CLAIMS

What is claimed is:

1. A method for controlling a focused ultrasound therapy device based on ARM architecture, the focused ultrasound therapy device being connected to a workstation and an ARM embedded processor running a Linux operating system, the ARM embedded processor storing a first control program, the workstation storing a second control program, the first control program communicating with the second control program via TCP/IP protocol, the method comprising:

the first control program controlling the focused ultrasound therapy device by executing a task and sending a task execution result to the second control program;

wherein the task comprises an internal event and an operation instruction, the internal event comprises a timer event of the first control program and an interrupt message of a Linux kernel, the second control program receives the operation instruction through the workstation and sends the operation instruction to the first control program;

wherein, the first control program comprises a main thread and a working thread, the main thread makes priorities for the task, priorities of the interrupt message, the operation instruction and the timer event decrease in order, and the working thread executes the task according to the priorities, and reads and writes a register of the ARM embedded processor according to the task execution result to control the focused ultrasound therapy device, and the main thread sends the task execution result to the second control program; and

wherein the first control program further comprises a signal thread, the task comprises an exit message, the exit message has a priority between the interrupt message and the operation instruction, the signal thread sends the exit message to the main thread by closing a socket, and the operation instruction comprises a control instruction and a status query instruction.

2. The method according to claim 1, wherein the signal thread closes the socket when receiving an exit signal, the exit signal comprises SIGNINT and SIGTERM.

3. The method according to claim 1, wherein the socket is a Unix Socket Pair.

4. The method according to claim 1, wherein the main thread sends a Linux signal to the working thread when receiving the interrupt message of the Linux kernel; and

the working thread interrupts a current task after receiving the Linux signal and executes an interrupt response processing function.

5. The method according to claim 4, wherein the Linux signal is SIGUSR1.

6. The method according to claim 1, wherein a protocol between the first control program and the second control program is encapsulated by a Google Protocol Buffer.

7. The method according to claim 1, wherein the first control program receives the interrupt message of the Linux kernel by Netlink.

8. The method according to claim 1, wherein the main thread receives the task via a libevent library.

9. The method according to claim 1, wherein the first control program uses a C++ abstraction mechanism to encapsulate the task.