd_disp_text(25,y,0x0e);
    lcd_disp_text(26,y,(i/100000%10));
    lcd_disp_text(27,y,(i/10000%10+0x10));
    }
  else if(i/100000)
       {
        lcd_disp_char(23,y,'-');
        lcd_disp_text(24,y,(i/100000+0x10));
        lcd_disp_text(25,y,(i/10000%10+0x10));
        lcd_disp_text(26,y,(i/1000%10+0x10));
        lcd_disp_char(27,y,'m');
       }
      else if(i/10000)
           {
            lcd_disp_char(24,y,'-');
            lcd_disp_text(25,y,(i/10000+0x10));
            lcd_disp_text(26,y,(i/1000%10+0x10));
```

```
                lcd_disp_char(27,y,'m');
               }
              else if(i/1000)
                  {
                   lcd_disp_char(23,y,'-');
                   lcd_disp_text(24,y,(i/1000+0x10));
                   lcd_disp_text(25,y,0x0e);
                   lcd_disp_text(26,y,(i/100%10+0x10));
                   lcd_disp_char(27,y,'m');
                  }
                 else if(i/100)
                     {
                      lcd_disp_char(23,y,'-');
                      lcd_disp_text(24,y,(i/100+0x10));
                      lcd_disp_text(25,y,(i/10%10+0x10));
                      lcd_disp_text(26,y,(i%10+0x10));
                      lcd_disp_char(27,y,'u');
                     }
                   else if(i/10)
                       {
                        lcd_disp_char(24,y,'-');
                        lcd_disp_text(25,y,(i/10+0x10));
                        lcd_disp_text(26,y,(i%10+0x10));
                        lcd_disp_char(27,y,'u');
                        }
                       else
                        {
                        lcd_disp_char(25,y,'-');
                        lcd_disp_text(26,y,(i+0x10));
                        lcd_disp_char(27,y,'u');
                        }
   }
}
```

4.fft.h□ □ □ :

```
int code(int i)
{int two[M]={0},j,su,bn,sum=0;
 for(j=0;j<M;j++)
 {bn=(int)(i/2);
  two[j]=i%2;
  i=bn;}
for(j=0;j<M;j++)
{su=two[j]*pow(2,M-1-j);
 sum=sum+su;
}
return sum;
}

typedef struct
{float real;
 float imag;
}complex;


void fft(void)
{
  float a,t;
  complex x[N],temp[N],W[N/2],temp1,tm;
  int i,j,b,c,d,p,l;
        IOWR(RUN_STOP_BASE,0,1);
        i=(160000000>>7)/CH1_F;
        if(i==0)i=1;
        IOWR(FP_FS_BASE,0,i);
        IOWR(BUFF1_START_BASE,0,1);
        usleep(1);
        IOWR(BUFF1_START_BASE,0,0);
        usleep(1);
        IOWR(FP_FS_BASE,0,i);
        IOWR(BUFF2_START_BASE,0,1);
        usleep(1);
```

```
        IOWR(BUFF2_START_BASE,0,0);
        usleep(1);
        for( i=0;i<128;i++)
          {
          IOWR(BUFF1_RDADDR_BASE,0,i);
          fft1_data[i]=127-(IORD(BUFF1_DATA_BASE,0)>>1);
          }
        for(i=0;i<128;i++)
          {
          IOWR(BUFF2_RDADDR_BASE,0,i);
          fft2_data[i]=127-(IORD(BUFF2_DATA_BASE,0)>>1);
          }
        IOWR(FP_FS_BASE,0,FS_FP[T_Index]);
        IOWR(BUFF1_START_BASE,0,1);
        usleep(1);
        IOWR(BUFF1_START_BASE,0,0);
        usleep(1);
 a=2*PA/N;
 if(shift_flag==0)
  {
   for(i=0;i<128;i++)
     fft_data[i]=fft1_data[i];
  }
 if(shift_flag==1)
 {
  for(i=0;i<128;i++)
     fft_data[i]=fft2_data[i];
 }
 for(i=0;i<N;i++)
   {x[i].imag=0;
    x[i].real=0;
   }
 for(i=0;i<128;i++)
    x[i].real=fft_data[i]-64;
 for(i=0;i<N/2;i++)
   {W[i].real=cos(i*a);
    W[i].imag=-sin(i*a);
   }
 for(i=0;i<N;i++)
   {b=code(i);
    temp[i]=x[b];}
    for(l=1;l<=M;l++)
       {c=pow(2,l-1);
        for(j=0;j<=c-1;j++)
            {p=j*pow(2,M-l);
             for(d=j;d<=N-1;d+=pow(2,l))
                {temp1=temp[d];
                 tm.real=temp[d+c].real*W[p].real-temp[d+c].imag*W[p].imag;
                 tm.imag=temp[d+c].real*W[p].imag+temp[d+c].imag*W[p].real;
                 temp[d].real=temp[d].real+tm.real;
                 temp[d].imag=temp[d].imag+tm.imag;
                 temp[d+c].real=temp1.real-tm.real;
                 temp[d+c].imag=temp1.imag-tm.imag;
                }
             }
       }
       for(i=0;i<N;i++)
          {t=pow(temp[i].real,2)+pow(temp[i].imag,2);
           fft_data[i]=sqrt(t);
          }
 }

void fft_disp(void)
{ int i=0,j=0;
  float temp_data[128]={0},max;
  for( i=0;i<20;i++)
    for(j=0;j<128;j++)
      {
       lcd_set_graph(i,j);
       lcd_disp8(0x00);
```

```
    }
  max=1.2*fft_data[1];
 for(i=0;i<128;i++)
   temp_data[i]=127*(1-fft_data[i]/max);

 for(i=0;i<20;i+=2)
   for(j=temp_data[i>>1];j<128;j++)
   {
    lcd_set_graph(i,j);
    usleep(2000);
    lcd_ctrl(0xff);
   }
 lcd_disp_text(1,1,0x11);
 lcd_disp_text(3,1,0x12);
 lcd_disp_text(5,1,0x13);
 lcd_disp_text(7,1,0x14);
 lcd_disp_text(9,1,0x15);
 lcd_disp_text(11,1,0x16);
 lcd_disp_text(13,1,0x17);
 lcd_disp_text(15,1,0x18);
 lcd_disp_text(17,1,0x19);
       lcd_disp_string(20,4," 01  100% ",10);
       lcd_disp_string(20,5," 03     % ",10);
       lcd_disp_string(20,6," 05     % ",10);
       lcd_disp_string(20,7," 07     % ",10);
       lcd_disp_string(20,8," 09     % ",10);
       lcd_disp_string(20,9," 11     % ",10);
       lcd_disp_string(20,10," 13     % ",10);
       lcd_disp_string(20,11," 15     % ",10);
 for(i=3;i<17;i+=2)
  {
  j=(fft_data[i]*10000)/fft_data[1];
  if(j/1000)
     {
     lcd_disp_text(24,(i+8)>>1,(j/1000+0x10));
     lcd_disp_text(25,(i+8)>>1,(j/100%10+0x10));
     lcd_disp_char(26,(i+8)>>1,'.');
     lcd_disp_text(27,(i+8)>>1,(j/10%10+0x10));
     }
  else if(j/100)
        {
         lcd_disp_text(24,(i+8)>>1,(j/100+0x10));
         lcd_disp_char(25,(i+8)>>1,'.');
         lcd_disp_text(26,(i+8)>>1,(j/10%10+0x10));
         lcd_disp_text(27,(i+8)>>1,(j%10+0x10));
        }
       else if(j/10)
            {
             lcd_disp_text(24,(i+8)>>1,0x10);
             lcd_disp_char(25,(i+8)>>1,'.');
             lcd_disp_text(26,(i+8)>>1,(j/10+0x10));
             lcd_disp_text(27,(i+8)>>1,(j%10+0x10));
            }
           else
            {
             lcd_disp_text(24,(i+8)>>1,0x10);
             lcd_disp_char(25,(i+8)>>1,'.');
             lcd_disp_text(26,(i+8)>>1,0x10);
             lcd_disp_text(27,(i+8)>>1,j+0x10);
            }

   }
}
```

5. SD_Card.h□ □ □ :

```
#include "SD_Card.h"
unsigned                                                                                                    short
crc_tab[256]={0,0x1021,0x2042,0x3063,0x4084,0x50a5,0x60c6,0x70e7,0x8108,0x9129,0xa14a,0xb16b,0xc18c,0xd1ad,0xe1ce,0xf1
```

```
ef,0x1231,0x210,0x3273,0x2252,0x52b5,0x4294,0x72f7,0x62d6,0x9339,0x8318,0xb37b,0xa35a,0xd3bd,0xc39c,0xf3ff,0xe3de,0x24
62,0x3443,0x420,0x1401,0x64e6,0x74c7,0x44a4,0x5485,0xa56a,0xb54b,0x8528,0x9509,0xe5ee,0xf5cf,0xc5ac,0xd58d,0x3653,0x26
72,0x1611,0x630,0x76d7,0x66f6,0x5695,0x46b4,0xb75b,0xa77a,0x9719,0x8738,0xf7df,0xe7fe,0xd79d,0xc7bc,0x48c4,0x58e5,0x68
86,0x78a7,0x840,0x1861,0x2802,0x3823,0xc9cc,0xd9ed,0xe98e,0xf9af,0x8948,0x9969,0xa90a,0xb92b,0x5af5,0x4ad4,0x7ab7,0x6a9
6,0x1a71,0xa50,0x3a33,0x2a12,0xdbfd,0xcbdc,0xfbbf,0xeb9e,0x9b79,0x8b58,0xbb3b,0xab1a,0x6ca6,0x7c87,0x4ce4,0x5cc5,0x2c22,
0x3c03,0xc60,0x1c41,0xedae,0xfd8f,0xcdec,0xddcd,0xad2a,0xbd0b,0x8d68,0x9d49,0x7e97,0x6eb6,0x5ed5,0x4ef4,0x3e13,0x2e32,0
x1e51,0xe70,0xff9f,0xefbe,0xdfdd,0xcffc,0xbf1b,0xaf3a,0x9f59,0x8f78,0x9188,0x81a9,0xb1ca,0xa1eb,0xd10c,0xc12d,0xf14e,0xe16
f,0x1080,0xa1,0x30c2,0x20e3,0x5004,0x4025,0x7046,0x6067,0x83b9,0x9398,0xa3fb,0xb3da,0xc33d,0xd31c,0xe37f,0xf35e,0x2b1,0
x1290,0x22f3,0x32d2,0x4235,0x5214,0x6277,0x7256,0xb5ea,0xa5cb,0x95a8,0x8589,0xf56e,0xe54f,0xd52c,0xc50d,0x34e2,0x24c3,
0x14a0,0x481,0x7466,0x6447,0x5424,0x4405,0xa7db,0xb7fa,0x8799,0x97b8,0xe75f,0xf77e,0xc71d,0xd73c,0x26d3,0x36f2,0x691,0
x16b0,0x6657,0x7676,0x4615,0x5634,0xd94c,0xc96d,0xf90e,0xe92f,0x99c8,0x89e9,0xb98a,0xa9ab,0x5844,0x4865,0x7806,0x6827,
0x18c0,0x8e1,0x3882,0x28a3,0xcb7d,0xdb5c,0xeb3f,0xfb1e,0x8bf9,0x9bd8,0xabbb,0xbb9a,0x4a75,0x5a54,0x6a37,0x7a16,0xaf1,0x
1ad0,0x2ab3,0x3a92,0xfd2e,0xed0f,0xdd6c,0xcd4d,0xbdaa,0xad8b,0x9de8,0x8dc9,0x7c26,0x6c07,0x5c64,0x4c45,0x3ca2,0x2c83,0x
1ce0,0xcc1,0xef1f,0xff3e,0xcf5d,0xdf7c,0xaf9b,0xbfba,0x8fd9,0x9ff8,0x6e17,0x7e36,0x4e55,0x5e74,0x2e93,0x3eb2,0xed1,0x1ef0};
int FileTotal2=0;
//---------------------------------------------------------------------------
typedef struct
{
    UINT16   BytePerSec;
    BYTE  SecPerClus;
    UINT16   RsvdSecCnt;
    UINT32  FATSz;
    UINT32  RootClus;

 }BPB_str;
//---------------------------------------------------------------------------
typedef struct
{
   char   Name[12];
   UINT32   FileSize;
   UINT32   FileClus[6];
   UINT32   FileSec;
}DIR_str;
//---------------------------------------------------------------------------
BPB_str BPB;
DIR_str DIR[16];
UINT32   FATRoot,DirRoot;
BYTE Buffer[4096]={0};
//---------------------------------------------------------------------------
BYTE read_status;
BYTE response_buffer[20];
BYTE RCA[2];
BYTE cmd_buffer[5];
const BYTE cmd0[5]   = {0x40,0x00,0x00,0x00,0x00};//Reset SD Card
const BYTE cmd55[5]  = {0x77,0x00,0x00,0x00,0x00};//Next CMD is ASC
const BYTE cmd2[5]   = {0x42,0x00,0x00,0x00,0x00};//asks to send the CID numbers
const BYTE cmd3[5]   = {0x43,0x00,0x00,0x00,0x00};//Send RCA
const BYTE cmd7[5]   = {0x47,0x00,0x00,0x00,0x00};//Select one card,put it into Transfer State
const BYTE cmd9[5]   = {0x49,0x00,0x00,0x00,0x00};//Ask send CSD
const BYTE cmd16[5]  = {0x50,0x00,0x00,0x02,0x00};//Select a block length
const BYTE cmd17[5]  = {0x51,0x00,0x00,0x00,0x00};//Read a single block
const BYTE acmd6[5]  = {0x46,0x00,0x00,0x00,0x02};//SET BUS WIDTH
const BYTE cmd24[5]  = {0x58,0x00,0x00,0x00,0x00};//Write a single block
const BYTE acmd41[5] = {0x69,0x0f,0xf0,0x00,0x00};//Active Card's ini process
const BYTE acmd51[5] = {0x73,0x00,0x00,0x00,0x00};//Read SCR(Configuration Reg)
//---------------------------------------------------------------------------
void Ncr(void)
{
  SD_CMD_IN;
  SD_CLK_LOW;
  SD_CLK_HIGH;
  SD_CLK_LOW;
  SD_CLK_HIGH;
}
//---------------------------------------------------------------------------
void Ncc(void)
{
  int i;
  for(i=0;i<8;i++)
  {
```

```
    SD_CLK_LOW;
    SD_CLK_HIGH;
  }
}
//-------------------------------------------------------------------------
BYTE SD_card_init(void)
{
    BYTE x,y;
    SD_CMD_OUT;
    SD_DAT_IN;
    SD_CLK_HIGH;
    SD_CMD_HIGH;
    SD_DAT_LOW;
    read_status=0;
    for(x=0;x<40;x++)
      Ncr();
    for(x=0;x<5;x++)
      cmd_buffer[x]=cmd0[x];
    y = send_cmd(cmd_buffer);//Reset SD Card
    do
    {
      for(x=0;x<40;x++);
        Ncc();
      for(x=0;x<5;x++)
        cmd_buffer[x]=cmd55[x];
      y = send_cmd(cmd_buffer);//Next CMD is ASC
      Ncr();
      if(response_R(1)>1) //response too long or crc error
        return 1;
      Ncc();
      for(x=0;x<5;x++)
        cmd_buffer[x]=acmd41[x];
      y = send_cmd(cmd_buffer);//Active Card's ini process
      Ncr();
    } while(response_R(3)==1);
    Ncc();
    for(x=0;x<5;x++)
      cmd_buffer[x]=cmd2[x];//asks to send the CID numbers
    y = send_cmd(cmd_buffer);
    Ncr();
    if(response_R(2)>1)
    return 1;
    Ncc();
    for(x=0;x<5;x++)
      cmd_buffer[x]=cmd3[x];
    y = send_cmd(cmd_buffer);//Send RCA
    Ncr();
    if(response_R(6)>1)
    return 1;
    RCA[0]=response_buffer[1];
    RCA[1]=response_buffer[2];
    Ncc();
    for(x=0;x<5;x++)
      cmd_buffer[x]=cmd9[x];
    cmd_buffer[1] = RCA[0];
    cmd_buffer[2] = RCA[1];
    y = send_cmd(cmd_buffer);//Ask send CSD
    Ncr();
    if(response_R(2)>1)
    return 1;
    Ncc();
    for(x=0;x<5;x++)
      cmd_buffer[x]=cmd7[x];
    cmd_buffer[1] = RCA[0];
    cmd_buffer[2] = RCA[1];
    y = send_cmd(cmd_buffer);//Select one card,put it into Transfer State
    Ncr();
    if(response_R(1)>1)
    return 1;
    Ncc();
```

```
    for(x=0;x<5;x++)
      cmd_buffer[x]=cmd16[x];
    y = send_cmd(cmd_buffer);//Select a block length
    Ncr();
    if(response_R(1)>1)
      return 1;
    read_status =1; //sd card ready
    return 0;
}
//-----------------------------------------------------------------------
BYTE SD_read_lba(BYTE *buff,UINT32 lba,UINT32 seccnt)
{
  BYTE c=0;
  UINT32  i,j,addr;
  lba+=235;
  for(j=0;j<seccnt;j++)
  {
    {
      Ncc();
      cmd_buffer[0] = cmd17[0];
      cmd_buffer[1] = (lba>>15)&0xff;
      cmd_buffer[2] = (lba>>7)&0xff;
      cmd_buffer[3] = (lba<<1)&0xff;
      cmd_buffer[4] = 0;
      addr=(cmd_buffer[1]<<24)|(cmd_buffer[2]<<16)|(cmd_buffer[3]<<8)|(cmd_buffer[4]);
      lba++;
      send_cmd(cmd_buffer);
      Ncr();
    }
    while(1)
    {
      SD_CLK_LOW;
      SD_CLK_HIGH;
      if(!(SD_TEST_DAT))
      break;
    }
    for(i=0;i<512;i++)
    {
      BYTE j;
      for(j=0;j<8;j++)
      {
        SD_CLK_LOW;
        SD_CLK_HIGH;
        c <<= 1;
        if(SD_TEST_DAT)
        c |= 0x01;
      }
      *buff=c;
      buff++;
    }
    for(i=0; i<16; i++)
    {
       SD_CLK_LOW;
       SD_CLK_HIGH;
    }
  }
  read_status = 1;  //SD data next in
  return 0;
}
//-----------------------------------------------------------------------
BYTE SD_write_lba(BYTE *buff,UINT32 lba,UINT32 seccnt)
{
    BYTE c=0;
    UINT32  i,j,l;
    unsigned short crc=0;
    lba+=235;
    for(l=0;l<seccnt;l++)
    {
       Ncc();
       cmd_buffer[0] = cmd24[0];
```

```
      cmd_buffer[1] = (lba>>15)&0xff;
      cmd_buffer[2] = (lba>>7)&0xff;
      cmd_buffer[3] = (lba<<1)&0xff;
      cmd_buffer[4] = 0;
      lba++;
      send_cmd(cmd_buffer);
      for(i=0;i<100;i++)
      {
         SD_CLK_LOW;
         SD_CLK_HIGH;
      }
      crc=0;
      SD_CLK_LOW;
      SD_DAT_OUT;
      SD_DAT_LOW;
      SD_CLK_HIGH;
      for(i=0;i<512;i++)
      {
         BYTE j;
         c=*buff;
         for(j=0; j<8; j++)
         {
            SD_CLK_LOW;
            if(c&0x80)
            SD_DAT_HIGH;
            else
            SD_DAT_LOW;
            SD_CLK_HIGH;
            c<<=1;
         }
         crc=crc_tab[(crc>>8)^(*buff)]^(crc<<8);
         buff++;
      }
      for(j=0; j<16; j++)
      {
         SD_CLK_LOW;
         if(crc&0x8000)
         SD_DAT_HIGH;
         else
         SD_DAT_LOW;
         SD_CLK_HIGH;
         crc<<=1;
      }
      c=0;
      SD_CLK_LOW;
      SD_DAT_HIGH;
      SD_DAT_IN;
      SD_CLK_HIGH;
      for(i=0; i<7; i++)
      {
         SD_CLK_LOW;
         SD_CLK_HIGH;
         c=c<<1;
         if(SD_TEST_DAT)
         c|=0x01;
      }
      Ncc();
       i=0;j=0;
      while(1)
      {
         SD_CLK_LOW;
         SD_CLK_HIGH;
         if(SD_TEST_DAT)
         break;
      }
    }
   return 0;
}
//-------------------------------------------------------------------------
BYTE response_R(BYTE s)
```

```
{
  BYTE a=0,b=0,c=0,r=0,crc=0;
  BYTE i,j=6,k;
  while(1)
  {
    SD_CLK_LOW;
    SD_CLK_HIGH;
    if(!(SD_TEST_CMD))
    break;
    if(crc++ >100)
    return 2;
  }
  crc =0;
  if(s == 2)
  j = 17;
  for(k=0; k<j; k++)
  {
    c = 0;
    if(k > 0)                    //for crc culcar
    b = response_buffer[k-1];
    for(i=0; i<8; i++)
    {
      SD_CLK_LOW;
      if(a > 0)
      c <<= 1;
      else
      i++;
      a++;
      SD_CLK_HIGH;
      if(SD_TEST_CMD)
      c |= 0x01;
      if(k > 0)
      {
        crc <<= 1;
        if((crc ^ b) & 0x80)
        crc ^= 0x09;
        b <<= 1;
        crc &= 0x7f;
      }
    }
    if(s==3)
    {
      if( k==1 &&(!(c&0x80)))
      r=1;
    }
    response_buffer[k] = c;
  }
  if(s==1 || s==6)
  {
    if(c != ((crc<<1)+1))
    r=2;
  }
  return r;
}
//-------------------------------------------------------------------------
BYTE send_cmd(BYTE *in)
{
  int i,j;
  BYTE b,crc=0;
  SD_CMD_OUT;
  for(i=0; i < 5; i++)
  {
    b = in[i];
    for(j=0; j<8; j++)
    {
      SD_CLK_LOW;
      if(b&0x80)
      SD_CMD_HIGH;
      else
      SD_CMD_LOW;
```

```
      crc <<= 1;
      SD_CLK_HIGH;
      if((crc ^ b) & 0x80)
      crc ^= 0x09;
      b<<=1;
    }
    crc &= 0x7f;
  }
  crc =((crc<<1)|0x01);
  b = crc;
  for(j=0; j<8; j++)
  {
    SD_CLK_LOW;
    if(crc&0x80)
    SD_CMD_HIGH;
    else
    SD_CMD_LOW;
    SD_CLK_HIGH;
    crc<<=1;
  }
  return b;
}

//-------------------------------------------------------
int fileindex(void)
{
  int index=0,i=0,j=0,k=0,FileTotal=0;
  SD_read_lba(Buffer,DirRoot,1);
  for(k=0;k<16;k++)
  {
      if ((!(Buffer[11+32*k]&0x10)) && Buffer[32*k]!=0 && Buffer[32*k]!=0xe5)
      {
          for(i=0;i<8;i++)
          {
              if(Buffer[i+32*k]!=0x20)
              {
                  DIR[index].Name[j++]=Buffer[i+32*k];
              }
          }
          DIR[index].Name[j++]='.';
          for(i=8;i<11;i++)
          {
              if(Buffer[i+32*k]!=0x20)
                  DIR[index].Name[j++]=Buffer[i+32*k];
          }
          j=0;
          DIR[index].FileClus[0]=(DIR[index].FileClus[0]|Buffer[21+32*k])<<8;
          DIR[index].FileClus[0]=(DIR[index].FileClus[0]|Buffer[20+32*k])<<8;
          DIR[index].FileClus[0]=(DIR[index].FileClus[0]|Buffer[27+32*k])<<8;
          DIR[index].FileClus[0]=DIR[index].FileClus[0]|Buffer[26+32*k];
          DIR[index].FileSize=(DIR[index].FileSize|Buffer[31+32*k])<<8;
          DIR[index].FileSize=(DIR[index].FileSize|Buffer[30+32*k])<<8;
          DIR[index].FileSize=(DIR[index].FileSize|Buffer[29+32*k])<<8;
          DIR[index].FileSize=DIR[index].FileSize|Buffer[28+32*k];
          FileTotal++;
          index++;
      }
  }
  return FileTotal;
}
//------------------------------------------------------------------------
void ClusIndex(int FileTotal)
{
    int i,j,ThisClus;

    SD_read_lba(Buffer,FATRoot,8);
    for(i=0;i<FileTotal;i++)
    {
        ThisClus=DIR[i].FileClus[0];
        if(ThisClus<2)
```

```
        continue;
      for(j=1;j<6;j++)
      {
         DIR[i].FileClus[j]=(DIR[i].FileClus[j]|Buffer[3+4*ThisClus])<<8;
         DIR[i].FileClus[j]=(DIR[i].FileClus[j]|Buffer[2+4*ThisClus])<<8;
         DIR[i].FileClus[j]=(DIR[i].FileClus[j]|Buffer[1+4*ThisClus])<<8;
         DIR[i].FileClus[j]=DIR[i].FileClus[j]|Buffer[4*ThisClus];
         ThisClus=DIR[i].FileClus[j];
         if(ThisClus==0x0FFFFFFF)
            break;
      }
   }

}
//---------------------------------------------------------------------------
void FileN(char * FileName,int i,int ChNum)
{
   *FileName=DIR[i].Name[ChNum];
}
//---------------------------------------------------------------------------
```

6. CHARBANK.H□□□:

```
#define uchar unsigned char
#define uint unsigned int
#define N 128
#define M 7
#define PA 3.1415927
#define GASAL     0x00
#define GASAH     0x00
#define TASAL     0x00
#define TASAH     0x0f
#define CGRAMSA   0x1e00
#define ZIFU_NUM  30
unsigned int CH1_T=0;
unsigned int CH1_F=0;
unsigned int CH1_PV=0;
unsigned int CH1_VV=0;
int CH1_AV=0;
unsigned int CH1_MAX=0;
unsigned int CH1_MIN=0;
unsigned int CH1_TNUM=0;
unsigned int CH2_T=0;
unsigned int CH2_F=0;
unsigned int CH2_PV=0;
unsigned int CH2_VV=0;
int CH2_AV=0;
unsigned int CH2_MAX=0;
unsigned int CH2_MIN=0;
unsigned int CH2_TNUM=0;
short T_addr=80;
short V_data1=0;
short V_data2=0;
unsigned char T_Index=6;
unsigned char CH1_V_Index=7;
unsigned char CH2_V_Index=7;
unsigned char VDIV[][5]=
{
  " 10mV"," 20mV"," 30mV"," 40mV"," 50mV"," 60mV"," 80mV",
  "100mV","150mV","200mV","300mV","400mV","500mV","600mV","800mV",
  "1.25V"," 2.5V"
};
unsigned int V_cell[]=
{
  10,20,30,40,51,61,80,100,146,207,311,415,498,622,830,1245,2490
};
unsigned char DIN[]=
{
 249,124,83,62,49,41,31,
 25,17,12,8,6,5,4,3,2,1
};
```

```
unsigned char TDIV[][5]=
{
 “  1us”,”  2us”,”  5us”,” 10us”,” 20us”,” 50us”,”100us”,”200us”,”500us”,
 “  1ms”,”  2ms”,”  5ms”,” 10ms”,” 20ms”,” 50ms”,”100ms”,”200ms”,”500ms”
};
unsigned int T_cell[]=
{
 1,2,5,10,20,50,100,200,500,
 1000,2000,5000,10000,20000,50000,100000,200000,500000
};
unsigned int FS[]=
{
 16000000,8000000,3200000,1600000,800000,320000,160000,80000,32000,
 16000,8000,3200,1600,800,320,160,80,32
};
unsigned int FS_FP[]=
{
 10,20,50,100,200,500,1000,2000,5000,
 10000,20000,50000,100000,200000,500000,1000000,2000000,5000000
};
unsigned char CHUFA[][3]=
{
 “-60”,”-50”,”-40”,”-30”,”-20”,”-10”,”  0”,”+10”,”+20”,”+30”,”+40”,”+50”,”+60”
};
unsigned int flash_addr[]=
{
 0x140000,0x150000,0x160000,0x170000,0x180000,0x190000,0x1a0000,0x1b0000,0x1c0000,0x1d0000,0x1e0000
};
unsigned int sd_addr[]=
{
 720,722,724,726,728,730,732,734,736,738,740
};
int trigger_level=64;
int trigger_sel=1;
unsigned char shift_flag=0;
unsigned char run_stop_flag=255;
unsigned char ch_flag=1;
unsigned char status1_mch_flag=0;
unsigned char status1_mtype_flag=0;
unsigned char status2_mch_flag=0;
unsigned char status2_mtype_flag=1;
unsigned char status3_mch_flag=0;
unsigned char status3_mtype_flag=2;
unsigned char status1_ctype_flag=1;
unsigned char status2_ctype_flag=0;
unsigned char status3_ctype_flag=0;
unsigned char status2_ttype_flag=255;
unsigned char status3_ttype_flag=255;
unsigned char status1_stype_flag=1;
unsigned char status2_stype_flag=0;
unsigned char status3_stype_flag=0;
unsigned char status3_fft_flag=0;
unsigned char status0_zice_flag=6;
unsigned char status1_zice_flag=0;
unsigned char status2_zice_flag=0;
unsigned char status3_zice_flag=1;
unsigned char status4_zice_flag=0;
unsigned char key_measure=255;
unsigned char key_cursor=0;
unsigned char key_save=0;
unsigned char key_trigger=0;
unsigned char key_fft=0;
unsigned char key_dsx=0;
unsigned char key_zice=0;
unsigned char line_T1=54;
unsigned char line_T2=106;
unsigned char line_V1=32;
unsigned char line_V2=96;
int buff1_data[523]={0};
int buff2_data[523]={0};
```

```
int uart_data[1024]={0};
float fft_data[128]={0};
float fft1_data[128]={0};
float fft2_data[128]={0};
unsigned char zice_f[][6]=
{
 “  10Hz”,”  20Hz”,”  50Hz”,” 100Hz”,” 200Hz”,” 500Hz”,
 “  1KHz”,”  2KHz”,”  5KHz”,” 10KHz”,” 20KHz”,” 50KHz”,
 “100KHz”,”200KHz”,”500KHz”
};

unsigned int zice_f_cell[]=
{
 10,20,50,100,200,500,1000,2000,5000,10000,20000,50000,100000,200000,500000
};
unsigned int DIN_zice_f[]=
{
 859,1718,4295,8590,17180,42950,
 85900,171799,429497,858993,1717987,4294967,
 8589935,17179869,42949672
};
unsigned char zice_x[][4]=
{
 “-180”,”-135”,” -90”,” -45”,”   0”,” +45”,” +90”,”+135”,”+180”
};
unsigned int DIN_zice_x[]=
{
 256,320,384,448,0,64,128,192,256
};
float zice_m[10]={1,1,0,0,0,0,0,0,0,0};
unsigned char zice_pv1=10;
unsigned char zice_pv2=0;
unsigned char zice_pv3=0;
unsigned char zice_pv4=0;
unsigned char zice_pv5=0;
unsigned char zice_pv6=0;
unsigned char zice_pv7=0;
unsigned char zice_pv8=0;
unsigned char zice_pv9=0;
unsigned char zice_pv[][4]=
{
 “:0.0”,”:0.1”,”:0.2”,”:0.3”,”:0.4”,”:0.5”,”:0.6”,”:0.7”,”:0.8”,”:0.9”,”:1.0”
};
float zice_pv_cell[]=
{
 0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0
};
/*****************************************************************/
void sys_clk_timer_init();
void sys_clk_timer_int (void* context ,alt_u32 id);
void timer1_init();
void timer1_int (void* context ,alt_u32 id);
void button_init();
void button_irq(void* context,alt_u32 id);
void key_init();
void key_int (void* context,alt_u32 id);
void uart_init(void);
void uart_irq(void *context,alt_u32 interrupt);
void Refresh(void);
void measure_Refresh(unsigned char status);
void cursor_Refresh(void);
void save_Refresh(void);
void trigger_Refresh(void);
void fft_Refresh(void);
void zice_Refresh(void);
void grid_Refresh(void);
int code(int i);
void autoset();
void off_inf();
void on_inf();
```

```
void clear(void);
void f_measure(unsigned char ch);
void pv_measure(unsigned char ch);
void vv_measure(unsigned char ch);
void av_measure(unsigned char ch);
void measure(void);
void measure_disp(void);
void measure123_disp(unsigned char status);
void cursor_disp(void);
void cursor1_disp(void);
void cursor2_disp(void);
void cursor3_disp(void);
void save_disp(void);
void save1_disp(void);
void save2_disp(void);
void save3_disp(void);
void trigger1_disp(void);
void trigger2_disp(void);
void trigger3_disp(void);
void fft(void);
void fft_disp(void);
void fft0_disp(void);
void fft3_disp(void);
void zice_disp(void);
void zice0_disp(void);
void zice1_disp(void);
void zice2_disp(void);
void zice3_disp(void);
/*****************************************************************/
int lcd_busy(void);
void lcd_st01(void);
void lcd_st3(void);
void lcd_ctrl(int dat);
void lcd_write(int dat);
void lcd_autowrite(int dat);
void lcd_initial(void);
void lcd_clear_ram(void);
void lcd_disp8(int dat);
void lcd_disp1608(unsigned int x,unsigned int y,unsigned char n);
void lcd_disp1616(unsigned int x,uint y,unsigned char n);
void lcd_set_graph(unsigned char x,unsigned char y);
void lcd_set_text(unsigned char x,unsigned char y);
void lcd_disp_text(unsigned char x,unsigned char y,unsigned char ccode);
void lcd_sin(void);
void lcd_image();
void Draw_sline(unsigned char x,unsigned char y,unsigned char len);
void Draw_hline(unsigned char x,unsigned char y,unsigned char len);
void Init_CGRAM(void);
void lcd_disp_char(unsigned char x,unsigned char y,unsigned char ccode);
void lcd_disp_string(unsigned char x,unsigned char y,unsigned char *string,unsigned int len);
void lcd_disp_hanzi(unsigned char x,unsigned char y,unsigned char *string,unsigned int len);
void Draw_Grid(void);
void Display_Wave(void);
/*****************************************************************************/
void cepin_start();
void trigger_level_disp();
void ch_f_disp(unsigned char ch,unsigned char y);
void ch_t_disp(unsigned char ch,unsigned char y);
void ch_pv_disp(unsigned char ch,unsigned char y);
void ch_vv_disp(unsigned char ch,unsigned char y);
void ch_av_disp(unsigned char ch,unsigned char y);
void V_disp(unsigned char status);
void T_disp(unsigned char status);
/*****************************************************************************/
//  SD Card Set I/O Direction
#define SD_CMD_IN   IOWR(SD_CMD_BASE, 1, 0)
#define SD_CMD_OUT  IOWR(SD_CMD_BASE, 1, 1)
#define SD_DAT_IN   IOWR(SD_DAT_BASE, 1, 0)
#define SD_DAT_OUT  IOWR(SD_DAT_BASE, 1, 1)
//  SD Card Output High/Low
```

```
#define SD_CMD_LOW  IOWR(SD_CMD_BASE, 0, 0)
#define SD_CMD_HIGH IOWR(SD_CMD_BASE, 0, 1)
#define SD_DAT_LOW  IOWR(SD_DAT_BASE, 0, 0)
#define SD_DAT_HIGH IOWR(SD_DAT_BASE, 0, 1)
#define SD_CLK_LOW  IOWR(SD_CLK_BASE, 0, 0)
#define SD_CLK_HIGH IOWR(SD_CLK_BASE, 0, 1)
//  SD Card Input Read
#define SD_TEST_CMD IORD(SD_CMD_BASE, 0)
#define SD_TEST_DAT IORD(SD_DAT_BASE, 0)
//-----------------------------------------------------------------------
#define BYTE    unsigned char
#define UINT16  unsigned int
#define UINT32  unsigned long
//-----------------------------------------------------------------------
void Ncr(void);
void Ncc(void);
BYTE response_R(BYTE);
BYTE send_cmd(BYTE *);
BYTE SD_read_lba(BYTE *,UINT32,UINT32);
BYTE SD_write_lba(BYTE *buff,UINT32 lba,UINT32 seccnt);
BYTE SD_card_init(void);
void ClusIndex(int FileTotal);
void ReadFile(UINT32 FileNum);
int WriteFile(void);
int fileindex(void);
void FileN(char * FileName,int i,int ChNum);
extern unsigned char in[17280];
extern unsigned char out[17280];
extern int val2,val1;
//-----------------------------------------------------------------------
#endif //_SD_CARD_H_
//-----------------------------------------------------------------------
#define BYTE    unsigned char
#define UINT16  unsigned int
#define UINT32  unsigned long
//-----------------------------------------------------------------------
void Ncr(void);
void Ncc(void);
BYTE response_R(BYTE);
BYTE send_cmd(BYTE *);
BYTE SD_read_lba(BYTE *,UINT32,UINT32);
BYTE SD_card_init(void);
//-----------------------------------------------------------------------
BYTE write_status;
/*************************************************************************/
unsigned char pinlv[]=
{
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x08,0x00,0x08,0xFE,0x4E,0x20,0x48,0x40,0x48,0xFC,0xFE,0x84,0x00,0xA4,0x08,0xA4,
0x4A,0xA4,0x4A,0xA4,0x84,0xA4,0x08,0x50,0x10,0x48,0x20,0x86,0xC3,0x02,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x02,0x00,0x01,0x00,0x7F,0xFE,0x41,0x00,0x22,0x28,0x17,0xD0,0x04,0x80,0x11,0x10,
0x22,0x48,0x47,0xC4,0x01,0x20,0xFF,0xFE,0x01,0x00,0x01,0x00,0x01,0x00,0x01,0x00,
};

unsigned char zhouqi[]=
{
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x1F,0xFC,0x10,0x84,0x13,0xE4,0x10,0x84,0x10,0x84,0x17,0xF4,0x10,0x04,
0x13,0xE4,0x12,0x24,0x12,0x24,0x13,0xE4,0x22,0x24,0x20,0x04,0x40,0x14,0x80,0x08,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x22,0x00,0x22,0x7C,0x7F,0x44,0x22,0x44,0x3E,0x44,0x22,0x7C,0x3E,0x44,0x22,0x44,
0x22,0x44,0xFF,0x7C,0x00,0x44,0x24,0x84,0x22,0x84,0x43,0x14,0x81,0x08,0x00,0x00,
};

unsigned char fengfengzhi[]=
```

```
{
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x20,0x80,0x20,0xFC,0x20,0x88,0xA9,0x88,0xAA,0x50,0xAC,0x20,0xA8,0xDC,0xAB,0x48,
0xA8,0x40,0xAB,0xFC,0xB8,0x40,0xEB,0xFC,0x80,0x40,0x07,0xFE,0x00,0x40,0x00,0x40,
0x20,0x80,0x20,0xFC,0x20,0x88,0xA9,0x88,0xAA,0x50,0xAC,0x20,0xA8,0xDC,0xAB,0x48,
0xA8,0x40,0xAB,0xFC,0xB8,0x40,0xEB,0xFC,0x80,0x40,0x07,0xFE,0x00,0x40,0x00,0x40,
0x10,0x40,0x18,0x60,0x17,0xFC,0x10,0x40,0x20,0x80,0x33,0xF8,0x62,0x08,0xA3,0xF8,
0x22,0x08,0x23,0xF8,0x22,0x08,0x23,0xF8,0x22,0x08,0x22,0x08,0x2F,0xFE,0x20,0x00,
};

unsigned char youxiaozhi[]=
{
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x01,0x00,0x01,0x00,0x7F,0xFE,0x02,0x00,0x04,0x00,0x0F,0xF0,0x18,0x10,0x28,0x10,
0x4F,0xF0,0x88,0x10,0x08,0x10,0x0F,0xF0,0x08,0x10,0x08,0x90,0x08,0x70,0x08,0x20,
0x10,0x20,0x08,0x30,0x08,0x20,0x7F,0x20,0x14,0x7E,0x12,0x44,0x21,0xC4,0x22,0x44,
0x62,0x28,0x14,0x28,0x08,0x10,0x0C,0x10,0x12,0x28,0x20,0xCE,0xC3,0x04,0x00,0x00,
0x10,0x40,0x18,0x60,0x17,0xFC,0x10,0x40,0x20,0x80,0x33,0xF8,0x62,0x08,0xA3,0xF8,
0x22,0x08,0x23,0xF8,0x22,0x08,0x23,0xF8,0x22,0x08,0x22,0x08,0x2F,0xFE,0x20,0x00,
};

unsigned char pingjunzhi[]=
{
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x7F,0xFC,0x01,0x00,0x21,0x10,0x11,0x18,0x09,0x10,0x0D,0x20,0x09,0x40,0x01,0x00,
0xFF,0xFE,0x01,0x00,0x01,0x00,0x01,0x00,0x01,0x00,0x01,0x00,0x01,0x00,0x01,0x00,
0x10,0x80,0x10,0xC0,0x10,0x84,0x10,0xFE,0xFD,0x04,0x11,0x04,0x12,0x84,0x14,0x64,
0x10,0x44,0x16,0x1C,0x18,0x64,0xE3,0x84,0x41,0x04,0x00,0x24,0x00,0x14,0x00,0x08,
0x10,0x40,0x18,0x60,0x17,0xFC,0x10,0x40,0x20,0x80,0x33,0xF8,0x62,0x08,0xA3,0xF8,
0x22,0x08,0x23,0xF8,0x22,0x08,0x23,0xF8,0x22,0x08,0x22,0x08,0x2F,0xFE,0x20,0x00,
};

unsigned char shijian[]=
{
0x00,0x10,0x00,0x10,0x7C,0x10,0x44,0x10,0x47,0xFE,0x44,0x10,0x7C,0x10,0x45,0x10,
0x44,0x90,0x44,0x90,0x7C,0x10,0x00,0x10,0x00,0x10,0x00,0x10,0x00,0x50,0x00,0x20,
0x20,0x00,0x13,0xFC,0x10,0x04,0x40,0x04,0x47,0xE4,0x44,0x24,0x44,0x24,0x47,0xE4,
0x44,0x24,0x44,0x24,0x47,0xE4,0x40,0x04,0x40,0x04,0x40,0x04,0x40,0x14,0x40,0x08,
};

unsigned char dianya[]=
{
0x01,0x00,0x01,0x00,0x01,0x00,0x3F,0xF8,0x21,0x08,0x21,0x08,0x3F,0xF8,0x21,0x08,
0x21,0x08,0x21,0x08,0x3F,0xF8,0x21,0x08,0x01,0x02,0x01,0x02,0x00,0xFE,0x00,0x00,
0x00,0x00,0x3F,0xFE,0x20,0x00,0x20,0x80,0x20,0x80,0x20,0x88,0x2F,0xFC,0x20,0x80,
0x20,0xA0,0x20,0x90,0x20,0x98,0x40,0x90,0x40,0x80,0xBF,0xFE,0x00,0x00,0x00,0x00,
};

unsigned char zhengliang[]=
{
0x22,0x08,0x21,0x8C,0x20,0x90,0x23,0xFE,0xFA,0x22,0x22,0xAA,0x22,0x72,0x23,0xFE,
0x20,0x00,0x21,0xFC,0x39,0x04,0xE1,0xFC,0x41,0x04,0x01,0x04,0x01,0xFC,0x01,0x04,
0x1F,0xF0,0x10,0x10,0x1F,0xF0,0x10,0x10,0x1F,0xF0,0x00,0x00,0xFF,0xFE,0x11,0x10,
0x1F,0xF0,0x11,0x10,0x1F,0xF0,0x01,0x00,0x1F,0xF8,0x01,0x00,0xFF,0xFE,0x00,0x00,
};

unsigned char guangbiao[]=
{
0x01,0x00,0x21,0x10,0x19,0x18,0x0D,0x10,0x09,0x20,0x01,0x04,0x7F,0xFE,0x04,0x40,
0x04,0x40,0x04,0x40,0x04,0x40,0x08,0x42,0x08,0x42,0x10,0x42,0x20,0x3E,0x40,0x00,
0x10,0x00,0x11,0xFC,0x10,0x00,0x10,0x00,0xFC,0x00,0x13,0xFE,0x38,0x20,0x34,0x20,
0x55,0x28,0x51,0x24,0x92,0x26,0x14,0x22,0x10,0x22,0x10,0x20,0x10,0xA0,0x10,0x40,
};


unsigned char wu[]=
{
```

```
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x3F,0xFC,0x01,0x00,0x01,0x00,0x01,0x00,0x01,0x00,0x7F,0xFE,0x01,0x80,
0x02,0x80,0x02,0x80,0x04,0x80,0x08,0x80,0x10,0x82,0x20,0x82,0xC0,0x7E,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
};

unsigned char guanbi[]=
{
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x08,0x20,0x04,0x30,0x06,0x20,0x04,0x48,0x7F,0xFC,0x01,0x00,0x01,0x00,0x01,0x00,
0xFF,0xFE,0x01,0x00,0x02,0x80,0x04,0x40,0x08,0x20,0x10,0x30,0x20,0x1C,0x40,0x08,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x20,0x00,0x1B,0xFC,0x08,0x04,0x20,0x84,0x20,0x84,0x20,0x84,0x2F,0xF4,0x20,0x84,
0x21,0x84,0x22,0x84,0x2C,0x84,0x20,0x84,0x22,0x84,0x21,0x04,0x20,0x14,0x20,0x08,
};

unsigned char jiyixuanzhe[]=
{
0x40,0x00,0x21,0xFC,0x30,0x04,0x20,0x04,0x00,0x04,0x00,0x04,0xF1,0xFC,0x11,0x00,
0x11,0x00,0x11,0x00,0x11,0x00,0x11,0x04,0x15,0x04,0x19,0x06,0x10,0xFC,0x00,0x00,
0x10,0x00,0x10,0x00,0x13,0xFC,0x10,0x04,0x14,0x08,0x52,0x10,0x52,0x20,0x50,0x40,
0x90,0x80,0x11,0x00,0x11,0x00,0x12,0x02,0x12,0x02,0x12,0x02,0x11,0xFC,0x10,0x00,
0x00,0x40,0x22,0x40,0x12,0x40,0x13,0xF8,0x04,0x40,0x00,0x40,0xF7,0xFC,0x11,0x20,
0x11,0x20,0x12,0x24,0x12,0x24,0x14,0x1C,0x10,0x00,0x28,0x00,0x47,0xFE,0x00,0x00,
0x20,0x00,0x23,0xF8,0x21,0x10,0xFC,0xA0,0x20,0x40,0x20,0xA0,0x2B,0x58,0x30,0x46,
0x63,0xF8,0xA0,0x40,0x20,0x40,0x27,0xFE,0x20,0x40,0x20,0x40,0xA0,0x40,0x40,0x40,
};

unsigned char chunchu[]=
{
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x03,0x00,0x02,0x00,0x7F,0xFC,0x04,0x00,0x04,0x00,0x0B,0xF8,0x18,0x10,0x10,0x20,
0x30,0x20,0x57,0xFE,0x90,0x20,0x10,0x20,0x10,0x20,0x10,0x20,0x10,0xA0,0x10,0x40,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x10,0x20,0x18,0x24,0x15,0xFE,0x24,0x24,0x20,0x28,0x5D,0xFE,0xA4,0x20,0x24,0x40,
0x25,0xFC,0x26,0x84,0x24,0xFC,0x24,0x84,0x25,0x84,0x2E,0xFC,0x24,0x84,0x00,0x00,
};

unsigned char diaochu[]=
{
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x43,0xFC,0x22,0x44,0x32,0x44,0x23,0xF4,0x02,0x44,0xE3,0xF4,0x22,0x04,0x22,0xF4,
0x22,0x94,0x22,0x94,0x2A,0xF4,0x32,0x94,0x24,0x04,0x04,0x04,0x08,0x14,0x10,0x08,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x01,0x00,0x01,0x00,0x21,0x04,0x21,0x04,0x21,0x04,0x21,0x04,0x3F,0xFC,0x21,0x04,
0x01,0x00,0x21,0x04,0x21,0x04,0x21,0x04,0x21,0x04,0x3F,0xFC,0x20,0x04,0x00,0x00,

};


unsigned char xinyuan[]=
{
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x08,0x80,0x0C,0x60,0x18,0x40,0x17,0xFE,0x30,0x00,0x33,0xF8,0x50,0x00,0x93,0xF8,
0x10,0x00,0x13,0xF8,0x12,0x08,0x12,0x08,0x12,0x08,0x13,0xF8,0x12,0x08,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x40,0x00,0x27,0xFE,0x24,0x40,0x04,0x80,0x85,0xFC,0x55,0x04,0x15,0xFC,0x15,0x04,
0x25,0xFC,0x24,0x20,0xC4,0xA8,0x44,0xA4,0x49,0x22,0x4A,0x22,0x50,0xA0,0x40,0x40,
```

```
};

unsigned char danci[]=
{
0x08,0x20,0x06,0x30,0x04,0x40,0x3F,0xF8,0x21,0x08,0x3F,0xF8,0x21,0x08,0x21,0x08,
0x3F,0xF8,0x21,0x08,0x01,0x00,0xFF,0xFE,0x01,0x00,0x01,0x00,0x01,0x00,0x01,0x00,
0x01,0x00,0x41,0x00,0x25,0x00,0x25,0xFE,0x2A,0x44,0x0A,0x48,0x14,0x40,0x10,0x40,
0x20,0xC0,0xE0,0xA0,0x41,0x20,0x42,0x10,0x44,0x08,0x18,0x0E,0x60,0x04,0x00,0x00,
};

unsigned char lianxu[]=
{
0x40,0x80,0x20,0x80,0x37,0xFC,0x21,0x00,0x01,0x40,0x02,0x40,0xF7,0xF8,0x12,0x40,
0x10,0x40,0x1F,0xFE,0x10,0x40,0x10,0x40,0x10,0x40,0x28,0x00,0x47,0xFE,0x00,0x00,
0x10,0x40,0x18,0x40,0x11,0xFC,0x20,0x40,0x4B,0xFE,0xF1,0x24,0x10,0xA8,0x22,0xA0,
0x41,0x20,0xF7,0xFE,0x00,0x20,0x00,0x50,0x18,0x48,0xE0,0x84,0x01,0x04,0x00,0x00,

};

unsigned char dianping[]=
{
0x01,0x00,0x01,0x00,0x01,0x00,0x3F,0xF8,0x21,0x08,0x21,0x08,0x3F,0xF8,0x21,0x08,
0x21,0x08,0x21,0x08,0x3F,0xF8,0x21,0x08,0x01,0x02,0x01,0x02,0x00,0xFE,0x00,0x00,
0x7F,0xFC,0x01,0x00,0x21,0x10,0x11,0x18,0x09,0x10,0x0D,0x20,0x09,0x40,0x01,0x00,
0xFF,0xFE,0x01,0x00,0x01,0x00,0x01,0x00,0x01,0x00,0x01,0x00,0x01,0x00,0x01,0x00,
};

unsigned char bianyan[]=
{
0x40,0x80,0x20,0x80,0x30,0x80,0x20,0x80,0x07,0xF8,0x00,0x88,0xF0,0x88,0x10,0x88,
0x11,0x08,0x11,0x08,0x12,0x08,0x14,0x70,0x10,0x20,0x28,0x00,0x47,0xFE,0x00,0x00,

0x20,0x00,0x11,0xF0,0x11,0x10,0x81,0x10,0x49,0x10,0x49,0x10,0x12,0x0E,0x14,0x00,
0x20,0x00,0xE3,0xF8,0x22,0x08,0x22,0x08,0x22,0x08,0x22,0x08,0x23,0xF8,0x22,0x08,
};

unsigned char chufa[]=
{
0x20,0x10,0x3F,0x10,0x21,0x10,0x42,0x10,0x7F,0x7C,0xC9,0x54,0x7F,0x54,0x49,0x54,
0x49,0x54,0x7F,0x7C,0x49,0x10,0x49,0x14,0x49,0x12,0x49,0x1E,0x85,0xF2,0x02,0x00,


0x02,0x00,0x22,0x40,0x22,0x30,0x22,0x10,0x7F,0xFE,0x24,0x00,0x04,0x00,0x07,0xE0,
0x0C,0x20,0x0A,0x20,0x0A,0x20,0x11,0x40,0x20,0x80,0x41,0x60,0x86,0x1C,0x18,0x08,
};

unsigned char pinpufenxi[]=
{
0x08,0x00,0x08,0xFE,0x4E,0x20,0x48,0x40,0x48,0xFC,0xFE,0x84,0x00,0xA4,0x08,0xA4,
0x4A,0xA4,0x4A,0xA4,0x84,0xA4,0x08,0x50,0x10,0x48,0x20,0x86,0xC3,0x02,0x00,0x00,
0x02,0x08,0x41,0x10,0x2F,0xFE,0x31,0x10,0x25,0x14,0x03,0x18,0xEF,0xFE,0x20,0x00,
0x23,0xF8,0x22,0x08,0x22,0x08,0x23,0xF8,0x2A,0x08,0x32,0x08,0x23,0xF8,0x02,0x08,
0x08,0x80,0x0C,0x80,0x08,0x40,0x10,0x20,0x10,0x30,0x20,0x18,0x40,0x0E,0x9F,0xE4,
0x04,0x20,0x04,0x20,0x04,0x20,0x04,0x20,0x08,0x20,0x10,0xA0,0x20,0x40,0x40,0x00,
0x10,0x00,0x10,0x3C,0x11,0xC0,0xFD,0x00,0x11,0x00,0x31,0xFE,0x39,0x10,0x55,0x10,
0x55,0x10,0x91,0x10,0x11,0x10,0x11,0x10,0x12,0x10,0x12,0x10,0x14,0x10,0x10,0x10,
};

unsigned char tongdaoxuanzhe[]=
{

0x40,0x00,0x27,0xF8,0x20,0x90,0x00,0x60,0x07,0xF8,0x04,0x48,0xE7,0xF8,0x24,0x48,
0x24,0x48,0x27,0xF8,0x24,0x48,0x24,0x68,0x24,0x50,0x50,0x00,0x8F,0xFE,0x00,0x00,
0x02,0x10,0x41,0x20,0x2F,0xFC,0x20,0x80,0x01,0x00,0x03,0xF0,0xE2,0x10,0x23,0xF0,
0x22,0x10,0x23,0xF0,0x22,0x10,0x23,0xF0,0x22,0x10,0x50,0x00,0x8F,0xFE,0x00,0x00,
0x00,0x40,0x22,0x40,0x12,0x40,0x13,0xF8,0x04,0x40,0x00,0x40,0xF7,0xFC,0x11,0x20,
0x11,0x20,0x12,0x24,0x12,0x24,0x14,0x1C,0x10,0x00,0x28,0x00,0x47,0xFE,0x00,0x00,
0x20,0x00,0x23,0xF8,0x21,0x10,0xFC,0xA0,0x20,0x40,0x20,0xA0,0x2B,0x58,0x30,0x46,
0x63,0xF8,0xA0,0x40,0x20,0x40,0x27,0xFE,0x20,0x40,0x20,0x40,0xA0,0x40,0x40,0x40,
```

```
};
unsigned char queren[]=
{
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x80,0x00,0xFC,0x7C,0x88,0x11,0x10,0x13,0xFC,0x21,0x24,0x3D,0x24,0x65,0xFC,
0xA5,0x24,0x25,0x24,0x25,0xFC,0x3D,0x24,0x22,0x24,0x22,0x24,0x04,0x14,0x08,0x08,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x40,0x40,0x20,0x40,0x18,0x40,0x10,0x40,0x00,0x40,0x00,0x40,0xF0,0x40,0x10,0xA0,
0x10,0xA0,0x10,0xA0,0x10,0x90,0x11,0x10,0x15,0x08,0x1A,0x08,0x12,0x06,0x04,0x04,
};

unsigned char quxiao[]=
{
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0xFF,0x80,0x22,0x7E,0x22,0x44,0x3E,0x44,0x22,0x44,0x22,0x44,0x3E,0x28,
0x22,0x28,0x22,0x90,0x27,0x10,0x3A,0x28,0xC2,0x46,0x02,0x84,0x02,0x00,0x02,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x20,0x40,0x10,0x40,0x12,0x44,0x81,0x48,0x49,0x50,0x4B,0xFC,0x12,0x04,0x12,0x04,
0x23,0xFC,0x22,0x04,0xE2,0x04,0x23,0xFC,0x22,0x04,0x22,0x04,0x22,0x14,0x22,0x08,
};

unsigned char yuzhipinlv[]=
{
0x00,0x00,0x7D,0xFE,0x04,0x20,0x28,0x40,0x10,0xFC,0xFE,0x84,0x14,0xA4,0x10,0xA4,
0x10,0xA4,0x10,0xA4,0x10,0xA4,0x10,0x50,0x10,0x48,0x50,0x86,0x23,0x02,0x00,0x00,
0x3F,0xF8,0x24,0x48,0x24,0x48,0x3F,0xF8,0x01,0x00,0x7F,0xFC,0x02,0x00,0x1F,0xF0,
0x10,0x10,0x1F,0xF0,0x10,0x10,0x1F,0xF0,0x10,0x10,0x1F,0xF0,0x10,0x10,0xFF,0xFE,
0x08,0x00,0x08,0xFE,0x4E,0x20,0x48,0x40,0x48,0xFC,0xFE,0x84,0x00,0xA4,0x08,0xA4,
0x4A,0xA4,0x4A,0xA4,0x84,0xA4,0x08,0x50,0x10,0x48,0x20,0x86,0xC3,0x02,0x00,0x00,


0x02,0x00,0x01,0x00,0x7F,0xFE,0x41,0x00,0x22,0x28,0x17,0xD0,0x04,0x80,0x11,0x10,
0x22,0x48,0x47,0xC4,0x01,0x20,0xFF,0xFE,0x01,0x00,0x01,0x00,0x01,0x00,0x01,0x00,

};


unsigned char yuzhixiangwei[]=
{

0x00,0x00,0x7D,0xFE,0x04,0x20,0x28,0x40,0x10,0xFC,0xFE,0x84,0x14,0xA4,0x10,0xA4,
0x10,0xA4,0x10,0xA4,0x10,0xA4,0x10,0x50,0x10,0x48,0x50,0x86,0x23,0x02,0x00,0x00,


0x3F,0xF8,0x24,0x48,0x24,0x48,0x3F,0xF8,0x01,0x00,0x7F,0xFC,0x02,0x00,0x1F,0xF0,
0x10,0x10,0x1F,0xF0,0x10,0x10,0x1F,0xF0,0x10,0x10,0x1F,0xF0,0x10,0x10,0xFF,0xFE,

0x10,0x00,0x10,0xFC,0x10,0x84,0x10,0x84,0xFE,0x84,0x18,0xFC,0x34,0x84,0x32,0x84,
0x52,0x84,0x50,0xFC,0x90,0x84,0x10,0x84,0x10,0x84,0x10,0x84,0x10,0xFC,0x10,0x84,


0x09,0x00,0x0C,0xC0,0x10,0x40,0x17,0xFC,0x20,0x00,0x34,0x10,0x52,0x18,0x52,0x10,
0x91,0x10,0x11,0x20,0x11,0x20,0x10,0x20,0x10,0x40,0x1F,0xFE,0x10,0x00,0x00,0x00,

};

unsigned char zhengxuanbo[]=
{
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x7F,0xFE,0x00,0x80,0x00,0x80,0x00,0x80,0x00,0x80,0x10,0x80,0x10,0xFC,
0x10,0x80,0x10,0x80,0x10,0x80,0x10,0x80,0x10,0x80,0xFF,0xFE,0x00,0x00,0x00,0x00,
0x00,0x40,0xF8,0x20,0x0B,0xFE,0x08,0x00,0x78,0x40,0x48,0x48,0x40,0x8C,0x41,0x78,
0xFB,0xD0,0x48,0x20,0x08,0x40,0x08,0x88,0x09,0x04,0x4B,0xFE,0x29,0x04,0x10,0x00,
0x20,0x40,0x10,0x40,0x10,0x40,0x07,0xFE,0x84,0x44,0x54,0x40,0x54,0x40,0x17,0xF8,
0x25,0x08,0x24,0x90,0xE4,0x90,0x24,0x60,0x28,0x60,0x28,0x98,0x31,0x0E,0x26,0x04,
```

```
};

unsigned char fangbo[]=
{
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x02,0x00,0x01,0x00,0x00,0x80,0xFF,0xFE,0x02,0x00,0x02,0x00,0x03,0xF0,0x02,0x10,
0x04,0x10,0x04,0x10,0x08,0x10,0x08,0x10,0x10,0x10,0x20,0x90,0xC0,0x60,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x20,0x40,0x10,0x40,0x10,0x40,0x07,0xFE,0x84,0x44,0x54,0x40,0x54,0x40,0x17,0xF8,
0x25,0x08,0x24,0x90,0xE4,0x90,0x24,0x60,0x28,0x60,0x28,0x98,0x31,0x0E,0x26,0x04,
};

unsigned char sanjiaobo[]=
{
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x7F,0xFC,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x3F,0xF8,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x7F,0xFE,0x00,0x00,0x00,0x00,


0x04,0x00,0x07,0xF8,0x04,0x10,0x08,0x20,0x1F,0xFC,0x30,0x84,0x50,0x84,0x1F,0xFC,
0x10,0x84,0x10,0x84,0x1F,0xFC,0x10,0x84,0x10,0x84,0x20,0x84,0x20,0x94,0x40,0x88,
0x20,0x40,0x10,0x40,0x10,0x40,0x07,0xFE,0x84,0x44,0x54,0x40,0x54,0x40,0x17,0xF8,
0x25,0x08,0x24,0x90,0xE4,0x90,0x24,0x60,0x28,0x60,0x28,0x98,0x31,0x0E,0x26,0x04,
};

unsigned char tongdao1[]=
{
0x40,0x00,0x27,0xF8,0x20,0x90,0x00,0x60,0x07,0xF8,0x04,0x48,0xE7,0xF8,0x24,0x48,
0x24,0x48,0x27,0xF8,0x24,0x48,0x24,0x68,0x24,0x50,0x50,0x00,0x8F,0xFE,0x00,0x00,
0x02,0x10,0x41,0x20,0x2F,0xFC,0x20,0x80,0x01,0x00,0x03,0xF0,0xE2,0x10,0x23,0xF0,
0x22,0x10,0x23,0xF0,0x22,0x10,0x23,0xF0,0x22,0x10,0x50,0x00,0x8F,0xFE,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x04,0x7F,0xFE,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
};

unsigned char tongdao2[]=
{
0x40,0x00,0x27,0xF8,0x20,0x90,0x00,0x60,0x07,0xF8,0x04,0x48,0xE7,0xF8,0x24,0x48,
0x24,0x48,0x27,0xF8,0x24,0x48,0x24,0x68,0x24,0x50,0x50,0x00,0x8F,0xFE,0x00,0x00,
0x02,0x10,0x41,0x20,0x2F,0xFC,0x20,0x80,0x01,0x00,0x03,0xF0,0xE2,0x10,0x23,0xF0,
0x22,0x10,0x23,0xF0,0x22,0x10,0x23,0xF0,0x22,0x10,0x50,0x00,0x8F,0xFE,0x00,0x00,
0x00,0x00,0x00,0x10,0x3F,0xF8,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x04,0x7F,0xFE,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
};


unsigned char bianjiboxing[]=
{
0x10,0x80,0x18,0x40,0x23,0xFC,0x22,0x04,0x4A,0x04,0xFB,0xFC,0x12,0x00,0x23,0xFC,
0x7B,0x54,0x03,0x54,0x05,0xFC,0x35,0x54,0xC5,0x54,0x09,0x54,0x11,0x0C,0x00,0x00,
0x20,0x00,0x21,0xF8,0xFD,0x08,0x21,0xF8,0x20,0x00,0x53,0xFE,0x51,0x08,0xFD,0xF8,
0x11,0x08,0x11,0xF8,0x3D,0x08,0xD1,0xFE,0x17,0x08,0x10,0x08,0x10,0x08,0x10,0x08,
0x20,0x40,0x10,0x40,0x10,0x40,0x07,0xFE,0x84,0x44,0x54,0x40,0x54,0x40,0x17,0xF8,
0x25,0x08,0x24,0x90,0xE4,0x90,0x24,0x60,0x28,0x60,0x28,0x98,0x31,0x0E,0x26,0x04,
0x00,0x04,0x7F,0x86,0x12,0x0C,0x12,0x10,0x12,0x20,0x12,0x08,0xFF,0xCC,0x12,0x18,
0x12,0x20,0x12,0x44,0x12,0x86,0x12,0x0C,0x22,0x10,0x22,0x20,0x42,0x40,0x80,0x80,
};

unsigned char jibo[]=
{
0x08,0x20,0x08,0x20,0x7F,0xFC,0x08,0x20,0x0F,0xE0,0x08,0x20,0x0F,0xE0,0x08,0x20,
0xFF,0xFE,0x08,0x20,0x11,0x18,0x3F,0xEE,0xC1,0x04,0x01,0x00,0x7F,0xFC,0x00,0x00,
0x20,0x40,0x10,0x40,0x10,0x40,0x07,0xFE,0x84,0x44,0x54,0x40,0x54,0x40,0x17,0xF8,
0x25,0x08,0x24,0x90,0xE4,0x90,0x24,0x60,0x28,0x60,0x28,0x98,0x31,0x0E,0x26,0x04,
};
```

```
unsigned char xiebo[]=
{
0x44,0x20,0x24,0x20,0x34,0x2C,0x27,0xB0,0x04,0x22,0x05,0x22,0xE6,0x5E,0x24,0x80,
0x23,0xF8,0x22,0x08,0x22,0x08,0x23,0xF8,0x2A,0x08,0x32,0x08,0x23,0xF8,0x00,0x00,
0x20,0x40,0x10,0x40,0x10,0x40,0x07,0xFE,0x84,0x44,0x54,0x40,0x54,0x40,0x17,0xF8,
0x25,0x08,0x24,0x90,0xE4,0x90,0x24,0x60,0x28,0x60,0x28,0x98,0x31,0x0E,0x26,0x04,
};


unsigned char hz_clear[]=
{
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

};

unsigned char asc8[]={
//16*8
//;-- 0 -- ** ▯▯ , 12 **
0x00,0x00,0x00,0x18,0x24,0x42,0x42,0x42,0x42,0x42,0x42,0x42,0x24,0x18,0x00,0x00,
//;-- 1 -- ** ▯▯ , 12 **
0x00,0x00,0x00,0x10,0x70,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x7C,0x00,0x00,
//;-- 2 -- ** ▯▯ , 12 **
0x00,0x00,0x00,0x3C,0x42,0x42,0x42,0x04,0x04,0x08,0x10,0x20,0x42,0x7E,0x00,0x00,
//;-- 3 -- ** ▯▯ , 12 **
0x00,0x00,0x00,0x3C,0x42,0x42,0x04,0x18,0x04,0x02,0x02,0x42,0x44,0x38,0x00,0x00,
//;-- 4 -- ** ▯▯ , 12 **
0x00,0x00,0x00,0x04,0x0C,0x14,0x24,0x24,0x44,0x44,0x7E,0x04,0x04,0x1E,0x00,0x00,
//;-- 5 -- ** ▯▯ , 12 **
0x00,0x00,0x00,0x7E,0x40,0x40,0x40,0x58,0x64,0x02,0x02,0x42,0x44,0x38,0x00,0x00,
//;-- 6 -- ** ▯▯ , 12 **
0x00,0x00,0x00,0x1C,0x24,0x40,0x40,0x58,0x64,0x42,0x42,0x42,0x24,0x18,0x00,0x00,
//;-- 7 -- ** ▯▯ , 12 **
0x00,0x00,0x00,0x7E,0x44,0x44,0x08,0x08,0x10,0x10,0x10,0x10,0x10,0x10,0x00,0x00,
//;-- 8 -- ** ▯▯ , 12 **
0x00,0x00,0x00,0x3C,0x42,0x42,0x42,0x24,0x18,0x24,0x42,0x42,0x42,0x3C,0x00,0x00,
//;-- 9 -- ** Song typeface, 12 **
0x00,0x00,0x00,0x18,0x24,0x42,0x42,0x42,0x26,0x1A,0x02,0x02,0x24,0x38,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
};


unsigned int mCGRAM[]=
{
        0xff,0x88,0x80,0x80,0xc0,0x80,0x80,0x80,    //0xc0
        0xff,0x88,0x00,0x00,0x00,0x00,0x00,0x00,    //0xc1
        0xff,0x88,0x00,0x00,0x80,0x00,0x00,0x00,    //0xc2
        0xff,0x88,0x00,0x00,0x01,0x00,0x00,0x00,    //0xc3
        0x80,0x80,0x80,0x80,0x80,0x80,0x80,0x80,    //0xc4
        0xc0,0x80,0x80,0x80,0xc0,0x80,0x80,0x80,    //0xc5
        0x80,0x00,0x00,0x00,0x80,0x00,0x00,0x00,    //0xc6
        0x01,0x00,0x00,0x00,0x01,0x00,0x00,0x00,    //0xc7
        0xc8,0x80,0x80,0x80,0xc0,0x80,0x80,0x80,    //0xc8
        0x88,0x00,0x00,0x00,0x00,0x00,0x00,0x00,    //0xc9
        0x88,0x00,0x00,0x00,0x80,0x00,0x00,0x00,    //0xca
        0x89,0x00,0x00,0x00,0x01,0x00,0x00,0x00,    //0xcb
        0xc0,0x80,0x80,0x80,0xc0,0x80,0x88,0xff,    //0xcc
        0x00,0x00,0x00,0x00,0x00,0x00,0x88,0xff,    //0xcd
        0x80,0x00,0x00,0x00,0x80,0x00,0x88,0xff,    //0xce
```

```
    0x01,0x00,0x00,0x00,0x01,0x00,0x88,0xff,    //0xcf
    0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,    //0xd0
    0x80,0x00,0x00,0x80,0x00,0x00,0x00,0x88,    //0xd1
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x88,    //0xd2
    0xc0,0x80,0x80,0x80,0xc0,0x80,0x80,0x88,    //0xd3
    0x01,0x00,0x00,0x00,0x01,0x00,0x00,0x88,    //0xd4
    0x88,0x88,0x00,0x00,0x00,0x00,0x00,0x00,    //0xd5
    0x88,0x88,0x00,0x00,0x80,0x00,0x00,0x00,    //0xd6
    0xc8,0x88,0x80,0x80,0xc0,0x80,0x80,0x80,    //0xd7
    0x89,0x88,0x00,0x00,0x01,0x00,0x00,0x00,    //0xd8
    0x01,0x00,0x00,0x00,0x01,0x00,0x00,0x00,    //0xd9
    0x89,0x00,0x00,0x00,0x01,0x00,0x00,0x00,    //0xda
    0xff,0x88,0x00,0x00,0x01,0x00,0x00,0x00,    //0xdb
    0x01,0x00,0x00,0x00,0x01,0x00,0x88,0xff,    //0xdc
    0xc0,0x00,0x00,0x00,0xc0,0x00,0x00,0x00,    //0xdd
    0xc8,0x00,0x00,0x00,0xc0,0x00,0x00,0x00,    //0xde
    0xff,0x88,0x00,0x00,0xc0,0x00,0x00,0x00,    //0xdf
    0xc0,0x00,0x00,0x00,0xc0,0x00,0x88,0xff,    //0xe0
    0x01,0x00,0x00,0x00,0x01,0x00,0x00,0x88,    //0xe1
    0x89,0x88,0x00,0x00,0x01,0x00,0x00,0x00,    //0xe2
    0xc0,0x00,0x00,0x00,0xc0,0x00,0x00,0x88,    //0xe3
    0xc8,0x88,0x00,0x00,0xc0,0x00,0x00,0x00,    //0xe4
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,    //0xe5
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,    //0xe6
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,    //0xe7
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,    //0xe8
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,    //0xe9
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,    //0xea
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,    //0xeb
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,    //0xec
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,    //0xed
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,    //0xee
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,    //0xef
    0xff,0x00,0x00,0x00,0x00,0x00,0x00,0x00,    //0xf0
    0x00,0xff,0x00,0x00,0x00,0x00,0x00,0x00,    //0xf1
    0x00,0x00,0xff,0x00,0x00,0x00,0x00,0x00,    //0xf2
    0x00,0x00,0x00,0xff,0x00,0x00,0x00,0x00,    //0xf3
    0x00,0x00,0x00,0x00,0xff,0x00,0x00,0x00,    //0xf4
    0x00,0x00,0x00,0x00,0x00,0xff,0x00,0x00,    //0xf5
    0x00,0x00,0x00,0x00,0x00,0x00,0xff,0x00,    //0xf6
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xff,    //0xf7
    0x80,0x80,0x80,0x80,0x80,0x80,0x80,0x80,    //0xf8
    0x40,0x40,0x40,0x40,0x40,0x40,0x40,0x40,    //0xf9
    0x20,0x20,0x20,0x20,0x20,0x20,0x20,0x20,    //0xfa
    0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,    //0xfb
    0x08,0x08,0x08,0x08,0x08,0x08,0x08,0x08,    //0xfc
    0x04,0x04,0x04,0x04,0x04,0x04,0x04,0x04,    //0xfd
    0x02,0x02,0x02,0x02,0x02,0x02,0x02,0x02,    //0xfe
    0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,    //0xff

};
```