ANALOG DEVICES

AN-XXX
APPLICATION NOTE

One Technology Way • P.O. Box 9106 • Norwood, MA 02062-9106, U.S.A. • Tel: 781.329.4700 • Fax: 781.461.3113 • www.analog.com

Cortex-M3 based ADuCxxx Serial Download Protocol

INTRODUCTION

A key feature of the Cortex-M3 based ADuCxxx is the ability of the devices to download code to their on-chip Flash/EE program memory while in-circuit. An in-circuit code download is conducted over the device UART serial port, and is thus commonly referred to as a serial download. The serial download capability allows developers to reprogram the part while it is soldered directly onto the target system, avoiding the need for an external device programmer. The serial download feature also enables system upgrades to be performed in the field; all that is required is serial port access to the Cortex-M3 based ADuCxxx. This means manufacturers can upgrade system firmware in the field without having to swap out the device.

Any Cortex-M3 based ADuCxxx device can be configured for serial down-load mode via a specific pin configuration at power-on or during the application of the external reset signal.

Refer to the device specific user manual for the entry criteria to serial download mode. For example on ADuCRF101, the P0.6 input pin is checked during kernel execution. If this pin is low and RSTSTA.EXTRST == 0x1 at the time that the P0.6 input pin is checked, then the part enters serial download mode.

In this mode, an on-chip resident loader routine is initiated. The on-chip loader configures the device UART and, via a specific serial download protocol, communicates with any host machine to manage the download of data into its Flash/EE memory spaces. The format of the program data to download must be little endian.

Note that serial download mode operates within the standard supply rating of the part. Therefore, there is no requirement for a specific high programming voltage because it is generated on-chip.

As part of the development tools, a Windows® program (CM3WSD.exe) is provided by Analog Devices, Inc. This program allows the user to download code from PC serial ports COM1 to COM31, inclusive, to the Cortex-M3 based ADuCxxx device. Note, however, that any master host machine (PC, micro-controller, or DSP) can download to the Cortex-M3 based ADuCxxx device once the host machine adheres to the serial download protocols detailed in this application note.

This application note details the Cortex-M3 based ADuCxxx device serial download protocol, allowing end users to understand and to successfully implement this protocol (embedded host to embedded Cortex-M3 based ADuCxxx device) in an end-target system.

For the purposes of clarity, the term host refers to the host machine (PC, microcontroller, or DSP) attempting to download data to the Cortex-M3 based ADuCxxx device. The term loader refers to the on-chip serial download firmware on the Cortex-M3 based ADuCxxx device.

TABLE OF CONTENTS

RUNNING THE MICROCONVERTER LOADER

The loader on the ADuCxxx device is run by pulling a specific GPIO pin low through a resistor (typically 1 kΩ pull-down) and resetting the part by toggling the RESET input pin on the part itself. Refer to the device specific user manual for the entry criteria to serial download mode.

For example on ADuCRF101, the P0.6 input pin is checked during kernel execution. If this pin is low and RSTSTA.EXTRST == 0x1 at the time that the P0.6 input pin is checked, then the part enters serial download mode.

THE PHYSICAL INTERFACE

Once triggered, the loader waits for the host to send a back-space (BS = 0x08) character to synchronize. The loader measures the timing of this character and accordingly configures the ADuCxxx UART serial port to transmit/receive at the host's baud rate with 8 data bits and no parity. The baud rate must be between 600 bps and 115,200 bps included.

On receiving the backspace, the loader immediately sends the following 24-byte ID data packet:

15 bytes = product identifier
3 bytes = hardware and firmware version number
4 bytes = reserved for future use
2 bytes = line feed and carriage return

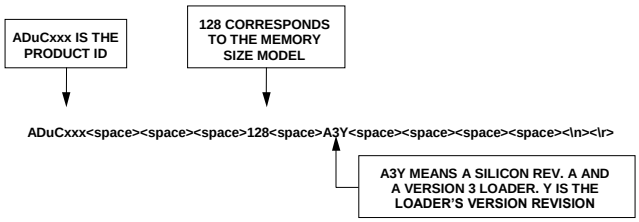

Figure 1.Example ID Data Packet

DEFINING THE DATA TRANSPORT PACKET FORMAT

Once the UART has been configured, a data transfer can begin. The general communications data transport packet format is shown in Table 1.

Packet Start ID Field

The first field is the packet start ID field, which contains two start characters (0x07 and 0x0E). These bytes are constant and are used by the loader to detect a valid data packet start.

Number of Data Bytes Field

The next field is the total number of data bytes, including Data 1 (command function). The minimum number of data bytes is five, which corresponds to the command function and the address. The maximum number of data bytes allowed is 255: a command function, a 4-byte address, and 250 bytes of data.

Command Function Field (Data 1)

The command function field describes the function of the data packet. One of four valid command functions is allowed. The four command functions are described by one of four ASCII characters: E, W, V, or R. The list of data packet command functions is shown in Table 2.

Address Field (Data 2 to Data 5)

The address field contains a 32-bit address (h, u, m, l) with MSB in the h location and LSB in the l location.

Data Byte Field (Data 6 to Data 255)

User code is downloaded by bytes. The data byte field contains a maximum of 250 data bytes.

Checksum Field

The data packet checksum is written into the checksum field. The two's complement checksum is calculated from the summation of the hexadecimal values in the number of bytes field and the hexadecimal values in the Data 1 to Data 255 fields (as many as exist). The checksum is the twos complement value of this summation. Thus, the LSB of the sum of all the bytes from the number of data bytes to the checksum inclusive should be 0. This can also be expressed mathematically as

$$CS = 0x00 - (Number\ of\ Data\ Bytes + \sum_{N-1}^{255} Data\ Byte_N)$$

Expressed differently, the 8-bit sum of all bytes excluding the Start ID must be 0x00.

Acknowledge of Command

The loader routine issues a BEL (0x07) as a negative response or an ACK (0x06) as a positive response to each data packet.

A BEL is transmitted by the loader if it receives an incorrect checksum or an invalid address. The loader does not give a warning if data is downloaded over old (unerased) data. The PC interface must ensure that any location where code is downloaded is erased.

Erase Command

The erase command allows the user to erase Flash/EE from a specific page determined by Data 2 to Data 5. The address is rounded down to the page start. This command also includes the number of pages to erase. If the address is 0x00000000 and the number of pages is 0x00, the loader interprets this as a mass erase command, erasing the entire user code space. The data packet for the erase command is shown in Table 3.

Table 1. Data Transport Packet Format

Start ID						
ID0	**ID1**	**No. of Data Bytes**	**Data 1 CMD**	**Data 2 to Data 5**	**Data x (x = 6 to 255)**	**Checksum**
0x07	0x0E	5 to 255	E, W, V, or R	h, u, m, l	xx	CS

Table 2. Data Packet Command Functions

Command Functions	Command Byte in Data 1 Field	Loader Positive Acknowledge	Loader Negative Acknowledge
Erase Page	E (0x45)	ACK (0x06)	BEL (0x07)
Write	W (0x57)	ACK (0x06)	BEL (0x07)
Verify	V (0x56)	ACK (0x06)	BEL (0x07)
Run (Jump to User Code)	R (0x52)	ACK (0x06)	BEL (0x07)

Table 3. Erase Flash/EE Memory Command

Start ID						
ID0	**ID1**	**No. of Data Bytes**	**Data 1 CMD**	**Data 2 to Data 5**	**Data 6 (Pages)**	**Checksum**
0x07	0x0E	6	E (0x45)	h, u, m, l	x pages (1 to 256)	CS

Table 4. Program Flash/EE Memory Command

Start ID						
ID0	**ID1**	**No. of Data Bytes**	**Data 1 CMD**	**Data 2 to Data 5**	**Data x (x = 1 to 250)**	**Checksum**
0x07	0x0E	5 + x (6 to 255)	W (0x57)	h, u, m, l	Data bytes	CS

Table 5. Verify Command

Start ID						
ID0	**ID1**	**No. of Data Bytes**	**Data 1 CMD**	**Data 2 to Data 5**	**Data 6 to Data 9**	**Checksum**
0x07	0x0E	9	V (0x56)	h, u, m, l	LFSR of page specified by h,u,m,l (little endian sequence)	CS
Start ID						
ID0	**ID1**	**No. of Data Bytes**	**Data 1 CMD**	**Data 2 to Data 5**	**Data 6 to Data 9**	**Checksum**
0x07	0x0E	9	V (0x56)	0x80, 0x00,0x00, 0x00	Data @ 0x1FC (little endian sequence)	CS

Table 6. Remote Reset Command

Packet ID					
ID0	**ID1**	**No. of Data Bytes**	**Data 1 CMD**	**Data 2 to Data 5**	**Checksum**
0x07	0x0E	0x05	R (0x52)	h, u, m, l = 0x1	0xA8

Write Command

The write command includes the number of data bytes (5 + x), the command, the address of the first data byte to program, and the data bytes to program. The bytes are programmed into Flash/EE as they arrive. The loader sends a BEL if the checksum is incorrect or if the address received is out of range. If the host receives a BEL from the loader, the download process should be aborted and the entire download sequence started again.

Verify Command

The verify command uses the SIGN command of the Flash Controller to generate an LFSR signature of a page.

The SIGN command of the Flash Controller excludes the last four bytes of a page. Therefore to validate that the contents of a page are what you expect it to be, you need to specify the LFSR value and what you expect the last four bytes to be.

To use this command, a 2-step sequence must therefore be followed for each page that is to be verified:

1. Send the value 0x80000000 in h, u, m, l. Send the data at 0x1FC of the page in Data Bytes 6 to 9; The loader stores the value for comparision in the next command.
2. Send the start address of the page in h, u, m, l. Send the LFSR of the page in Data Bytes 6 to 9;

The loader now computes the LFSR of the specified page and compares it to a supplied value. If it is correct and the value at address 0x1FC of that page matches the value specified in step 1, ACK (0x06) is returned; otherwise BEL (0x07) is returned.

Remote Reset Command

Once the host has transmitted all data packets to the loader, the host can send a final packet instructing the loader to perform a reset. A single type of remote reset is implemented, a core and peripheral software reset, with h, u, m, l = 0x1.

The host should ensure that the P0.6 pin is no longer asserted before issuing this command. The part will reset, re-enter the kernel as normal. The boot-loader entry check will be performed once more so P0.6 pin must be de-asserted at this time. (The kernel does not modify the RSTSTA so the check for an external reset will still detect that an external reset occurred)

Table 6 shows an example of a Remote Reset.

Remote Run Command

The following is an example of this command for reference captured using a port analyzer.

IRP_MJ_WRITE Length 9 : 07 0E 05 52 00 00 00 01 A8

IRP_MJ_READ Length 1 : 06

Verify Command

The following is an example of this command for reference captured using a port analyzer.

Value at 0x1FC for the next verify command is specified to be 0x11223344

IRP_MJ_WRITE Length 13: 07 0E 09 56 80 00 00 00 *44 33 22 11 77*

IRP_MJ_READ Length 1 : 06

Verify of page at 0x00000200, LFSR specified to be 0x00841B81, last value will be checked against 0x11223344

IRP_MJ_WRITE Length 13: 07 0E 09 56 00 00 02 00 81 1B 84 00 7F

IRP_MJ_READ Length 1 : 06

Erase Command

Erase 1 page at 0x00000200,

IRP_MJ_WRITE Length 10: 07 0E 06 45 00 00 02 00 01 B2

IRP_MJ_READ Length 1 : 06

Mass Erase entire user space

IRP_MJ_WRITE Length 10: 07 0E 06 45 00 00 00 00 00 B5

IRP_MJ_READ Length 1: 06

Write Command

Write 16 data bytes starting at 0x00000200,

IRP_MJ_WRITE Length 25: 07 0E 15 57 00 00 02 00 77 FF 2C B1 00 20 00 F0 5A FC 08 B1 01 20 00 E0 1F

IRP_MJ_READ Length 1 : 06