Implementing In-Application Programming on the ADuC702x

By Johnson Jiao [Johnson.Jiao@analog.com] and Raven Xue [Raven.Xue@analog.com]

Background

The ADuC702x Precision Analog Microcontroller provides a serial downloader for loading an assembled program into the on-chip program Flash/EE memory via the serial port on a standard PC, but entering this mode requires a user to manually tie the BM pin low while the ADuC702x is reset or powered on. Users who want to upgrade the program in an industrial environment will not normally have access to this pin, making it difficult to update the application program in this way. In-application programming (IAP) allows upgrades without touching the board. This article provides an easy method for ADuC702x users to upgrade their programs.

IAP function methodology

The ADuC702x's factory configured boot-loader is located in the upper end of Flash memory, from 0x0008F800 to 0x0008FFFF. The ADuC702x will automatically enter this area after a reset or power-up cycle if the BM pin is pulled low. Normally, however, the BM pin is tied high after all programs have been downloaded into the FLASH/EE, so new programs cannot be downloaded without changing the BM logic.

To introduce an IAP function that can re-download assembled application code to the desired flash address range without changing the hardware status of the BM pin, three programs must be located in certain Flash memory addresses from 0x00080000 to 0x0008FFFF, as shown below.

(1). **0x00080000 to 0x00080FFF, IAP function program.** Think of this as a user-designed boot-loader. The user can change the end address according to the IAP function program size.
(2). **0x00081000 to 0x0008F7FF, User Application program.** This is the actual application designed by the user. When the IAP function is running, this area can be replaced by a new application program.
(3). **0x0008F800 to 0x0008FFFF, Factory Configured Boot-Loader.** This code is hidden and cannot be accessed by the user.

Once the user has downloaded the IAP and application programs into the desired areas as shown above, operation proceeds as follows. When a normal reset (BM pin is high) occurs, execution automatically starts in the factory programmed internal configuration code. Next, the ADuC702x jumps to the Interrupt vector address, 0x00000000, to execute the user's reset exception routine. By default, the Flash/EE is mirrored at the bottom of the memory array.

Next, ADuC702x will execute the IAP function program. This will initialize hardware configurations for the UART and timer, and will enable UART receive and transmit interrupts and a five second timer interrupt. During this time, the ADuC702x will wait five seconds for

the host computer to command an upgrade. If the ADuC702x doesn't receive this command within five seconds, the pointer will jump to the user application location at 0x00081040 to execute the user application program; this is a normal reset action. If the ADuC702x receives a command, 0xA, for example, within five seconds, it will continue executing the IAP function program, copy the IAP program into SRAM, and enable REMAP function. The interrupt vector and the upgrade function are both in SRAM. During download, the Flash memory will provide storage. The ADuC702x is now expecting to receive the package message from the host computer, and save the new code in the Flash user application area from 0x00081000 to 0x0008F7FF. When the download is complete, the IAP function will reset the entire chip. If no further command is received from the UART, the pointer will jump to the upgraded user application program after five seconds.

The flowchart is shown in Figure 1:

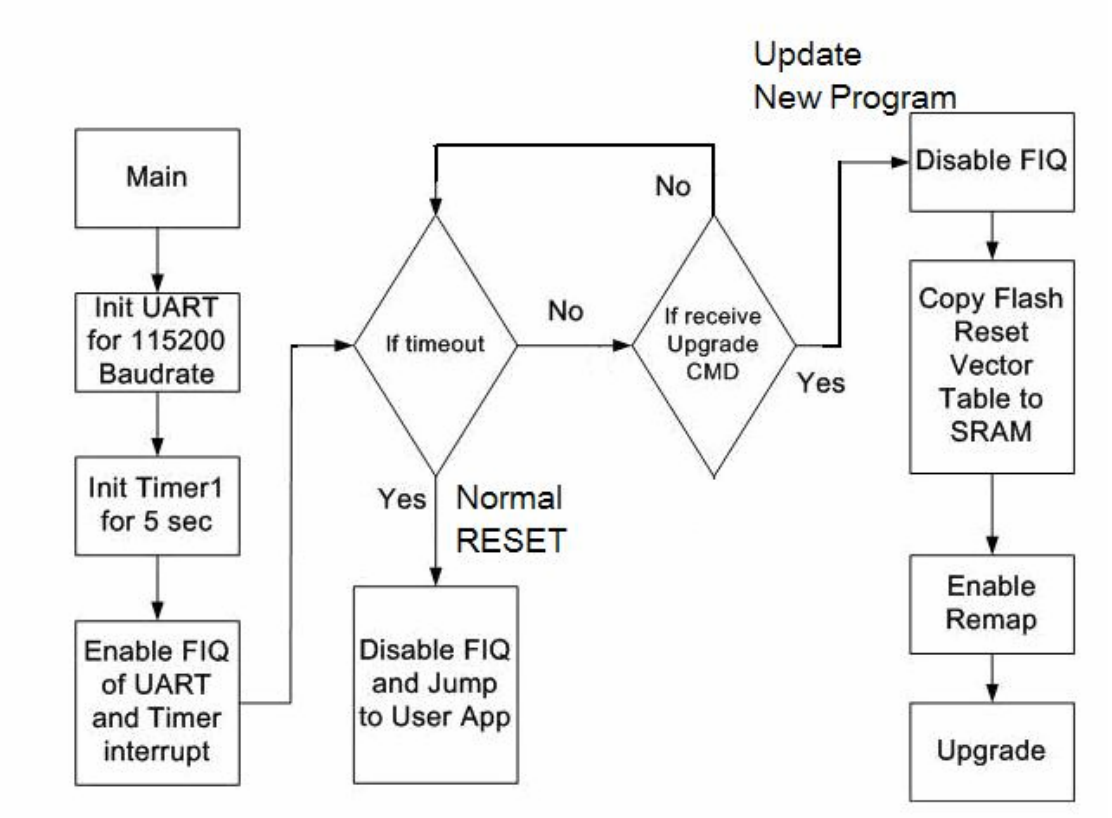

Figure 1. Main Program Flowchart

As shown in Figure 2, two types of packages can be sent from the host computer to the IAP program: a full package, where the first byte is counter value 0xFF and the following 256 bytes are content; or a smaller package, where the first byte stands for the length of the package and the following bytes are content. If, for example, the counter value is 0x10, the package has 16 bytes.

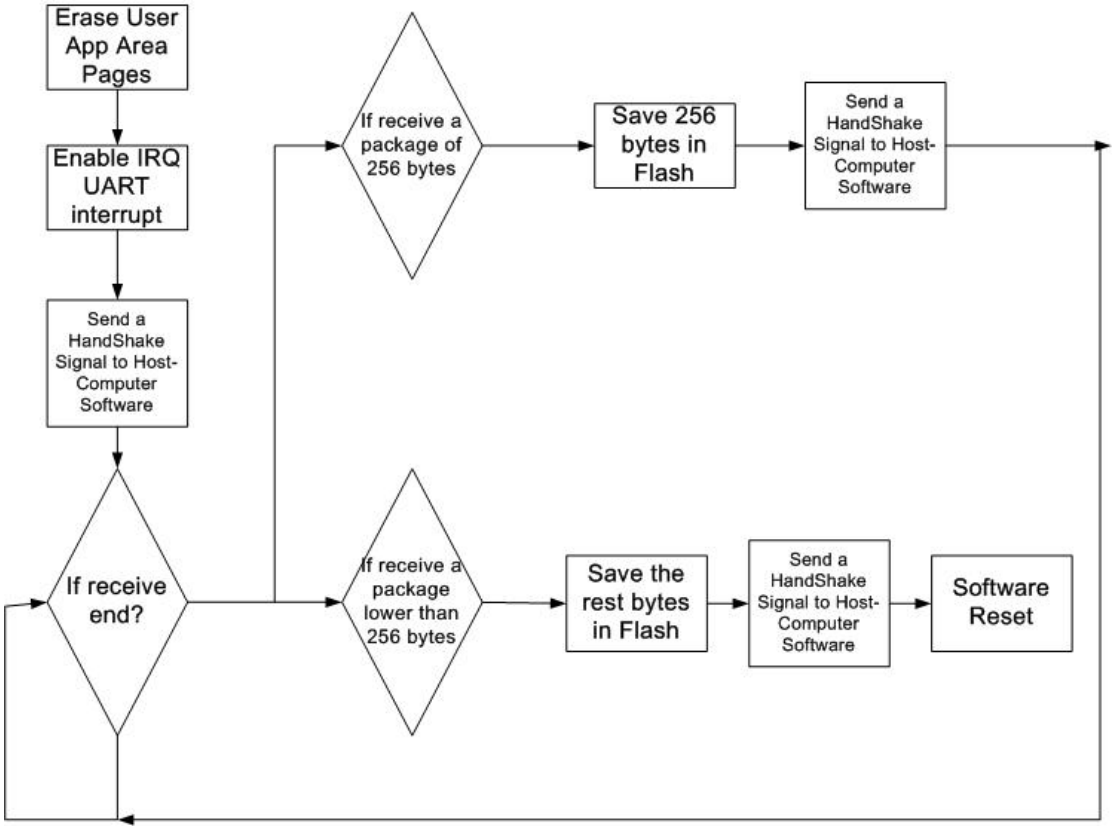

Figure 2. Upgrade Function Flow Chart

Some important precautions for the ARM7TDMI IAP function follow.

Interrupt Vector Table and Remap

ARM interrupt vectors are mirrored at the bottom of the memory array, from 0x00000000 to 0x0000001F. Linked interrupt ISR address are located from 0x00000020 to 0x0000003F. By default, the Flash/EE is mirrored at the bottom of the memory array when a reset occurs. The core will jump to Flash to search the ISR entry when an interrupt is triggered. In upgrade mode, however, the Flash acts as a storage memory that will be erased and rewritten, so locating interrupt vectors in Flash will result in an error. To avoid this problem, the user must use the REMAP function, which mirrors the SRAM at the bottom of the memory array. This facilitates execution of interrupt routines from SRAM instead of from Flash. To use the REMAP function, the user must first copy the interrupt vector table from Flash to SRAM, and the first 64 bytes in SRAM can't be used as normal. See Figure 3.

Function run in SRAM

IAP provides the functionality to erase and rewrite Flash without pulling the BM pin low, but core operation is waiting until the requested read or erase mode is completed. To solve this problem, put all the function including ISR into SRAM and execute from there. With Keil uVision3 v3.23, the user could execute the function in SRAM as follows:

```
void Ram_Function() __ram       // __ram attribute indicates code is is placed in RAM
```

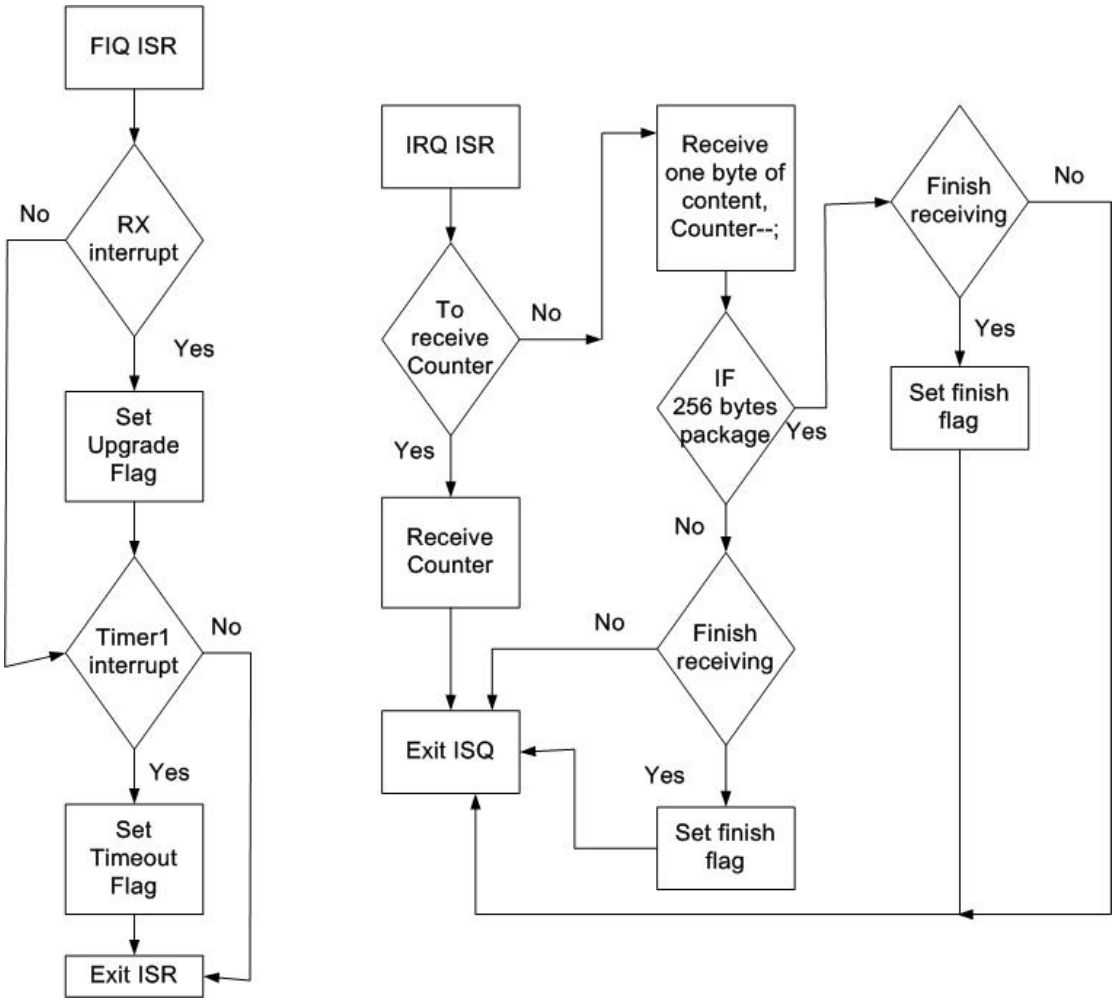

Figure 3. FIQ and IRQ ISR Flowchart

Locate RO and RW segment in desired area.

In "Project" → "Options for Target 'IAP_boot'"→ LA Locate, the user could deselect "Use Memory Layout from Target Dialog", and assign RO and RW segment in "User Classes" as below:

DATA (0x10040-0x11FFF),
CODE (0x80000-0x80FFF), CONST (0x80000-0x80FFF), ERAM(0x10040-0x11FFF)

By this assignment, the area from 0x10040 to 0x11FFF can be used as an RW segment, the area from 0x80000 to 0x80FFF can be used as an RO segment, and the function can be run in SRAM at locations from 0x10040 to 0x11FFF.

User Application arrangement.

In the user's application, RO and RW should be assigned to the segment reasonably.

In "Project" → "Options for Target 'IAP_boot'"→ LA Locate, user could de-select "Use Memory Layout from Target Dialog", and assign RO and RW segment in "User Classes" as below:

DATA (0x10040-0x11FFF),
CODE (0x81000-0x8F7FF), CONST (0x81000-0x8F7FF)

By this assignment, the area from 0x10040 to 0x11FFF can be used as an RW segment, and the area from 0x81000 to 0x8F7FF can be used as an RO segment. Please note that the startup code should be modified to assign the startup object code memory area. This can be modified in startup code as shown:

AREA STARTUPCODE, CODE, AT 0x00081000

The startup object code will be located from 0x00081000. In the main program, the user must first copy the interrupt vector table from Flash 0x00081000 to 0x0008103F and enable the REMAP function. Then, the user could write the application as normal.