

In System Programming

Communication Protocol Specification

Revision 2.1

© 2002 National Semiconductor Corporation. All rights reserved. No warranty is provided and no liability is assumed by National Semiconductor Corporation with respect to the accuracy of this documentation or the merchantability or fitness of the product for a particular application. No license of any kind is conveyed by National Semiconductor Corporation with respect to its intellectual property or that of others. All information in this document is subject to change without notice.

National Semiconductor Corporation products are not authorized for use in life support systems or under conditions where failure of the product would endanger the life or safety of the user, except when prior written approval is obtained from National Semiconductor Corporation.

CompactRISC, MICROWIRE, and TRI-STATE are trademarks of National Semiconductor Corporation.

The National Semiconductor logo is a trademark of National Semiconductor Corporation.

All other trademarks mentioned in this document are property of their respective companies.

Preface

Scope of this document

This document is a guide and reference manual for the In System Programming (ISP) capability provided on National Semiconductors CompactRISC™ based microcontrollers.

The information contained in this document provides information on how to enter ISP Mode, how the physical interfaces are used, and the functionality of ISP. It also describes the ISP Communications Protocol for those users who wish to implement their own ISP programmer, perhaps as part of an automated end-of-line tool during manufacture.

Revision History

Revision	Release Date	Summary of Changes
1.0	6 Aug 1999	Original 'isp_spec.pdf' documentation
2.0	4 July 2002	First major release of updated ISP Specification in PDF format-stuart.cording@nsc.com
2.1	10 July 2002	Modified minor typos in Chapter 2

Preface

Table of Contents

1 General Information	13
1.1 Introduction	13
1.2 The CompactRISC Family	13
1.3 Features	14
1.4 Functional Description	15
1.5 Entering ISP Mode	15
1.6 How In-System-Programming works	17
1.7 Resource Allocation	20
2 Host Interface	23
2.1 UART/USART Physical Interface	23
2.1.1 UART Target/Host Synchronisation	24
2.1.2 Protocol	27
2.1.3 Data Format	28
2.1.4 Command Structure	29
2.2 USB Physical Interface	30
2.2.1 USB Enumeration	31
2.2.2 Protocol	35
2.2.3 Data Format	36
2.2.4 Command Structure	39
2.3 Erroneous events during ISP updates	40

Table of Contents

2.4 Alternative Physical Interfaces	41
3 ISP Firmware Commands	43
3.1 Read Byte - 'r'	45
3.2 Read Word - 'R'	46
3.3 Write Byte - 'w'	47
3.4 Write Word - 'W'	48
3.5 Program - 'P'	49
3.6 Program Flash - 'p'	51
3.7 Load - 'L'	53
3.8 Verify - 'V'	55
3.9 Go - 'G' (CR16C core products only)	55
3.10 Disconnect - 'D'	58
3.11 Checksum calculation	58
4 Legacy ISP Firmware	61
4.1 Watchdog Timer	61
4.2 Watchdog support	62
4.3 Watchdog Information Field	62
4.4 Default Settings for CR16MHS9 and CR16MCS9	63
4.5 Supporting the 'Go' command (CR16C)	66

4.6 Jumping the ISP Mode 67

A ISP RAM Download Files 71

B Standard Clock Speeds. 73

Table of Contents

List of Figures

Figure 1-1. Main flow diagram for ISP Firmware	19
Figure 2-1. Recommended circuit for USART transceiver based upon the DS14C232	24
Figure 2-2. Flow of baud-rate detection algorithm and target/host synchronisation	26
Figure 2-3. ISP firmware serial data format	28
Figure 2-4. ISP data transfer example (0x55 - ASCII "U")	28
Figure 2-5. USB connection interface	31
Figure 2-6. Flow diagram for enumeration of USB and synchronisation with host	33
Figure 2-7. Handling of ISP Commands that are 64 bytes in length	38

List of Figures

List of Tables

Table 1-1. The generic term used in this document to define particular CompactRISC product families	14
Table 1-1. The generic term used in this document to define particular CompactRISC product families	14
Table 1-2. Usual environment pin settings for ISP and IRE Modes	16
Table 2-1. - CompactRISC device recognition characters	27
Table 2-2. Enumeration descriptors for ISP over USB	34
Table 3-1. -ISP Code commands, valid operational area and op-codes	44
Table 4-1. Unavailable EEPROM memory locations in CR16B core devices	61
Table 4-2. Watchdog Information Field	63
Table 4-3. Default Watchdog Information Field Settings	64
Table 4-4. Watchdog Information Field Settings for the CR16MCS9	64
Table 4-5. Watchdog Information Field Settings for the CR16MHS9	65
Table A-1. List of files for downloading into RAM to support flash writes and page erases	71
Table B-1. Expected clock source frequencies for the various CompactRISC™ microcontrollers	73

List of Tables

General Information

1.1 Introduction

In-System-Programming support was added with the release of the CR16B core CompactRISC™ microcontroller devices.

The In-System-Programming communications protocol defines the way in which firmware update information is passed to CR16B and CR16C core devices. The physical communication link can be achieved through any of the microcontrollers on-chip communications modules. The host device is generally a PC running a suitable Windows based 'firmware update' application software package. The contents of this document explain the In-System-Programming protocol so that the user can develop their own host software, or alternatively develop an alternative physical interface. This document describes the serial and USB interfaces that have been implemented on various products.

1.2 The CompactRISC Family

The ISP facility is currently supported by microcontrollers that use the CR16B and CR16C processor cores. Due to architectural differences there are slight variations in the way certain aspects of the ISP capabilities are implemented. Table 1 below should be used to determine which processor core the microcontroller of interest uses and the generic term that will be used in this document to refer to

General Information

Features

that product. This will clarify the features and capabilities of ISP with that product.

Table 1-1. The generic term used in this document to define particular CompactRISC product families

CompactRISC™ Products	Generic Term
CR16MAR/MAS/MBR/MCS/ MES/MFS/MHS/MNS/MPS/ HCS/MCT/HCT	CR16B Core
CP3BT, LMX5100, LMX9814	CR16C Core

1.3 Features

The In-System-Programming (henceforth ISP) firmware provides the following features:

- Automatic physical interface detection
- Automatic baud-rate detection for serial interfaces
- Programming of the Flash program memory
- Programming of the Flash data memory
- Read/Write access to RAM
- Read/Write access to all core bus accessible registers and memory locations

1.4 Functional Description

The user must first set up the target device to start in ISP Mode. Once the mode has been successfully entered, the ISP firmware receives and executes commands from a host device (PC, ATE or other device) using the chosen communication interface. Once all programming or memory accesses are complete, the host device can issue a 'Disconnect' message, thereby terminating all further ISP communication. The user can then set up the target device to start in IRE Mode in order to execute any firmware stored in the target device, or initiate a new ISP connection to perform further programming or memory accesses.

1.5 Entering ISP Mode

The CompactRISC™ range of microcontrollers have two main operating environments; ISP Mode used for In-System-Programming, and IRE Mode (Internal ROM Enable Mode) which is the mode used for executing firmware stored in the flash memory of the device.

There are two situations where the user may wish to use ISP; for in-the-field updates of firmware and laboratory programming, or end-of-line programming in a production environment. There are two

General Information

Entering ISP Mode

ways in which the ISP firmware in the CompactRISC device can be initiated.

1. In-the-field and laboratory programming - This is achieved by applying a reset or powering up the device when the ENV0 and ENV1 pins are set as per the relevant device specification for ISP Mode. This method is most suitable for in-the-field updates and laboratory based programming.
2. End-of-line programming - This is achieved by applying a reset or powering up the device when, according to the ENV pins, the device is intended to be started in IRE Mode but the ENPTY bits in the Flash Memory Protection word indicate that the flash memory does not contain user code. Non-programmed production devices are delivered with the EMPTY bits set to indicate an empty flash memory, so this method of entering ISP mode is usually invoked during end-of-line programming.

Table 1-2. Usual environment pin settings for ISP and IRE Modes

ENV0	ENV1	Mode
0	1	ISP Mode
1	1	IRE Mode

More information on the ENV pins settings, and the EMPTY bits of the Flash Protection Word can be found in the relevant device specification under the heading 'Operation Environment' or 'System Configuration'. The location of the ISP firmware memory is also specified in the relevant data sheet.

The usual environment pin settings are shown in Table 2 opposite.

A third method for entering ISP Mode from IRE Mode exists for microcontrollers using the CR16C Core without requiring the user to change the state of the ENV pins and reset the part or cycle the power. Instead a firmware jump is used to instigate entry into the ISP Mode. This method is described further in the section titled 'Supporting the 'Go' command (CR16C)'

1.6 How In-System-Programming works

Assuming one or both conditions for executing the ISP firmware are present, the device will start execution at the memory location where the ISP firmware is stored.

The resident ISP firmware only supports read/write access to RAM, Flash EEPROM (where available), core accessible registers as well as read access to program memory. This is done to provide a high

General Information

How In-System-Programming works

level of security, because in this case no routines reside on-chip that could accidentally overwrite the application software.

In order to be able to write to the flash memory several routines have to execute out of RAM. Therefore the host device (PC, ATE etc.) should use the ISP Communications Protocol reserved for programming RAM memory (i.e. the Load command) to download the appropriate routines to the devices RAM memory prior to attempting to program the flash program memory. The code that should be downloaded is device specific and a table of the appropriate files is given in "Appendix A - ISP RAM Download Files". The files are stored using the Intel Hex File format, as are the binary files National's compiler tools create.

The program flow is shown in Figure 1-1 below. The firmware initialises the BIU (bus interface unit) registers and the registers that change the timings used for flash programming. The flash timing registers are programmed assuming a predetermined clock speed as listed in "Appendix B - Standard Clock speeds expected by ISP Code". These values can be updated by writing to the appropriate registers in memory.

Next the program searches for an active communications interface. If the ISP version supports more than one interface then it will poll each in turn to see which one responds and use that interface for the remainder of the existence of the connection. Depending on the

How In-System-Programming works

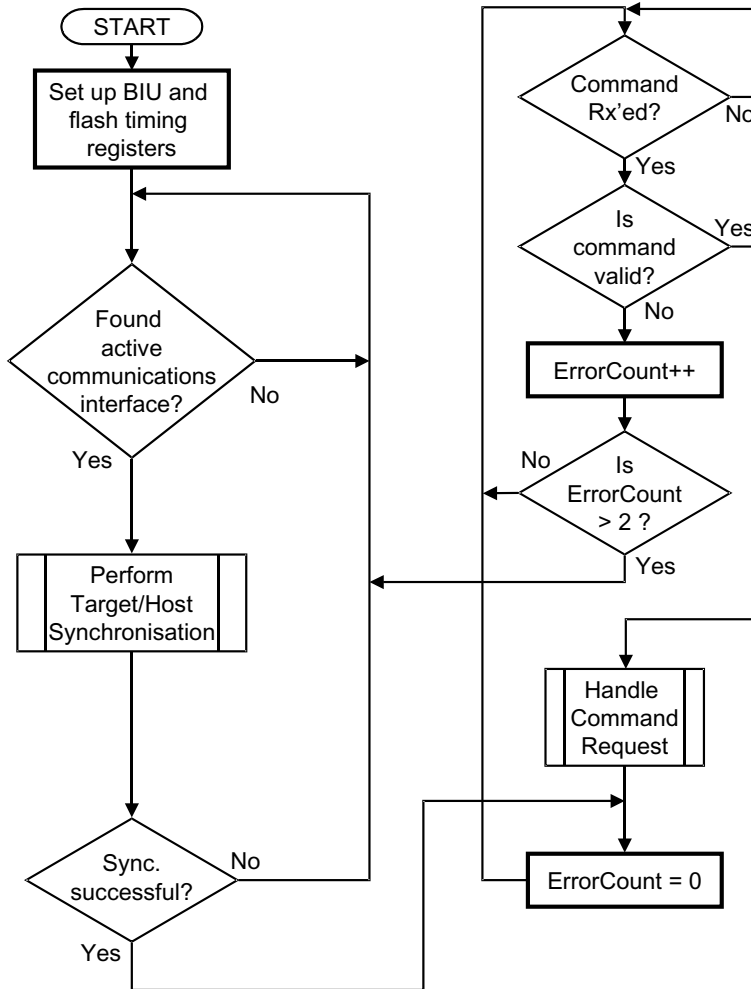


Figure 1-1. Main flow diagram for ISP Firmware

General Information

Resource Allocation

physical interface, a synchronisation procedure is executed which allows the protocol to determine which CompactRISC™ product is being connected to, and, in the case of a serial link, the baud-rate of the connection.

Upon successful completion of the synchronisation procedure the program waits for the reception of valid ISP commands that it responds to as defined in this specification. In order to ensure that the connection is maintained, and that erroneous data does not cause the link to latch-up, a simple error checking routine has been implemented. If the command that is sent to the target is not recognised, or if the checksum is incorrect, and error counter is incremented. If this error counter detects three consecutive erroneous commands the current communications interface is closed and the target device attempts to reconnect with the host.

Once the connection has been reestablished it is important to reconfigure the flash timing registers and download the RAM based programming code required for erasing and programming the flash memory.

1.7 Resource Allocation

The ISP firmware requires the following resources:

For CR16B core based devices

- ISP Memory
- RAM
- MFT1
- MFT2
- USART1

For CR16C core based devices with USB and UART support

- Boot Memory (7kBytes)
- RAM
- Multi Function Timer
- UART
- Watchdog Timer
- General Purpose I/O pins I3, I4 and I5

General Information

Resource Allocation

All communications between the host and the CompactRISC target device shall conform to the details as laid out in the following section. The currently defined physical interfaces are either the serial interface (USART or UART) or the USB interface where it is available. The definition for other interfaces may be made in future revisions of this document, but if the developer wishes create their own interface definition based on the ISP protocol, or modify current ones to suit a different host/target scheme, they are free to do so.

2.1 UART/USART Physical Interface

The serial interface definitions for the UART and USART interfaces are the same. This type of interface will be referred to as the UART interface in the following functional description since it is only used in the asynchronous mode. The available serial interface module depends on the target device. The ISP firmware is designed to communicate through a RS-232 three-wire asynchronous serial link with the host device (TxD/RxD/GND). No hardware or software handshaking is implemented. When interfacing a RS-232 port (on a PC for example) the user should also ensure that appropriate level shifting is connected to the CompactRISC™ serial interface device pins. An example is given in Figure 2-1.

Host Interface

UART/USART Physical Interface

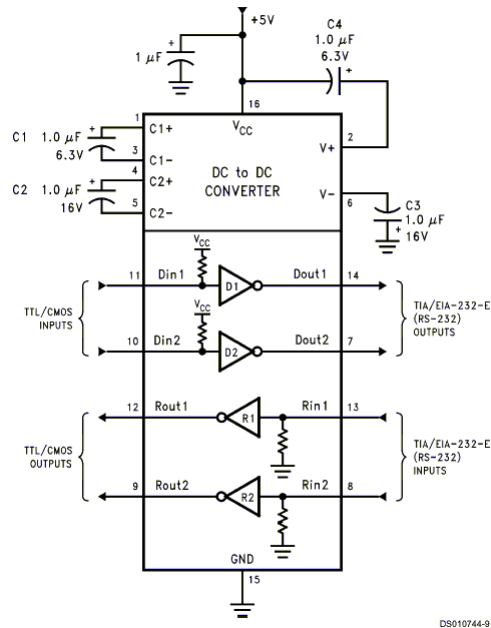


Figure 2-1. Recommended circuit for USART transceiver based upon the DS14C232

2.1.1 UART Target/Host Synchronisation

The ISP firmware has to establish a communication link with the external host. The target device shall attempt to synchronize with the PC by determining the relative baud rate at which it is operating. It will be relative because the target device system clock may be operating at any frequency in the continuum of allowable frequencies and will, therefore, be completely unknown to the internal ISP code. Consequently, the ISP firmware must determine the pres-

UART/USART Physical Interface

caler and divisor register values necessary for synchronizing its UART based on a synchronization character sent by the host. These values may only be ascertained by examining the bit-width (in system-clock cycles) of the incoming serial data.

The host shall attempt to establish communication with the target by initially transmitting the ASCII character "U" (0x55), allowing the target UART to lock on the host's baud rate,

As soon as the baud rate calculating is done, the device initializes its UART and transmits a certain character dependant on the chip type. The host shall then reply with the ASCII character "\$" (0x24). After this synchronisation the host is able to issue any defined command sequence.

In case of a synchronisation failure the target will re-initiate the synchronisation process until a correct resynchronisation is achieved with any active interface. Table 2-1 below lists the character used to identify the target device, while Figure 2-2 below shows the flow of the baud-rate detection algorithm as implemented by the ISP firmware on the current products.

Host Interface

UART/USART Physical Interface

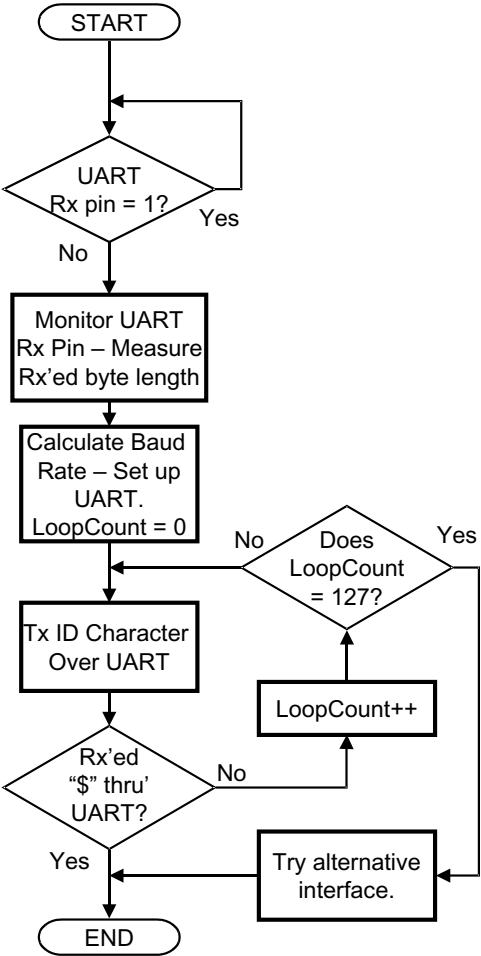


Figure 2-2. Flow of baud-rate detection algorithm and target/host synchronisation

Table 2-1. - CompactRISC device recognition characters

Chip ID	ASCII Character
CR16MHS/MNS/MPS/ MUS	"0"
CR16MCT/HCT/MAR/ MAS/MCS/MBR/MES/ MFS/HCS	"1"
LMX5100/LMX9814/ CP3BT	"2"

2.1.2 Protocol

Once a communication link has been established between the target and the host, the target device is ready to accept further commands from the host. These commands are listed and defined in this document under the 'Command Structure' section. All operation during ISP Mode proceeds on the basis of these commands.

The communication does not support any data flow control (hand-shaking). This means that both the host and the target device must be able to support reception of many bytes of data with the full speed of the selected baud rate. The maximum baud rate supported is 115200 baud.

Host Interface

UART/USART Physical Interface

2.1.3 Data Format

The communication with the ISP firmware is in a standard non-return-to-zero- (NRZ) mark/space data format. The following diagram shows examples for possible transmissions. The ISP firmware data format consists of one start bit, eight data bits and one stop bit (no parity check) as shown below in Figure 2-3 and Figure 2-4 below.

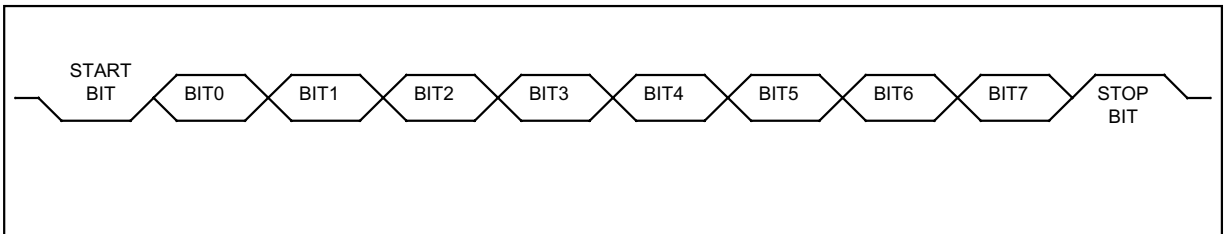


Figure 2-3. ISP firmware serial data format

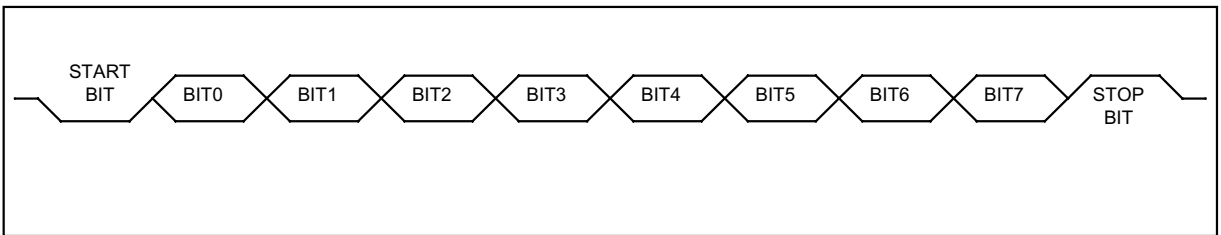


Figure 2-4. ISP data transfer example (0x55 - ASCII "U")

2.1.4 Command Structure

Any ISP firmware command follows the same structure. The host starts transmission of a command, any data bytes and a one-byte checksum. The ISP firmware then checks this command sequence and will send back a positive acknowledgment (ACK) if there was no error during the command transmission and during the execution of the command. A negative acknowledgment (NACK) indicates that the transmission of the command was unnecessarily, or that the execution of the command failed.

A NAK character (0x15) will immediately terminate the command flow and the ISP firmware will expect a new command to be sent. After three consecutive errors the ISP firmware will expect a new command to be sent. After three consecutive errors the ISP firmware will switch back to the synchronisation mode and attempt to establish a fresh link with the host using any available active interface.

Any expected result of a command will be transmitted after the ACK character (0x06) that indicates that the command was received correctly and that execution of the command was successful.

Host Interface

USB Physical Interface

2.2 USB Physical Interface

A full speed USB 1.1 compliant interface is available on some CompactRISC™ devices. This can also be used as the physical interface to perform ISP. The advantages of using USB rather than other interfaces is that the target/host synchronisation is inherent in the USB enumeration process, and that USB is also hot pluggable allowing for simple end-of-line programming. However, this does limit the ISP update host device to being a USB enabled Windows '98 PC or better.

The USB interface described here is specific to the National Semiconductor ISP implementation and does not conform to the 'Device Firmware Upgrade (DFU)' defined by the USB Implementers Forum, Inc.

To include USB functionality a few additional components need to be added to complete the CompactRISC™ devices USB interface. If USB is already an interface in the developers design this should have already been implemented. The circuit diagram for the interface is shown in Figure 2-5.

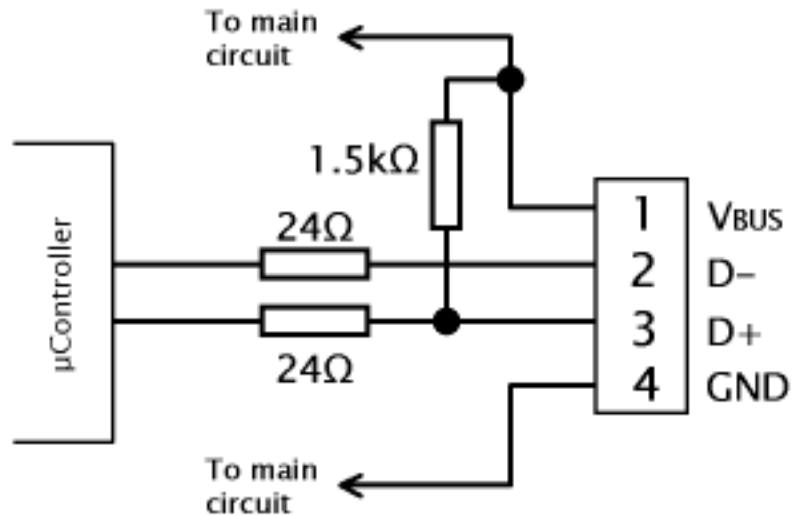


Figure 2-5. USB connection interface

The USB implementation uses the standard Endpoint 0 for control data. In addition it uses a Bulk Receive and Bulk Transmit endpoint. The Bulk endpoints are used for the handling of the ISP protocol and data. This is discussed further below.

2.2.1 USB Enumeration

The enumeration of the USB interface initiates the physical connection between the host and the target devices. However the Vendor

Host Interface

USB Physical Interface

ID (VID) and Product ID (PID) remain the same for all USB enabled devices and thus the same synchronisation process used by the serial interface is used to ensure that the host can determine the type of target device it is connected to. The details of the enumeration descriptor are given in Table 2-2.

Thus, once the enumeration process is complete and the USB device has been configured, the host device should transmit the ASCII character "U" (0x55) in a single byte packet to the Bulk receive endpoint. The Bulk transmit endpoint of the target will then respond with the ASCII character as described in Table 2-1 in a single byte packet to allow the host device to confirm the Compact RISC™ device being interfaced with. The host will then respond with the ASCII character "\$" (0x24) in a single byte packet to confirm that protocol of ISP has been initiated. The procedure implemented allows other physical interfaces to be tested. This ensures that if the USB interface is connected to the host, but the user wishes to use an alternative interface (for example the UART), the alternative interface is not ignored. The enumeration of the node does not necessarily indicate that the user wishes to use that interface, as enumeration will occur if the appropriate driver is installed on the host PC. After this synchronisation the host is able to issue any defined command sequence.

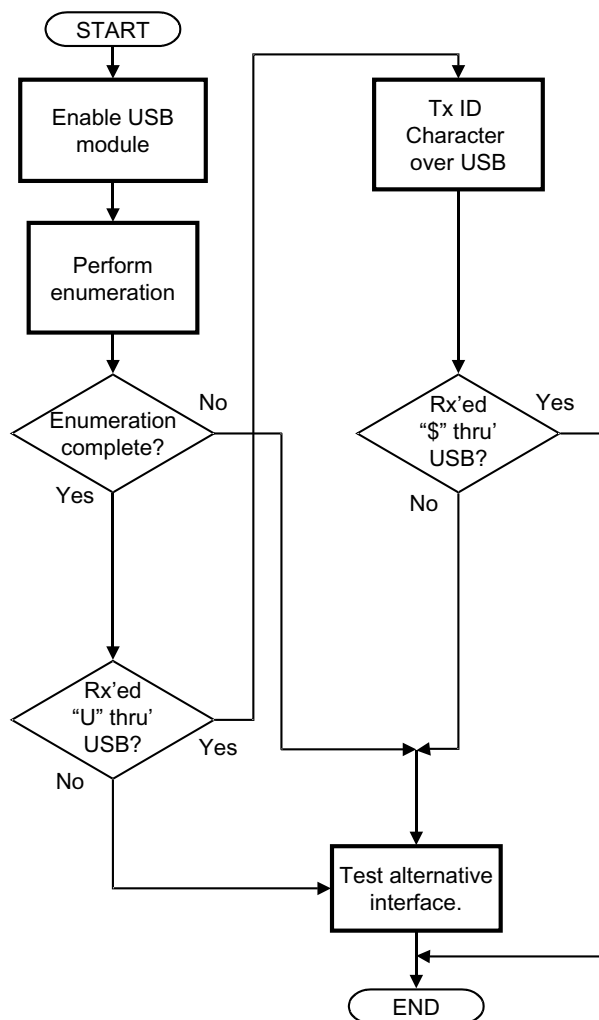


Figure 2-6. Flow diagram for enumeration of USB and synchronisation with host

Host Interface

USB Physical Interface

In case of a synchronisation failure the target will reinitiate the synchronisation process until a correct resynchronisation is achieved with any active interface.

The flow diagram for enumeration and synchronisation with the host is shown in Figure 2-6.

Table 2-2. Enumeration descriptors for ISP over USB

Device Descriptor	
VID	0x0400
PID	0x0806
Device Class	0xE0
Device Sub Class	0x01
Device Protocol	0x01
Endpoint 0 Max Packet Size	0x08
Number of configurations	1
Configuration Descriptor	
Number of interfaces	0x01
Attributes	Self Powered / Remote Walk-up
Max Power	100mA

Table 2-2. Enumeration descriptors for ISP over USB

Interface Descriptor	
Interface Number	0x01
Alternate Settings	0x00
Number of Endpoints	0x02
Interface Class	0x00
Interface Sub Class	0x00
Interface Protocol	0x00
Endpoint Descriptor	
Endpoint Address	0x81
Transfer Type	Bulk
Max Packet Size	0x0040
Interval	0x00
Endpoint Descriptor	
Endpoint Address	0x01
Transfer Type	Bulk
Max Packet Size	0x0040
Interval	0x00

2.2.2 Protocol

The USB protocol for transfer of data over the USB bus is well defined and this implementation conforms to the USB1.1 specification. The following information outlines some further details of using the USB interface to support ISP in this implementation.

Once a communication link has been established between the target and the host, the target device is ready to accept further commands

Host Interface

USB Physical Interface

form the host. These commands are listed and defined in this document under the 'Command Structure' section. All operation during ISP Mode proceeds on the basis of these commands. The two Bulk endpoints are used for reception and transmission of the ISP data. Both endpoints are 64 bytes deep. However, not all ISP commands are exactly 64 bytes in length; in fact most commands will either be shorter than 64 bytes in length, or will need to span several 64 byte packets. When transmitting or receiving data, any packet that is not 64 bytes long is assumed to be the last or only data of a single ISP command. If an ISP command should span a precise multiple of 64 bytes, then a zero length packet should be issued in the next available frame to indicate the end of the ISP command.

2.2.3 Data Format

The communication with the ISP firmware should follow the format as defined in the section titled "ISP Firmware Commands". All ISP commands should be sent as specified, and should not have extra bytes added to fill the remaining bytes of the 64 byte endpoint.

The first six packets shown in Figure 2-7 describe the normal case where an ISP command or data is longer than 64 bytes, but is not modulo 64 in total length. The first 64 bytes are sent in the first

packet (in this case a Data0 packet), whilst the remaining bytes are sent in the next (Data1) data packet.

If a commands length is a precise multiple of 64 bytes, a zero length packet should be sent to indicate the command is complete. This is shown in the last six packets of Figure 2-7. The ISP command or data is exactly 64 bytes long in the Data0 packet. Thus the next packet (Data1) that is sent contains no data.

Although these example only show OUT packets, the same rules apply to IN packets.

Host Interface

USB Physical Interface

SYNC	OUT	ADDR	EP	CRC5	EOP
00000001	0xE1	0xnn	0xnn	0xnn	2

SYNC	DATA0	ADDR	DATA	CRC16	EOP
00000001	0xC3	0x03	12, 05, ... [64bytes] ... E3, 55	0xnn	2

SYNC	ACK	EOP
00000001	0xD2	2

SYNC	OUT	ADDR	EP	CRC5	EOP
00000001	0xE1	0xnn	0xnn	0xnn	2

SYNC	DATA1	ADDR	DATA	CRC16	EOP
00000001	0x4B	0x03	12, 05, .. [10bytes] ... E3, 55	0xnn	2

SYNC	ACK	EOP
00000001	0xD2	2

SYNC	OUT	ADDR	EP	CRC5	EOP
00000001	0xE1	0xnn	0xnn	0xnn	2

SYNC	DATA0	DATA	CRC16	EOP
00000001	0xC3	12, 05, ... [64bytes] ... E3, 55	0xnn	2

SYNC	ACK	EOP
00000001	0xD2	2

SYNC	OUT	ADDR	EP	CRC5	EOP
00000001	0xE1	0xnn	0xnn	0xnn	2

SYNC	DATA1	DATA	CRC16	EOP
00000001	0x4B	[No Data]	0xnn	2

SYNC	ACK	EOP
00000001	0xD2	2

Figure 2-7. Handling of ISP Commands that are 64 bytes in length

2.2.4 Command Structure

Any ISP firmware command follows the same structure. The host starts transmission of a command, any data bytes and a one-byte checksum. The ISP firmware then checks this command sequence and will send back a single byte packet containing a positive acknowledgment (ACK) if there was no error during the command transmission and during the execution of the command. A single byte packet containing a negative acknowledgment (NACK) indicates that the transmission of the command was unsuccessful., or that the execution of the command failed.

The single byte packet containing the NAK character (0x15) will only be sent after the host has completed transmission of an entire ISP command and the ISP firmware has had time to examine the buffered command to check the command and the checksum. After three consecutive errors the ISP firmware will switch back to the synchronisation mode and attempt to establish a fresh link with the host using any available active interface.

Any expected result of a command will be transmitted in the same packet as the ACK character (0x06) that indicates that the command was received correctly and that execution of the command was successful.

Host Interface

Erroneous events during ISP updates

2.3 Erroneous events during ISP updates

Certain events can occur during the use of ISP capability that may affect the ISP processes. The most important function that ISP provides is the erasure and updating of the flash memory. If a hardware or watchdog instigated reset occurs during a flash has completed its process.

If the physical connection between the host and the target device is broken one of two events will occur

- If the connection is broken during the transfer of an ISP command, then the command will be ignored, as the checksum for the command will likely fail.
- if the connection is broken after a valid command has been received then that command will be fully executed.

Once the connection is reestablished the transmission of the "U" character is likely to instigate the automatic interface detection algorithm. Alternatively the user can cycle the power to the device or provide a hardware reset. If power is lost or a drop in power occurs (brown-out) during programmed area cannot be guaranteed to have been correctly programmed or erased. It is recommended that the power to the device is guaranteed during the flash erase and programming process. Any ISP command being sent to the device

will either be corrupted or lost. The power should be restored or the device should be reset in order to restart the ISP firmware.

2.4 Alternative Physical Interfaces

Theoretically any available communications interface on a CompactRISC™ microcontroller can be used as the physical interface to perform ISP. The developer is free to use the ISP Command Protocol defined in this document to provide ISP using an alternative physical interface. National Semiconductor may define the standard use of alternative physical interfaces for ISP using CAN, ACCESS bus and others as the need arises in the future.

Host Interface

Alternative Physical Interfaces

ISP Firmware Commands

The ISP firmware supports ten commands, which are listed in Table 3-1 below along with their ASCII op-codes. The function of each of these commands is described in detail in the following section. The 'Valid Memory Area' is only an indication of the memory type where the command should be used and as such should be checked by the ISP firmware and will not cause a NACK to be returned if used incorrectly. Commands that are used on memory areas that they are not designed for will at best not perform the requested function, and in the worst cases cause memory corruption.

ISP Firmware Commands

Table 3-1. -ISP Code commands, valid operational area and op-codes

Command	Valid Memory Area						ASCII Op-code
	All Addressable Memory	Flash Program Memory	EEPROM Memory	ISP Memory	Data Memory	Static RAM	
Read Byte	●						'r' (0x72)
Read Word	●						'R' (0x52)
Write Byte			●			●	'w' (0x77)
Write Word			●			●	'W' (0x57)
Program				● ¹	● ¹		'P' (0x50)
Program Flash		● ¹					'p' (0x70)
Load			●			●	'L' (0x4C)
Verify	●						'V' (0x56)
Go	● ²						'G' (0x47)
Disconnect							'D' (0x44)

1. -Usage varies between CR16B and CR16C core products - see description for details
2. -Only available on CR16C core products

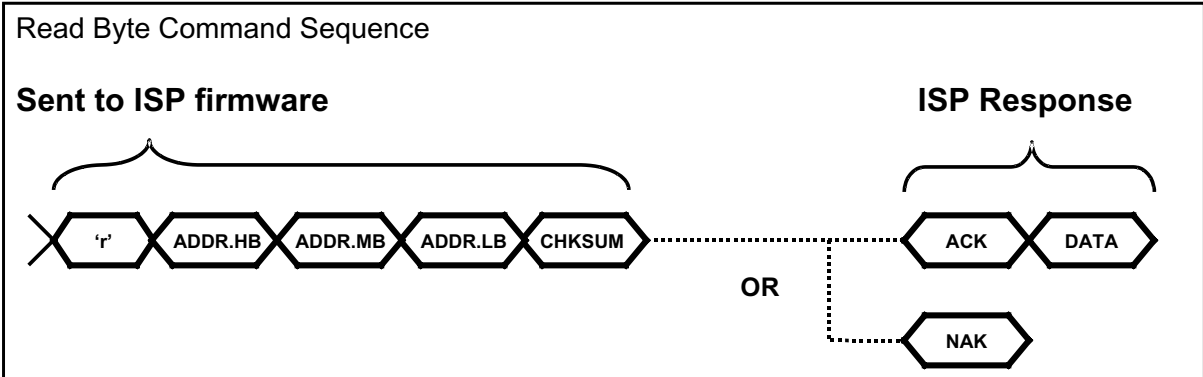
3.1 Read Byte - 'r'

The Read Byte command will return the data byte contents of the specified address. The command can address the complete address range of the microcontroller.

Description: Read single word from memory
Operand: Opcode, three byte memory address in High Byte memory address in High Byte (HB), Middle Byte (MB), Low Byte (LB) order, and checksum.

Data Returned: Contents of addressed word in Low Byte (LB) High Byte (HB) order.

OpCode: 0x72('r')



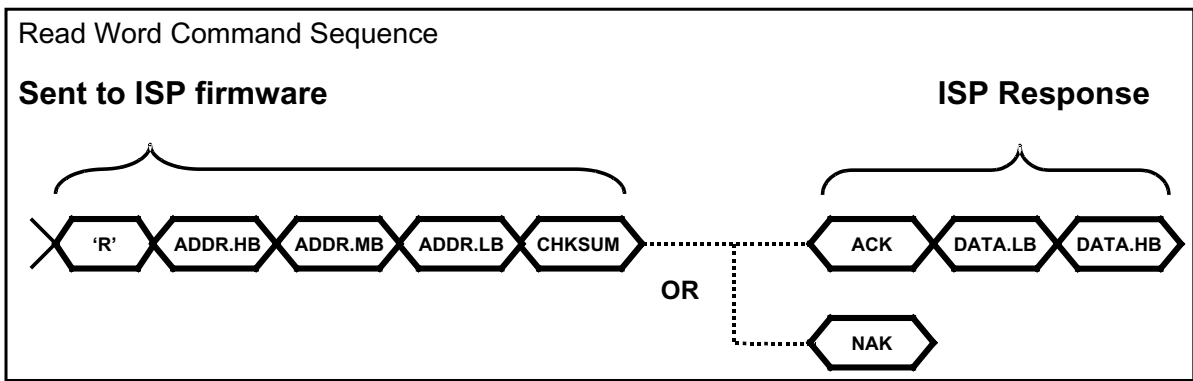
ISP Firmware Commands

Read Word - 'R'

3.2 Read Word - 'R'

The Read Word command will return the data word contents of the specified address in Low Byte/High Byte format. The command can address the complete address range of the microcontroller.

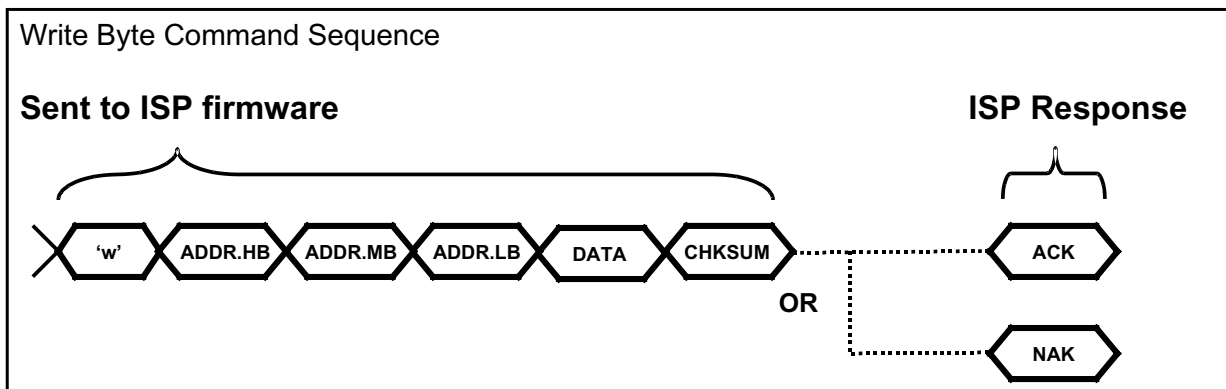
- Description:** Read single word from memory
- Operand:** Opcode, three byte memory address in High byte (HB), Middle Byte (MB), Low Byte (LB) order, and checksum.
- Data Returned:** Contents of addressed word in Low Byte (LB) High Byte (HB order).
- Opcode:** 0x52 ('R')



3.3 Write Byte - 'w'

The Write Byte command will write the data byte sent to the specified address. The command can write to the static RAM and EEPROM memory locations only.

Description:	Write single byte to memory
Operand:	Opcode, three byte memory address in High Byte (HB), Middle Byte(MB),Low Byte (LB) order, data and checksum.
Data Returned:	None
Opcode:	0x77 ('w')



ISP Firmware Commands

Write Word - 'W'

3.4 Write Word - 'W'

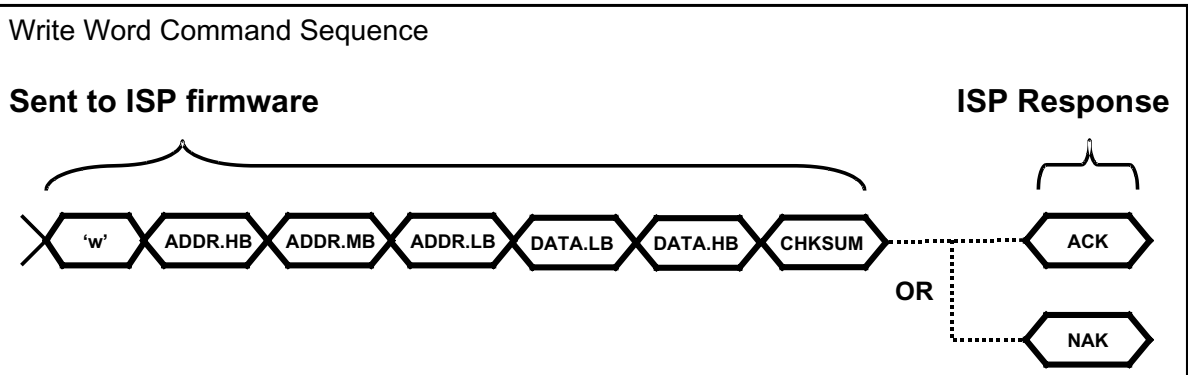
The Write Word command will write the data word sent, in Low Byte/High Byte order, to the specified address. The command can write to the static RAM and EEPROM memory locations only.

Description: Write single word to memory

Opcode, three byte memory address in High Byte (HB), Middle Byte (MB), Low Byte (LB) order, data in Low Byte (LB) High Byte (HB) order and checksum.

Data Returned: None

Opcode: 0x57 ('W')



3.5 Program - 'P'

The Program command will write a number of bytes of data to either the ISP Memory (CR16B core devices) or Data Memory (CR16C core devices) equivalent to the size accepted by the target device. On current CR16B core devices the size is 32 bytes. For current CR16C core devices the size is 512 bytes. The address sent with this command should point to the start address where the data is to be located. The data should be ordered from lowest byte to highest byte, i.e. Address+0, Address+1 to Address+(Size- 1). The address issued with the command must be page aligned. Writing to a non-aligned Flash Memory location could result in data corruption. The programming routine handles the erasure of the page prior to programming, and uses the appropriate flash programming timings. It is recommended that, upon completion of programming, the user should then issue the Verify command, or a series of Read Byte/Word commands to check that the data was written correctly.

For the ISP Memory of CR16B core devices the memory area is grouped into pages of eight bytes. Therefore the address issued with the command must be page aligned. Writing to a non-aligned ISP Memory location could result in data corruption. If the 32-byte data array exceeds the last address of the ISP Memory, the ISP code will stop programming at the end of the last ISP Memory page. Thus this command can also be used to program the security bytes (Flash

ISP Firmware Commands

Program - 'P'

Protection Words) in the last eight bytes of ISP Memory in CR16B core devices.

For the Data Memory of CR16C core devices, the memory area is split into 512 byte pages. There are no security bytes contained in this memory area. The memory can be treated the same as the main flash memory but with a smaller page size which this command supports.

Note that, prior to using this command the appropriate RAM based support file must be downloaded as specified in 'Appendix A - ISP RAM Download Files'.

Description: Program data bytes to ISP Memory or Data Memory.

Operand: Opcode, three byte memory address in High Byte (HB), Middle Byte (MB), Low Byte (LB) order, Size number bytes of data in Address+0, Address+1(Size- 1) order and checksum.

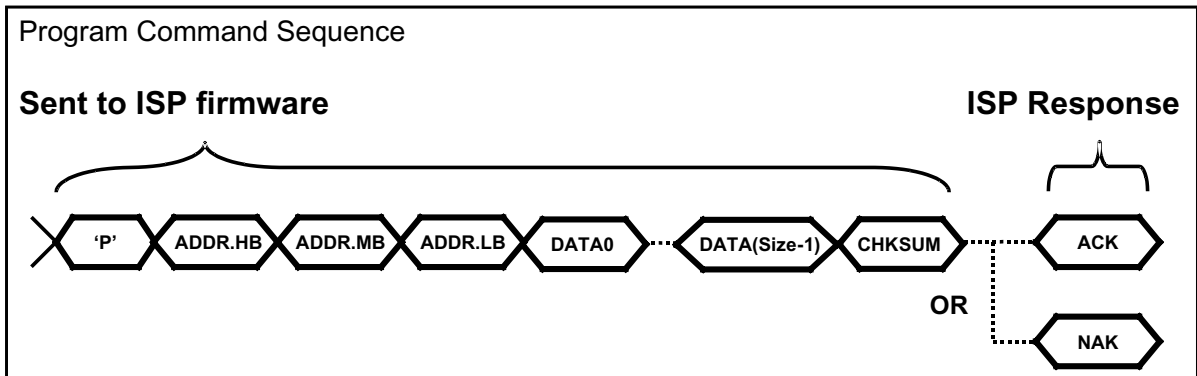
Data Returned: None

Opcode: 0x50 ('P')

Note: Note the length of the data part of the command is dependant on the memory area and microcontroller being programmed as

Program Flash - 'p'

explained above. Also see the data sheet for further detailed information.



3.6 Program Flash - 'p'

The Program Flash command will write a number of bytes of data to Flash Program Memory equivalent to the page size of the target device. On current CR16B core devices the page size is 128 bytes. For current CR16 core devices one page is 1024 bytes in size. The address sent with this command should point to the start address where the data is to be located. The data should be ordered from lowest byte to highest byte, i.e. Address+0, Address+1 to

ISP Firmware Commands

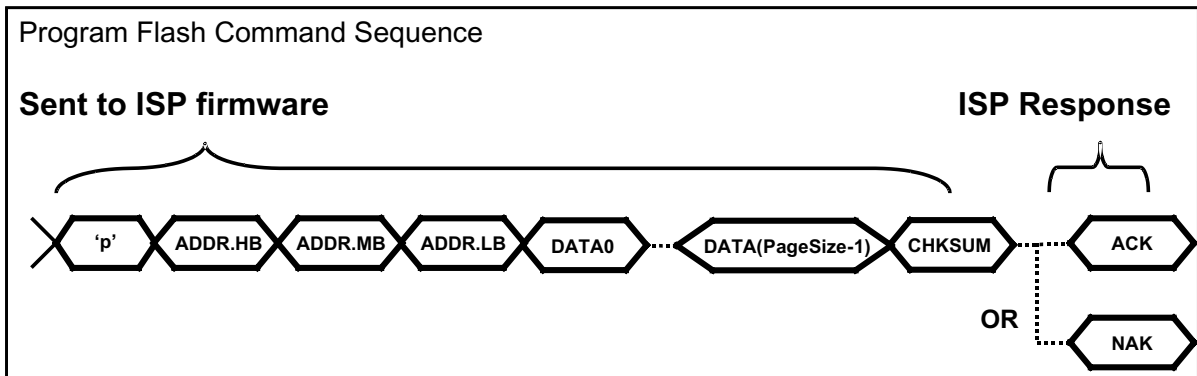
Program Flash - 'p'

Address+(PageSize - 1). The address issued with the command must be page aligned. Writing to a non-aligned Flash Program Memory location could result in data corruption. The programming routine handles the erasure of the page prior to programming, and uses the appropriate flash programming timings. It is recommended that, upon completion of programming, the user should then issue the Verify command, or a series of Read Byte/Word commands to check that the data was written correctly.

Note that, prior to using this command the appropriate RAM based support file must be downloaded as specified in 'Appendix A - ISP RAM Download Files',**Note that, prior to using this command the appropriate RAM based support file must be downloaded as specified in 'Appendix A - ISP RAM Download Files'.**

Description:	Program data bytes to Flash Program Memory
Operand:	Opcode, three byte memory address in High Byte (HB), Middle Byte (MB), Low Byte (LB) order, PageSize number of bytes of data in Address+0, Address+1 ... Address+(PageSize - 1) order and checksum
Data Returned:	None
Opcode:	0x70 ('p')
Note:	Note the length of the data of the command

is dependant on the page size of the microcontroller being used. Check the Data sheet for further detailed information.



3.7 Load - 'L'

The Load command will write 32 bytes of data to RAM or EEPROM memory. The address sent with this command should point to the start address where the data is to be located. The data should be ordered from lowest byte to highest byte, i.e. Address+0, Address+1 to Address+31. It is recommended that, upon completion of programming, the user should then issue the Verify com-

ISP Firmware Commands

Load - 'L'

mand, or a series of Read Byte/Word commands to check that the data was written correctly.

Description: Load 32 bytes into RAM or EEPROM memory

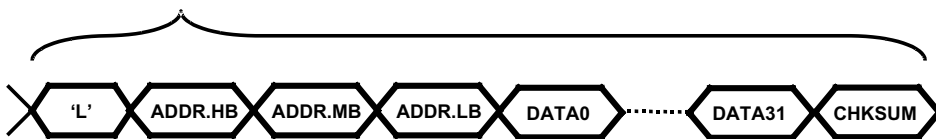
Operand: Opcode, three byte memory address in High Byte (HB), Middle Byte (MB), Low Byte (LB) order, 32 bytes of data in Address+0, Address+1 ... Address+31 order and checksum.

Data Returned: None

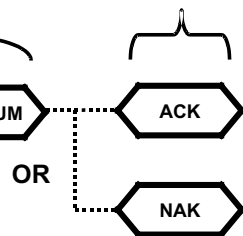
Opcode: 0x4C ('L')

Load Command Sequence

Sent to ISP code



ISP Response



3.8 Verify - 'V'

The Verify command will read 32 bytes of data. the command can address the complete address range of the microcontroller. The address sent with this command should point to the address where the data to be read starts. If the command is recognised and the checksum is valid, the command is acknowledged and the data requested is returned ordered from lowest byte to highest byte, i.e. Address+0, Address+1, Address+1 to Address+31.

Description:	Read out 32 bytes from any memory location
Operand:	Opcode, three byte memory address in High Byte (HB), Middle Byte (MB), Low Byte (LB) order, and checksum.
Data Returned:	32 bytes of data, starting with the data located at Address+31
Opcode:	0x56 ('V')

3.9 Go - 'G' (CR16C core products only)

The 'Go' command allows the user to jump back to IRE Mode from ISP Mode when the defined firmware jump was used to enter ISP Mode (see the section titled "Supporting the 'Go' command (CR16C)" for more details) and terminates further ISP support. The

ISP Firmware Commands

Go - 'G' (CR16C core products only)

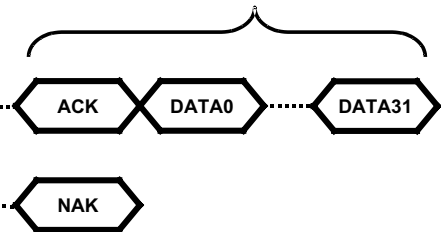
Verify Command Sequence

Sent to ISP code



ISP Response

OR



environment pins must be set for IRE Mode for this command to work, as this command allows the firmware stored at the end of the BOOTAREA to be executed upon its completion. The connection with the host is also torn down. The host should also perform its disconnection routine after issuing this command. The Go command clears the EMPTY bits in the Flash Protection Word and erases any programming routines in the RAM memory before enabling the Watchdog timer. When the Watchdog timer times out, it issues a global reset. After reset the CR16C core examines the status of the environment pins and the EMPTY bits in the flash protection word. If set appropriately, the device will restart in IRE Mode executing the users firmware. After issuing the 'Go' command power should be maintained and no external resets should be applied until the user is certain that the microcontroller is executing the code stored in the Code Area of the flash memory. If the 'Go' command is stopped from clearing the EMPTY bits the microcontroller will start in ISP Mode after the next hardware reset or power

Go - 'G' (CR16C core products only)

cycle. In order to return to IRE Mode the user can use a host device to re-issue the 'Go' command when the microcontroller is restarted.

This command is a single byte command, and requires no response from the ISP firmware.

Description: Resets device into mode defined by environment pins and EMPTY bits.

Operand: Opcode.

Data Returned: None

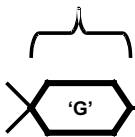
Opcode: 0x47 ('G')

Note: This command is only available in ISP firmware stored on the CR16C core products.

Go Command Sequence

Sent to ISP code

No ISP Response



ISP Firmware Commands

Disconnect - 'D'

3.10 Disconnect - 'D'

The Disconnect command allows the connection between the CR16 device and the host device (such as a PC) to be torn down in readiness for a new connection. The ISP firmware will then reset the USART settings and await a new connection. This command is a single byte command, and requires no response from the ISP firmware.

After disconnection no previous settings can be guaranteed to remain in the device registers. Thus if the user needs to reconnect to the device, any previous settings, such as flash timing values, should be re-programmed and the flash programming routines should be downloaded once again.

Description:	Tears down serial connection between CR16 device and host.
Operand:	Opcode
Data Returned:	None
Opcode:	0x44 ('D')

3.11 Checksum calculation

The checksum for each of the commands is should be calculated as the byte-wise sum of all of the bytes that form the ISP command.

Disconnect Command Sequence

Sent to ISP code

No ISP Response



For example, if the 'Read Byte' command is to be sent, requiring a read from address 0x007000, then the checksum is:

$$0x72 + 0x00 + 0x70 + 0x00 = 0xE2$$

The same command but requesting the byte stored at address 0x2F207E would be:

$$0x72 + 0x2F + 0x20 + 0x7E = 0x3F$$

ISP Firmware Commands

Checksum calculation

4.1 Watchdog Timer

The CR16B core devices include support for a Watchdog timer. This allows the developer to ensure that the ISP firmware is running correctly at all times. This feature has been removed in the latest version of the ISP firmware. However, all current manufactured CR16B core devices contain Watchdog support. Developers should be aware that the dog support section of the code utilises some of the EEPROM memory of the device. If proper care is not taken it is possible that the developer will not be able to re-enter ISP Mode at a later date. If this should occur the only work around is to use a Parallel Flash.

Programmer to reprogram the ISP Memory area. The addresses affected are listed in Table 4-1 below.

Table 4-1. Unavailable EEPROM memory locations in CR16B core devices

Chip ID	Unavailable Memory EEPROM Locations
CR16MHS/MES/MFS/MNS/MUS	0xF278 - 0xF27F
CR16MCS/MAS/MAR/MBR/HCS/ MCT/HCT	0xEFF8 - 0xEFFF

Legacy ISP Firmware

Watchdog support

4.2 Watchdog support

At the beginning of the ISP code the device checks whether a Power-on Reset (POR) occurred or the device started execution after a hardware reset (RESET pin at low level). In case a POR occurred the ISP software enters a loop and waits for a hardware reset for about 260k clock cycles. If there is no hardware reset, the device does not generate a Watchdog signal because it assumes there is no external Watchdog circuitry connected to it. In a case where a hardware reset started the ISP Code, or restarted the ISP Code during the wait loop, all parameters needed to service the Watchdog correctly are read from a Watchdog information field in the Flash EEPROM data memory. This Watchdog info is then used to configure Multi Function Timer 2 and the Watchdog output pin. Timer 2 is set into the Dual Independent Timer Mode and periodically generates a T2A interrupt. The T2A interrupt handler toggles the specified output pin. In case there are no valid data written into the Watchdog info field (checksum incorrect) the default settings predefined within the ISP code will be activated. The default settings can be found in Table 4-3.

4.3 Watchdog Information Field

The last 6 bytes of the low endurance Flash EEPROM data memory contain the Watchdog info field for the ISP software. Table 4-2

Default Settings for CR16MHS9 and CR16MCS9

shows the addresses and functions of each byte of the info field. The value of the start delay is loaded into Counter register T2CNT1, the value of the toggle rate is written into the auto-reload register T2CRA.

Table 4-2. Watchdog Information Field

Function	Address CR16MCS9	Address CR16MHS9
Port Address (word)	EFF8 ₁₆	F278 ₁₆
Toggle Rate (word)	EFFA ₁₆	F27A ₁₆
Start Delay (word)	EFFC ₁₆	F27C ₁₆
Pin Position (byte)	EF FE ₁₆	F27E ₁₆
Checksum (byte)	EFF F ₁₆	F27F ₁₆

4.4 Default Settings for CR16MHS9 and CR16MCS9

The default values are set in order to generate a Watchdog signal that toggles the PF5-pin every 250us after a start delay of 500us at a system clock of 16MHz. These values are not programmed into the Watchdog info field, but are set by the ISP Code as given in Table 4-4.

Legacy ISP Firmware

Default Settings for CR16MHS9 and CR16MCS9

Table 4-3. Default Watchdog Information Field Settings

Function	CR16MCS9	CR16MHS9
Port Address (word)	FD20 ₁₆	FD20 ₁₆
Toggle Rate (word)	1388 ₁₆	1388 ₁₆
Start Delay (word)	3650 ₁₆	7100 ₁₆
Pin Position	20 ₁₆	20 ₁₆

Example for CR16MCS9:

In order to generate a Watchdog signal using a CR16MCS9 device that toggles the PF5-pin every 250us after a start delay of 500us at a system clock of 20MHz, you need to configure the Watchdog information field with the values given in Table 4-4.

Table 4-4. Watchdog Information Field Settings for the CR16MCS9

Function	Address CR16MCS9	Value CR16MCS9
Port Address (word)	EFF8 ₁₆	FD20 ₁₆
Toggle Rate (word)	EFFA ₁₆	1388 ₁₆
Start Delay (word)	EFFC ₁₆	1630 ₁₆

Default Settings for CR16MHS9 and CR16MCS9

Table 4-4. Watchdog Information Field Settings for the CR16MCS9

Function	Address CR16MCS9	Value CR16MCS9
Pin Position (byte)	EFFE ₁₆	20 ₁₆
Checksum (byte)	EFFF ₁₆	1E ₁₆

Example for CR16MHS9

In order to generate a Watchdog signal using a CR16MHS9 device that toggles the PI3-pin every 250us after a start delay of 500us at a system clock of 20MHz, you need to configure the Watchdog info field with the values given in Table 4-5.

Table 4-5. Watchdog Information Field Settings for the CR16MHS9

Function	Address CR16MHS9	Value CR16MHS9
Port Address (word)	F278 ₁₆	FEE0 ₁₆
Toggle Rate (word)	F27A ₁₆	1388 ₁₆
Start Delay (word)	F27C ₁₆	2500 ₁₆
Pin Position (byte)	F27E ₁₆	08 ₁₆
Checksum (byte)	F27F ₁₆	A6 ₁₆

Any port pin which is used for the Watchdog output signal is initialized to low level and will be toggled at a rate which is equal to the

Legacy ISP Firmware

Supporting the 'Go' command (CR16C)

Timer Value divided by the System Clock, e.g. $0x1388/20\text{MHz} = 5000 / 20\text{MHz} = 250\text{us}$ shows the reset sequence and the resulting output signal.

4.5 Supporting the 'Go' command (CR16C)

The 'Go' command allows the ISP user to force the CR16C core to execute their firmware without recourse to physical interaction (i.e. setting of the environment pins and application of a hardware reset or power cycling). However, in order for this feature to function, the CR16C must already be in IRE Mode - the mode required to execute the users firmware, and a boot area must be defined. This feature relies upon the fact that the CR16C core will start up in ISP Mode if the EMPTY bits of Flash Protection Word are set and the environment pins are set to IRE Mode. For the user this means that they can directly jump from their firmware to the ISP firmware in

the boot area of the CR16C device, and also return back to restart their firmware from ISP Mode without applying a hardware reset.

4.6 Jumping the ISP Mode

To jump to ISP Mode, the CR16C core device must currently be in IRE Mode, the CR16C core device must currently be in IRE Mode and executing a program from the flash program memory. As a result of some user interaction the users firmware should perform a jump to address 0x0x000006 using the following assembler routine.

```
movw $0x0003, r3
```

```
movw $0x0000, r4
```

```
jump (r4,r3)
```

Note that any valid register pair can be used to perform the jump.

Once the jump has been performed the users code will no longer have control of the CPU. The code at this address jumps to a routine that sets the EMPTY bits in the Flash Protection Word and then initiates the Watchdog timer. When the Watchdog times out it will reset the device, causing it to start up in ISP Mode.

Legacy ISP Firmware

Jumping the ISP Mode

The jump erases the contents of Information Block 1, which also erases the contents of Main Block 1 (i.e. the contents of Program Flash memory between addresses 0x020000 and 0x03FFFF). Thus if the update fails, the users code, if it also fills Main Block 1, is not guaranteed to function after the 'Go' command is issued.

The routine that the user jumps to takes over control of the CPU, and also requires certain resources and performs certain functions that the user should be aware of. These are listed in the points below.

- Interrupts - all interrupts should be disabled both in the IENAM0/1 registers and the PSR register in the core before performing the jump.
- Watchdog timer - The watchdog timer should be available for the jump routine. If it is already in use in the users application it must be switched off prior to performing the jump. If the module is only being used as a Real-Time Timer, then it should be disabled prior to performing the jump. If the registers of the watchdog timer are **locked** this feature **must not be used**.
- Main Block 1 Flash Memory (0x020000 to 0x03FFFF) - this memory area will be erased during the jump routine.
- Flash Protection Word - It is necessary to set the EMPTY bits of flash protection word to '111b' in order to make the microcontroller restart in ISP Mode. This requires the erasure of Informa-

tion Block 1 (0x80 to 0xFF) which also erases Main Block 1 (see above). Since user data may be stored in Information Block 1, the following routine is performed to ensure that the contents are not corrupted:

1. - A RAM area 128 bytes deep is tested for data storage integrity. If the RAM shows no sign of any 'sticky bits', the code continues at point 2. Otherwise the code will jump to point 5.
2. - The contents of addresses 0x80 to 0xFF in Information Block 1 are copied to RAM.
3. - Information Block 1 and Main Block 1 are erased.
4. - The contents of Information Block 1 are re-programmed from the copy RAM, with the EMPTY bits of the flash protection work set.
5. - The watchdog timer is used to perform a global reset upon time out.

The jump routine requires time to perform its RAM tests and erasure and reprogramming of Information Block 1. During this time the microcontroller must not be reset, and the correct power input must be maintained throughout the process.

Legacy ISP Firmware

Jumping the ISP Mode

ISP RAM Download Files



The file must be downloaded into RAM to enable flash memory programming and page erases is different according to the architecture of the device being programmed. There are currently 4 versions which are listed in Table A-1.

Table A-1. List of files for downloading into RAM to support flash writes and page erases

National Device ID	ISP RAM Code File
CR16MHS/MNS/MPS/MUS	prg_tb30.xpr
CR16MAR/MAS/MCS/MBR/MES/MFS	prg_cb30.xpr
CR16HCS/HCT	prg_cb33.xpr
LMX5100 / CP3BT/LMX9814 (UART and USB)	prg_bl30.xpr

ISP RAM Download Files

Standard Clock Speeds.

B

The ISP Code is preprogrammed to perform flash memory writes using timing registers which are expecting a pre-defined clock source. The timing registers should be re-programmed using the 'Write Byte' or 'Write Word' ISP command if the clock source being used doesn't match that specified for the device as shown in Table B-1. The location of the clock registers can be found in the appropriate data sheets.

For devices where the clock speed of the microcontroller cannot be determined, such as the LMX9814, or where the user has no direct control, such as the LMX5100 with a clock source from the LMX5250, the ISP code will automatically determine the how to set the clock source to 12MHz from an internally stored data table.

Table B-1. Expected clock source frequencies for the various CompactRISC™ microcontrollers

National Device ID	Expected Clock Source Frequency
CR16MHS/MES/MFS/MNS/ MUS	16MHz
CR16MCS/MAS/MAR/MBR/ HCS	16MHz
CR16MCT/HCT	24MHz
LMX5100 / CP3BT/LMX9814 (UART and USB)	12MHz

Standard Clock Speeds.

Notes

National Semiconductor
2900 Semiconductor Drive
PO Box 58090
Santa Clara, CA 95052

Tel: 1-800-272-9959
Fax: 1-800-737-7018

Visit our Web site at::
www.national.com

**For more information, send
Email to:**
support@nsc.com

**National Semiconductor
Europe**

Fax: +49 (0) 180-530 85 86
Email: europe.support@nsc.com
Deutsch Tel: +49 (0) 69 9508 6208
English Tel: +44 (0) 870 24 0 2171
Francais Tel: +33 (0) 1 41 91 8790

**National Semiconductor
Asia Pacific
Customer Response Group**

Tel: 65-254-4466
Fax: 65-250-4466
Email: ap.support@nsc.com

**National Semiconductor
Japan Ltd.**

Tel: 81-3-5639-7560
Fax: 81-3-5639-7507