NITRO-SDK

About Components

Version 1.0.2

The contents in this document are highly confidential and should be handled accordingly.

NTR-06-0338-001-A2
Released: November 22, 2005

Table of Contents

Revision History

Version	Revision Date	Description
1.0.2	09/01/2005	• Specified in the summary description that components cannot be dynamically switched during program execution.
1.0.1	06/09/2005	• Revised the description to note that the SND library is inserted in the SDK. • Revised the location of the program on the ARM7 side to be address 27e0000.
1.0.0	03/11/2004	Initial version.

1 Overview of Components

1.1 The NITRO-SDK Components

In the NITRO-SDK, "components" are programs that execute on the ARM7 processor. Applications are executed on the ARM9 processor, so components running on the ARM7 are primarily tasked with controlling devices.

The ARM9 controls the operations of the components using FIFO communications between the ARM9 and the ARM7. The NITRO-SDK provides a library of functions for controlling the components for every device, so the application does not need to directly communicate with the ARM7 through FIFO communications.

Although multiple components are provided with NITRO-SDK, a single component must be selected when an application is created. ARM7 operates the selected component, and components cannot be switched while the application is running.

1.2 Devices Controlled by Components

This section explains the devices that are controlled by components and the libraries that are provided to operate these devices from the ARM9.

1.2.1 Digital Buttons

In the NITRO hardware, the states of the X Button, the Y Button, and the Open/Close Detector Button cannot be directly read from the ARM9. For this reason, the states of these buttons are monitored by components. The PAD library provides functions to read the states of the buttons.

1.2.2 Sound Circuitry

In the NITRO hardware, the sound output circuitry cannot be directly accessed from the ARM9, so the sound circuit is controlled with components instead. The ARM9 library is provided as the SND library.

1.2.3 SPI Devices

In the NITRO hardware, the ARM9 cannot directly access SPI devices. Moreover, multiple SPI devices cannot be accessed at the same time. There are four SPI devices -- the microphone, the touch panel, the power-management chip and the built-in flash memory -- and components are used to manage access to each of these.

The component that manages access to the microphone works to make sure that continuous sampling is performed with the correct periodicity. The MIC library provides the necessary functions for sampling the microphone.

Similarly, for the touch panel there is a component that makes sure that continuous sampling of the touch panel is performed with the correct periodicity. The functions for this are provided by the TP library.

The various capabilities of the power-management chip are collected in the PM library.

Although there is a component for reading and writing to the built-in flash memory, there is no API available that gives the ARM9 the ability to control these features. However, the OS library supports reading of owner information and some of the other data stored in flash.

1.2.4 Realtime Clock

In the NITRO hardware, the realtime clock cannot be directly accessed from the ARM9, so it is controlled with components instead. The RTC library provides the functions for controlling the realtime clock.

1.2.5 Wireless Communications Module

In the NITRO hardware, the ARM9 cannot directly access the wireless communications module. Instead, components control the wireless communications module, manage the timing required by the communications protocol, and manage all the various core control tasks related to wireless communications, including synchronizing auto-V-Blanks with peers. The WM library provides the functions for the ARM9 to control wireless communications.

2 The Types of Components

The NITRO-SDK now provides two kinds of components: *mongoose* components and *ichneumon* components. The *idle* and *ferret* components that had been available up through version NITRO-SDK 2.0 RC4 were discarded in NITRO-SDK 2.0 RC5. This section describes the two kinds of components that are now available.

2.1 mongoose

These are the set of default components for controlling the devices described in section 1.2.

The majority of the execution code for the functions relating to wireless communications is located in main memory. As a result, main memory is accessed frequently when performing processes related to communications.

When the ARM7 is accessing main memory, access from the ARM9 and DMA are forced to wait. Thus (assuming that the right of priority to access main memory is set to the default), as the frequency of communications rises and the volume of communications data increases, the ARM9 has a smaller and smaller window of opportunity to access main memory. In special situations, such as when making a connection where some continuous operation is processing, the ARM9 may not be able to access main memory for as long as several picture frames.

2.2 ichneumon

These components control all of the devices described in section 1.2, but they have special features relating to wireless communications. They differ from the *mongoose* components in that the majority of the execution code for the functions relating to wireless communications is located in VRAM. As a result, the ARM7 accesses main memory during communications-related processing far less frequently with *ichneumon* components than it does with *mongoose* components.

When using the *ichneumon* components, you need to allocate at the very least either VRAM-C or VRAM-D to the ARM7 when using the wireless communications feature. If neither one of these regions is allocated to the ARM7, the program image that should be allocated to VRAM is saved to the system reserve region of main memory. Therefore, when the wireless communications feature is not being used, the ARM9 can use VRAM for other purposes.

The wireless driver control library (the WVR library) provides the ability to change the VRAM allocation state and thereby toggle between enabling and disabling the wireless communications feature.

3 Selecting a Component

By using the `makerom` tool to combine an ARM9 execution image (application) and an ARM7 execution image (component) in a single binary file, you can specify the use of any component. What you do is specify the Arm7 section of the `rsf` file handed to `makerom.exe`. For details, see the `makerom` explanation in the Reference Manual.

If you are using the command line build environment with the `rsf` file in the default state, you can select which component to use by explicitly describing the path to that component in the `MAKEROM_ARM7` variable of the application's `Makefile`. Unless specified otherwise, the *mongoose* component will be used by default.

4 Determining the Component Type

When the application executes, the following method is used to determine the component type.

1. If, after the `PXI_Init` function is called, `PXI_IsArm7CallbackReady(PXI_FIFO_TAG_WVR)` is called and returns TRUE, then the component is an *ichneumon* component.
2. If FALSE is returned in step 1, then call `PXI_IsArm7CallbackReady(PXI_FIFO_TAG_WM)`. If this returns TRUE, then the component is a *mongoose* component.
3. If FALSE is returned in step 2, then the component is some other type of component. It might be one of the types of components that existed in version NITRO-SDK 2.0 RC4 or earlier.

5 Precautions

To execute a component, there must be enough free space in memory to place the program itself. The NITRO hardware includes dedicated work RAM for the ARM7, but this work RAM is not sufficient to hold the program, so the NITRO-SDK allocates to ARM7 the ARM9-ARM7 shared work RAM (16KB x 2) where the program is placed. Do not allocate this shared work RAM to the ARM9.

Furthermore, simply allocating work RAM is not enough to place the wireless communications control program in memory. Memory space secured as a system reserve region in main memory gets used by the NITRO-SDK as the region for placing the ARM7 program. Do not use this system reserve region (0x027e0000 to 0x02800000) for other purposes. The NITRO-SDK places data TCM in the ARM9 memory space 0x27e0000 by default.