ANALOG DEVICES

AN-806
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

Flash Programming via I^{2}C—Protocol Type 5

INTRODUCTION

A key feature of the Analog Devices, Inc., microcontroller product family is the ability of the devices to download code to on-chip Flash/EE memory while in circuit.

This application note applies to devices in which the in circuit code download feature is conducted over the device I^{2}C serial port.

A Windows® executable program (**I2CWSD.exe**) allows users to download code to the microcontroller from a USB port via an I^{2}C pod, such as USB-I2C/LIN-CONV-Z. Any master host machine with an I^{2}C master (microcontroller, DSP, or other) can download to the microcontroller after the host machine adheres to the I^{2}C download protocols.

This application note outlines the microcontroller I^{2}C download protocol, which allows end users to both understand the protocol and, if required, to implement this protocol in an end target system (with an embedded host to an embedded microcontroller).

For the clarity purposes, the term host refers to the host machine (microcontroller, DSP, or other machine) attempting to download data to the microcontroller. In addition, the term loader refers specifically to the on-chip serial download firmware resident on the microcontroller.

Note that throughout this application note, multifunction pins, such as the P0.0/nTRST/ADC$_{BUSY}$/PLAI[8]/BM pin, are referred to either by the entire pin name or by a single function of the pin, for example, P0.0, when only that function is relevant.

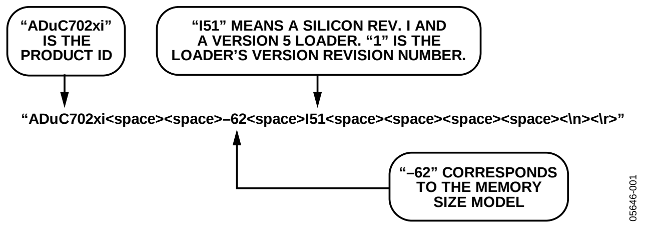
"ADuC702xi" IS THE PRODUCT ID
"I51" MEANS A SILICON REV. I AND A VERSION 5 LOADER. "1" IS THE LOADER'S VERSION REVISION NUMBER.
"ADuC702xi<space><space>–62<space>I51<space><space><space><space><\n><\r>"
"–62" CORRESPONDS TO THE MEMORY SIZE MODEL
05646-001

Figure 1. Example Product Identifier

TABLE OF CONTENTS

Introduction 1
Revision History 2
I²C Download Protocol 3
Running the Microcontroller Loader 3
The Physical Interface 3
Defining the Data Transport Packet Format 3
Data Packet Command Functions 5

REVISION HISTORY

3/15—Rev. C to Rev. D

Changed MicroConverter to Microcontroller .... Throughout
Changes to Title and Introduction Section 1
Moved Revision History Section 2
Added I²C Download Protocol Section, Downloader Entry for the ARM7 Core Devices Section, Downloader Entry for the Cortex-M3 Devices Section, and Node Address Section 3
Changes to The Physical Interface Section 3
Changes to Acknowledge of Command Section, Table 3, Table 4, and Table 5 4
Changes to Table 6, Table 7, Erase Command Section, Write Command Section, and Verify Command Section 5
Added Table 8 and Table 9; Renumbered Sequentially 5
Changed Flash/EE Memory Protection Command Section to Protect Flash/EE Memory Command Section 6
Changes to Protect Flash/EE Memory Command Section and Run Command Section 6

8/12—Rev. B to Rev. C

Changes to Title, Introduction Section, and Figure 1 Caption 1
Changes to Running the MircoConverter Loader Section and The Physical Interface Section 2

5/12—Rev. A to Rev. B

Changes to Introduction Section 1
Changes to Flash/EE Memory Protection Command Section 4

I^{2}C DOWNLOAD PROTOCOL

RUNNING THE MICROCONTROLLER LOADER

Downloader Entry for the ARM7 Core Devices

To allow an unattended download via I^{2}C, the device enters loader mode only if boot mode (BM) (for example P0.0 on the ADuC7023) is low during reset, and the content of Flash/EE memory at Address 0x14 is 0xFFFFFFFF. For BM to determine if loader mode is entered, the user must ensure that the word at Flash Address 0x14 is 0xFFFFFFFF.

Alternatively, BM can be kept low and entry to download mode can be determined by the content of Flash Address 0x14. Typically, the value at Flash Address 0x14 is not 0xFFFFFFFF; therefore, user code must have a built-in mechanism for erasing Page 0 (Flash Address 0x0 to Flash Address 0x200) and for resetting the device. This mechanism allows entry to download mode to reprogram the device.

Ideally, the value at Flash Address 0x14 is programmed last to allow re-entry to download mode in case power fails or another error occurs during reprogramming of the bulk of the program.

Downloader Entry for the Cortex-M3 Devices

If the BM pin (P2.3) is high, the kernel exits to user code.

If the BM pin (P2.3) is low, the kernel waits for a user defined time for the first I^{2}C frame to arrive. If the first frame consists of Node Address 0x04 and one data byte of 0x08 (backspace) exactly, the kernel enters the downloader. Any other first frame causes the kernel to exit to user code. In addition, the user can set a timeout at Address 0x3FFE0. The 16-bit value gives the number of milliseconds before the timeout occurs. When the timeout occurs, the kernel exits to user code. The maximum timeout is 65 sec, which is when the value is erased. As a special case, a timeout value of 0xFFFE sets the timeout to 15 minutes. In addition, if after the device enters download mode no valid download frames arrive for 15 minutes, the downloader exits via a software reset.

Note that if during the timeout period the first frame arriving is Node Address 0x04, the device provides ACK signals as required by the I^{2}C standard, which has to happen even if it is not a 0x04, 0x08 frame. It is, therefore, recommended that Node Address 0x04 be reserved for entry to the downloader and that it is not used in a system application.

THE PHYSICAL INTERFACE

After the loader triggers, it configures the I^{2}C0 port as a slave device; for example, P0.4 (SCL0) and P0.5 (SDA0) on the ADuC7023. Its slave address is 0x04; therefore, the host must start each transmission of data to the loader with Byte 0x04 (I^{2}C write), and each command acknowledge request from the loader with Byte 0x05 (I^{2}C read). The data of the first packet sent by the loader must be backspace (BS = 0x08) to start the protocol.

When in download mode and after receiving the backspace character, the loader sends the following 24-byte ID data packet:

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

DEFINING THE DATA TRANSPORT PACKET FORMAT

After the I^{2}C has been configured, the transfer of data can begin. The general communications data transport packet format is shown in Table 1.

Node Address

Note that each frame must start with the I^{2}C slave node address as required by the I^{2}C specification; however, this is not shown in the tables in this application note.

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 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 5, which corresponds to the command function and the address. The maximum number of data bytes allowed is 255: command function, 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 five valid command functions is allowed. The five command functions are described by one of five ASCII characters: E, W, V, P, 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. High (h), upper (u), medium (m), and lower (l), with MSB in h and LSB in l.

Data Byte Field 6 to Data Byte Field 255

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

Typically, the data must be stripped out of the Intel® HEX extended 16-byte record format and reassembled by the host as part of the given data format before transmission to the loader.

Checksum Field

The data packet checksum is written into this field. The twos complement checksum is calculated by summing the hex values in the number of bytes field and the hex values in the Data 1 to Data 255 fields (if they exist). The checksum is the twos complement value of this summation, the 8-bit sum of the number of data bytes and Data Byte 1 to Data Byte 255, and can be expressed as follows:

$$CS = 0x00 - \left(\mathit{Number\ of\ Data\ Bytes} + \sum_{N-1}^{255} \mathit{Data\ Byte}_N \right)$$

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

Acknowledge of Command

After each command, the host must request an acknowledge from the loader for a negative response, a BEL (0x07), or a positive response, an ACK (0x06). If the loader receives an incorrect data packet format upon verification of the checksum byte, a BEL transmits. The loader does not give a warning if data is downloaded over old (unerased) data or to an invalid address. The PC interface must ensure that any location that code is downloaded to is erased. The acknowledge indicates that no frame error was detected. It does not indicate that the frame had meaningful content or was executed correctly. The only reliable check for an error free download is the verify command.

Table 1. Data Transport Packet Format

Start ID		Number of Data Bytes	Data 1 CMD	Data 2 to Data 5 (Address: h, u, m, l)[1]	Data x (x = 6 to 255)	Checksum
0x07	0x0E	5 to 255	E, W, V, P, or R	h, u, m, l	xx	CS

[1] High = h, upper = u, medium = m, and lower = l.

Table 2. Data Packet Command Functions

Command Function	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)
Protect	P (0x50)	ACK (0x06)	BEL (0x07)
Run	R (0x52)	ACK (0x06)	BEL (0x07)

Table 3. Erase Flash/EE Memory Command

Start ID		Number of Data Bytes	Data 1 CMD	Data 2[1]	Data 3[1]	Data 4[1]	Data 5[1]	Data 6 (Pages)	Checksum
0x07	0x0E	6	E (0x45)	h	u	m	l	x pages (1 to 255)	CS

[1] High = h, upper = u, medium = m, and lower = l.

Table 4. Write Flash/EE Memory Command

Start ID		Number of Data Bytes	Data 1 CMD	Data 2[1]	Data 3[1]	Data 4[1]	Data 5[1]	Data x (x = 1 to 250)	Checksum
0x07	0x0E	5 + x (6 to 255)	W (0x57)	h	u	m	l	Data bytes	CS

[1] High = h, upper = u, medium = m, and lower = l.

Table 5. Verify Flash/EE Memory Command for ARM7 Devices

Start ID		Number of Data Bytes	Data 1 CMD	Data 2[1]	Data 3[1]	Data 4[1]	Data 5[1]	Data x (x = 1 to 250)	Checksum
0x07	0x0E	5 + x (6 to 255)	V (0x56)	h	u	m	l	Complemented data bytes	CS

[1] High = h, upper = u, medium = m, and lower = l.

Table 6. Verify Flash/EE Memory Command Step 1 for Cortex-M3 Devices

Start ID		Number of Data Bytes	Data 1 CMD	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Data 8	Data 9	Checksum
0x07	0x0E	9	V (0x56)	0x80	0x00	0x00	0x00	0x..7FC	0x..7FCD	0x..7FCE	0x..7FCF	CS

Table 7. Verify Flash/EE Memory Command Step 2 for Cortex-M3 Devices

Start ID		Number of Data Bytes	Data 1 CMD	Data 2[1]	Data 3[1]	Data 4[1]	Data 5[1]	Data 6 CRC	Data 7 CRC	Data 8 CRC	Data 9 CRC	Checksum
0x07	0x0E	9	V (0x56)	h	u	m	l	[0 to 7]	[8 to 15]	[16 to 23]	[24 to 31]	CS

[1] High = h, upper = u, medium = m, and lower = l.

Table 8. Protect Flash/EE Memory Command for ARM7 Devices

Start ID		Number of Data Bytes	Data 1 CMD	Data 2[1]	Data 3[1]	Data 4[1]	Data 5[1]	Data 6	Checksum
0x07	0x0E	6	P (0x50)	h	u	m	l	Type	CS

[1] High = h, upper = u, medium = m, and lower = l.

Table 9. Run Command

Start ID		Number of Data Bytes	Data 1 CMD	Data 2	Data 3	Data 4	Data 5	Checksum
0x07	0x0E	5	R (0x52)	0x00	0x00	0x00	0x01	0xA8

DATA PACKET COMMAND FUNCTIONS

Erase Command

The erase command can erase one page or up to all pages of Flash/EE from a specific address determined by Data 2 to Data 5. This command also specifies the number of pages to erase. If the address is 0x00000000 and the number of pages is 0x00, the loader interprets it as a mass erase command, erasing the entire user code space and the Flash/EE protection.

The data packet for the erase command is shown in Table 3. The address in h, u, m, and l is truncated to the page boundary.

Write Command

The write command specifies the number of data bytes (which is Data 1 + Data 2 to Data 5 + Data x), the command, the address of the first data byte to program, and the data bytes to program. For Cortex-M3 devices, the address must start on a double word (8-byte) boundary and end on a double word boundary. As the bytes arrive, they are programmed into the Flash/EE memory. 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, the download process aborts, and the entire download sequence starts again.

Verify Command

The verify command for the ARM7 devices is almost identical to the write command, as shown in Table 5. The command field is V (0x56); however, to improve the chance of detecting errors, the data bytes are modified: the low five bits are shifted to the high five bits, and the high three bits are shifted to the low three bits.

Table 10. Verify Command, Bit Modifications

Original Bits	Transmitted Bits	Restored Bits
7	4	7
6	3	6
5	2	5
4	1	4
3	0	3
2	7	2
1	6	1
0	5	0

The loader restores the correct bit sequence and compares it to the flash contents. If it is correct and the checksum is correct, ACK (0x06) returns; otherwise, BEL (0x07) returns.

For Cortex-M3 devices, the verify command is a two-step sequence for each page to be verified, as follows:

1. Send the 0x80000000 value in the value field and the last four bytes of the page in the data bytes field.
2. Send the page address in the value field and the result of the SIGN command of the page in the data bytes field.

After receiving these two packets, the loader checks whether the value received in Step 1 is correct and then obtains the SIGN value for the page and compares it to the value from Step 2. If both values are correct, an ACK is subsequently returned. An error in either value results in a BEL. For details on the SIGN command, see the appropriate device data sheet or hardware reference manual.

Protect Flash/EE Memory Command

To use this command, follow this three-step sequence for the ARM7 devices:

1. Initiate the command. The type must be 0x00 and h, u, m, and l can be any value.
2. Send the address of the group of pages to protect. Repeat this step for each group of pages. Type must be 0x0F.
3. Send the key in h, u, m, and l; type must be 0x01. FEEADR takes the value of h and u, and FEEDAT takes the value of m and l. If no keys are required, h, u, m, and l must be 0xFFFFFFFF.

For example, to protect Page 0 to Page 7 against writing, set the read protection and use key to 0x12345678.

Send the following commands:

1. Start sequence:
   0x07 0x0E 0x06 0x50 0xXXXXXXXX 0x00 CS
2. Protection:
   0x07 0x0E 0x06 0x50 0x00000000 0x0F CS (Page 0 to Page 3)
   0x07 0x0E 0x06 0x50 0x00000800 0x0F CS (Page 4 to Page 7)
   0x07 0x0E 0x06 0x50 0x0000F800 0x0F CS (read protection)
3. Key and end of sequence:
   0x07 0x0E 0x06 0x50 0x12345678 0x01 CS

Note that the protection command is only available in Revision 0 and later versions of the loader. In Revision 0, FEEADR = ml and FEEDAT = ml. In later versions, FEEADR = hu.

This protocol does not allow Flash/EE memory to be unprotected. To remove the protection, use a mass erase command.

The ADuCM320 does not support the protect command. Use a normal write command to Address 0x1FFE8 to Address 0x1FFF7 and/or Address 0x3FFE8 to Address 0x3FFF7 as described in the ADuCM320 Hardware Reference Manual, instead of the protect command. Note that the values at Address 0x1FFF4 and Address 0x3FFF4, used for user read protection as described in the ADuCM320 Hardware Reference Manual, also disable the page erase and write commands to prevent removing the protection.

Run Command

After the host has transmitted all data packets to the loader, the host can send a final packet instructing the loader to exit and start executing user code.

For ARM7 devices, implement the following two remote runs:

- A software reset, with h, u, m, l = 0x1
- A jump to user code, with h, u, m, l = 0x0

Table 9 shows an example of a remote run or reset. Executing a software reset is recommended because it resets all peripherals. However, when the P0.0 pin is permanently grounded and Address 0x80014 clears, it can be necessary to use a jump to user code. Note that after a jump to user code, the I^2C peripheral memory mapped registers (MMRs) do not contain default values.

Cortex-M3 devices support the software reset; however, these devices do not support the jump to user code.

I^2C refers to a communications protocol originally developed by Philips Semiconductors (now NXP Semiconductors).