-

CHAPTER 1
GENERAL INFORMATION

1.1 INTRODUCTION

The In-System-Programming Monitor (ISP-Monitor) allows end-of-line or field programming as well as on-chip debugging of the CR16B based microcontrollers CR16MCS9 and CR16MHS9 through a serial link with a host computer.

A Windows95 based software provides a userfriendly interface using the described functions of the ISP-Monitor. For more details please refer to the WIN95-ISPMon documentation.

1.2 FEATURES

The CR16B ISP-Monitor provides the following features:

- Automatic baudrate detection
- Programming of the Flash program memory
- Programming of the Flash data memory
- Read/Write access to RAM and Peripheral Registers
- Debugging using on-chip Debugging Support (Single step, Breakpoint)
- Programmable watchdog handling if system requires triggering of external watchdog circuits.
- Supports echo handling for single-wire data transmission.

1.3 FUNCTIONAL DESCRIPTION

The ISP-Monitor receives and executes commands from a host computer (PC, ATE or other). The communication to the host is established by a RS-232 link. The UART communication supports both two-wire and single-wire data transmission. The ISP-Monitor automatically checks during the connecting sequence whether the UART receives its own echo on the RxD line and accordingly enables or disabled the echo handling. The Baudrate can be in a range between 9600 Baud and 115200 Baud. The device will automatically adopt to the Baudrate selected by the host.

This monitor provides read/write access to on-chip RAM, Flash EEPROM and Flash program memory. It also allows limited debugging of the application software by utilizing the CR16B core debugging features.

The ISP-Monitor also takes care to service any required triggering of an external watchdog circuit in order to keep system integrity while operating in ISP-Mode. The user is able to define the Watchdog pin(s) and the timing. For that purpose a Watchdog info field in Flash EEPROM data memory is reserved.

1.4 ENTERING THE ISP-MONITOR MODE

There are two possible ways to invoke the ISP-Monitor:

- Hardware ISP-Mode through the environment pins ENV0 and ENV1
- Software ISP-Mode through FLCTRL.EMPTY-bit in IRE-Mode

1.4.1 Hardware ISP-Mode

The Hardware ISP-Mode is entered if ENV1-pin is held high and ENV0-pin is held low during RESET and for at least 32 clock cycles after the RESET pin has gone high.
With this mode set, the device will start execution from the defined ISP address and not from address 0x0000 which is the program entry in the IRE-Mode (Internal ROM Enabled).

Operating in ISP-Mode provides read/write access to the FLCTRL register. Care has to be taken when the user application is operating within this mode (start of the user program using the GO or STEP command) as any unexpected write attempt to the Flash program memory or control registers could lead to an unexpected behavior of the device.

1.4.2 Software ISP-Mode

Another way to enter ISP-Mode is through the FLCTRL register bit FLCTRL.EMPTY which indicates, whether the Flash program memory is empty and hence not yet programmed. If the environment select logic detects IRE environment after RESET and FLCTRL.EMPTY-bit set, it will switch program control to the on-chip boot ROM to allow programming of the internal Flash program memory through the ISP-Monitor.

-

In order to allow further reprogramming of the program memory the ISP code uses FLCTRL bit 7 as an auxiliary EMPTY-bit to determine whether the user program at address 0x0000 should be executed or the ISP program should be continued. As long as In-System reprogramming is required (Hardware ISP-Mode cannot be entered within the system) FLCTRL.EMPTY should not be cleared.

1.5 HOW IN-SYSTEM-PROGRAMMING WORKS

Assuming one or both conditions for entering the ISP-Mode are present, the device will start program execution at the first address of the on-chip boot area.
The ISP code then checks bit 7 of the FLCTRL register and will jump to address 0x0000 if the bit is cleared or continue the ISP program in case it is set.

In order to be able to write to the Flash program memory several routines have to execute out of RAM. Therefore the ISP code transfers part of its code into RAM memory and initializes the part.

If a Power-On reset started the ISP-Monitor, a wait loop determines whether the ISP-Monitor has to support an external Watchdog circuitry. In case a LOW-level on the Reset-pin occurs during this wait time, the ISP code will restart, initialize the Watchdog parameters and start to service the external Watchdog circuitry with either the predefined default settings or the user defined values read out from the Watchdog info field. Otherwise the ISP-Monitor operates without Watchdog support.

The USART1 interface is used to establish a RS-232 communication with the host PC. The ISP-Monitor automatically adopts to the baudrate selected within the PC software. As soon as the user connects to the CR16 device, the host PC starts to send out the character “U” (55_{16}). This pattern is used to measure the bit timing by means of the Multi-Function Timer 1, calculate the baudrate and configure the UART 1 accordingly. At a clock frequency of 8MHz the maximum baudrate is 115kBaud.

The ISP-Monitor code resident in the on-chip boot ROM only supports read/write access to RAM and Flash EEPROM data memory as well as read access to program memory, but does not support write access to program memory. This is done in order to provide a high level of security, because in this case no routines reside on-chip which could accidentally overwrite the application software.

All routines required to program the Flash program memory need to be loaded separately from the host PC into RAM and are executed out of RAM. This is also true for all on-chip debugging routines.

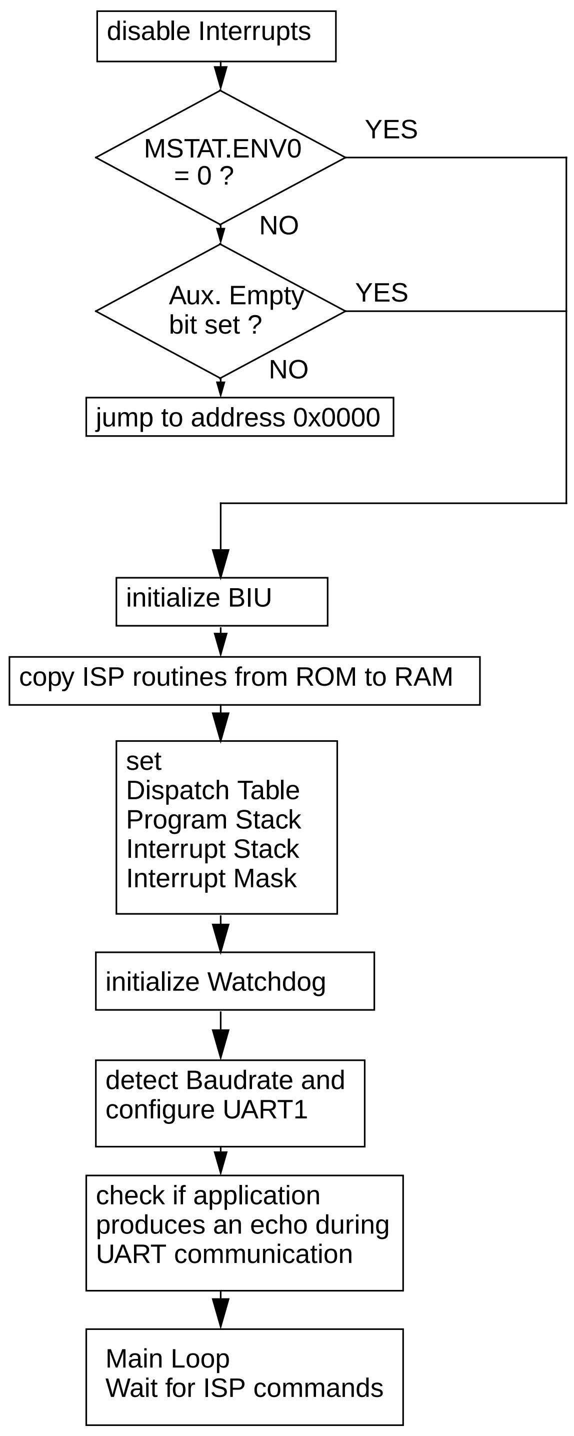
disable Interrupts
MSTAT.ENV0 = 0 ?
YES
NO
Aux. Empty bit set ?
YES
NO
jump to address 0x0000
initialize BIU
copy ISP routines from ROM to RAM
set Dispatch Table Program Stack Interrupt Stack Interrupt Mask
initialize Watchdog
detect Baudrate and configure UART1
check if application produces an echo during UART communication
Main Loop Wait for ISP commands

Figure 1-1. ISP-Monitor Initialization Flow

-

1.6 RESOURCE ALLOCATION

The ISP-Monitor requires the following resources:

- Boot ROM address space
- USART1 and the associated RxD and TxD pins
- Multi Function Timer 1 for baudrate detection
- Multi Function Timer 2 in case Watchdog support is required
- User defined I/O pin(s) for the Watchdog output signal
- RAM

Table 1-1 summarizes the RAM locations used by the CR16MCS9 ISP-Monitor.

Data Type	occupied Address Range	Function
code (.text)	0xC500 ... 0xC5BF	Dispatch Table, T1A Interrupt handler, Jump Table to program and debug routines
code (.text)	0xC500 ... 0xC6E3	as above plus Flash programing routines
code (.text)	0xC500 ... 0xC92D	as above plus Debugging routines
data (.bss)	0xCA00 ... 0xCB65	Variables

Table 1-1. RAM space occupied by the CR16MCS9 ISP-Monitor

-

Table 1-2 summarizes the RAM locations used by the CR16MHS9 ISP-Monitor.

Data Type	occupied Address Range	Function
code (.text)	0xE200 ... 0xE297	Dispatch Table, T1A Interrupt handler, Jump Table to program and debug routines
code (.text)	0xE200 ... 0xE3B5	as above plus Flash programing rou-tines
code (.text)	0xE200 ... 0xE5EF	as above plus Debugging routines
data (.bss)	0xE600 ... 0xE765	Variables

Table 1-2. RAM space occupied by the CR16MHS9 ISP-Monitor

CHAPTER 2
HOST INTERFACE

All communications between the host PC and the CR16B target shall conform to the serial parameters and protocol defined herein.

2.1 PHYSICAL INTERFACE

The ISP-Monitor communicates through a RS-232 three-wire asynchronous serial link with a host computer (TxD/RxD/GND, without hardware handshake). Also single-wire communication is supported by disregarding the echo received on the RxD line. The user has to make sure, that the appropriate level shifting is provided in order to connect to the controller input pins.

2.2 TARGET/HOST SYNCHRONIZATION

The ISP-Monitor has to establish a communication link with the external host. The controller shall attempt to synchronize with the PC by determining the relative baudrate at which it is operating. It will be relative because the controller 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 monitor must determine the prescaler 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 character ’U’ (55_{16}), allowing the target UART to lock on the host’s baudrate.

As soon as the baudrate calculation is done, the controller initializes its USART1 and transmits a certain character dependant on the chip type. The host shall reply with the ’$’ ($24_{16}$) character. After this synchronization the host is able to issue any defined command sequence. In case of a synchronization failure the target will restart the synchronization until a correct re-synchronization is achieved. Table 2-1 on page 8 summarizes the characters used by the host to identify the chip type.

Chip Type	Character
CR16MHS9	‘0’
CR16MCS9	‘1’

Table 2-1. Characters used for Chip Type Identification

2.3 PROTOCOL

Once a communication link has been established between the target and the host, the controller is ready to accept further commands from the host. These commands are listed and defined in the ISP-Monitor commands section. All operations during ISP proceed on the basis of these commands.

The communication does not support any data flow control (handshake). This means, that the host has to be able to receive up to 32 bytes of data with full speed at the selected baudrate.

2.3.1 Data format

The communication with the ISP-Monitor is in standard non-return-to-zero (NRZ) mark/space data format. The following diagrams show examples for possible transmissions. The monitor data format consists of one start bit, eight data bits and one stop bit (no partity check) as shown in Figure 2-1.

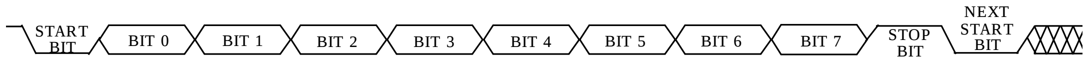

Figure 2-1. Monitor Data Format

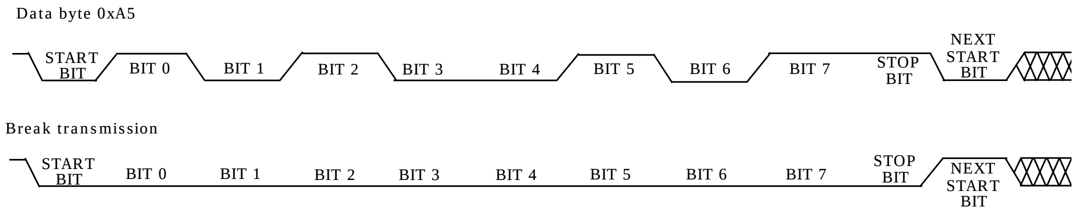

Figure 2-2. Sample Monitor Waveforms

2.3.2 Command Structure

Any ISP-Monitor command follows the same structure. The host starts transmission of a command including a checksum of the transmitted bytes. The ISP-Monitor will check this command sequence and will send back a positive acknowledge (ACK) if no error was encountered during the command transmission or the execution of the command. A negative acknowledgment will indicate that the command was received incorrectly or the command was not successful.

A NAK (0x15) character will immediately terminate the command flow and the ISP-Monitor expects a new command. After three consecutive errors the ISP-Monitor will switch back to the synchronization mode.

Any expected result of a command will be transmitted to the host after the transmission of the ACK (0x06) character which indicates that a command was received correctly and execution is in progress.

CHAPTER 3
ISP-MONITOR COMMANDS

The ISP-Monitor provides the following commands:

ISP-Monitor Commands	ASCII-Opcode
• READB (read memory BYTE)	“r”
• READW (read memory WORD)	“R”
• WRITEB (write memory BYTE)	“w”
• WRITEW (write memory WORD)	“W”
• PROGRAM (program 32 bytes to memory)	“P”
• PROGRAM FLASH (program a 128 bytes page)	“p”
• LOAD (load 32 bytes into RAM)	“L”
• VERIFY (read 32 bytes from memory)	“V”
• GO (execute user program)	“R”
• STEP (single step user program)	“S”
• BREAKPOINT (set hardware breakpoint/condition)	“B”
• STOP (abort user program execution)	“A”

-

3.1 READB (MEMORY BYTE READ)

The READB command will return the byte contents of the specified address. The complete 64k address range can be accessed. The user has to take care and determine whether a byte or a word access is necessary.

Description	Read byte from memory
Operand	3-byte memory address in HB, MB, LB order
Data returned	contents of addressed byte
Opcode	72_{16} ("r")

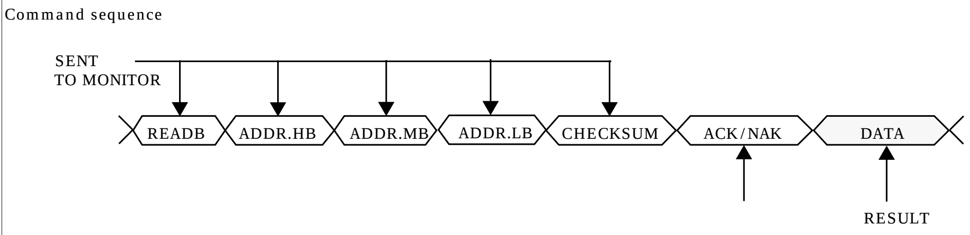

Figure 3-1. Read BYTE Command Transaction

3.2 READW (MEMORY WORD READ)

The READW command will return the word contents of the specified address. The data word is transmitted in two consecutive bytes with high byte first followed by the low byte.

The complete 64k address range can be accessed. The user has to take care and determine whether a byte or a word access is necessary.

Description	Read word from specified address
Operand	3-byte memory address in HB, MB, LB order
Data returned	contents of addressed word : low byte, high byte order
Opcode	52_{16} (“R”)

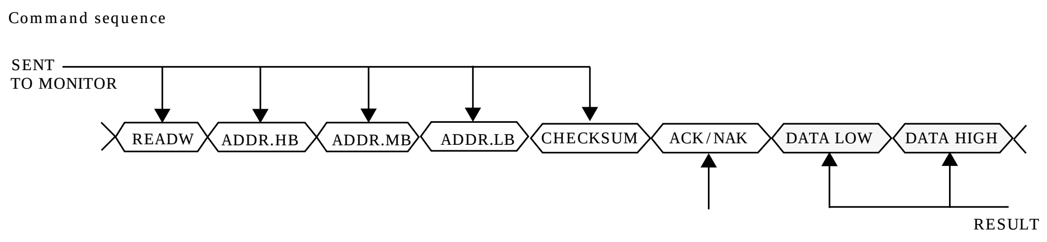

Figure 3-2. Read WORD Command Transaction

-

3.3 WRITEB (MEMORY BYTE WRITE)

The WRITEB command will change the byte contents of the specified address. The complete 64k address range can be accessed. The user has to take care and determine whether a byte or a word access is necessary.

Description	Write byte to memory
Operand	3-byte memory address in HB, MB, LB order followed by data byte
Data returned	None
Opcode	77_{16} ("w")

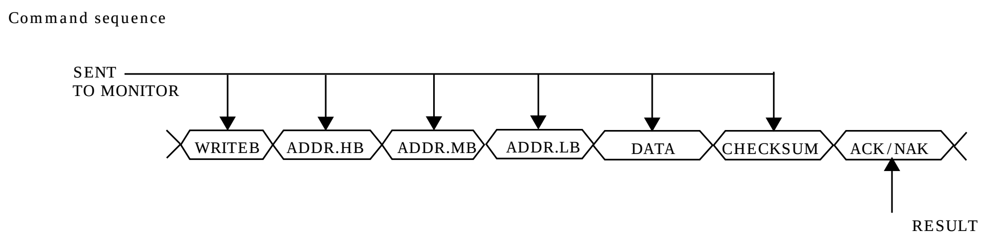

Figure 3-3. Write BYTE Command Transaction

3.4 WRITEW (MEMORY WORD WRITE)

The WRITEW command will change the word contents of the specified address. The complete 64k address range can be accessed. The user has to take care and determine whether a byte or a word access is necessary.

Description	Write word to memory
Operand	3-byte memory address in HB, MB, LB order followed by a WORD data in low byte, high byte order
Data returned	None
Opcode	57_{16} ("W")

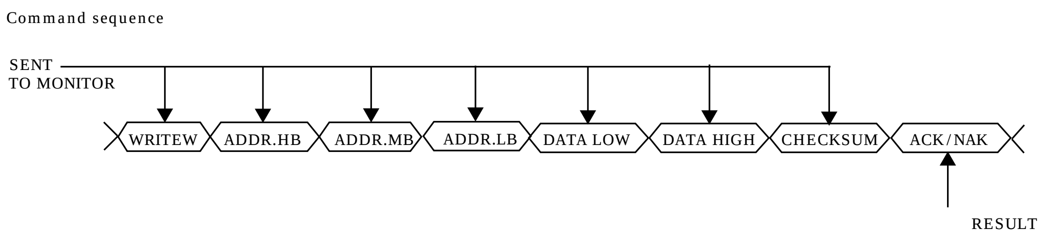

Figure 3-4. Write WORD Command Transaction

3.5 PROGRAM (PROGRAM 32 BYTE TO MEMORY)

The PROGRAM command writes 32 bytes into Flash memory. The complete 64k address range can be accessed. The command expects to receive a complete array of 32 bytes from the host, otherwise the command will not be acknowledged. The Flash programming sequence uses the default Flash programming timing settings. The whole programming sequence is devided into sub-sequences of 8 bytes according to the 4-word page size of the Flash ISP memory. The "Programming Busy" flag (MSTAT.PGMBUSY) is polled in order to guarantee that programming of one word is finished before writing to the next memory location. Each page is erased prior to programming. In case the 32-byte data array exceeds the Flash ISP memory the programming sequence stops after the last Flash data memory page.

This command can for example be used to set the program memory security registers within the last page of the ISP memory. A data stream has to be transmitted starting with the last ISP memory page, containing 8 valid data bytes followed by 24 dummy bytes. The PROGRAM command will stop programming after the last ISP memory page is written and completely ignore the remaining dummy bytes.

Description	Programs 32 byte starting from specified address
Operand	3-byte memory address HB, MB, LB order followed by 32 data bytes
Data returned	None
Opcode	50_{16} ("P")

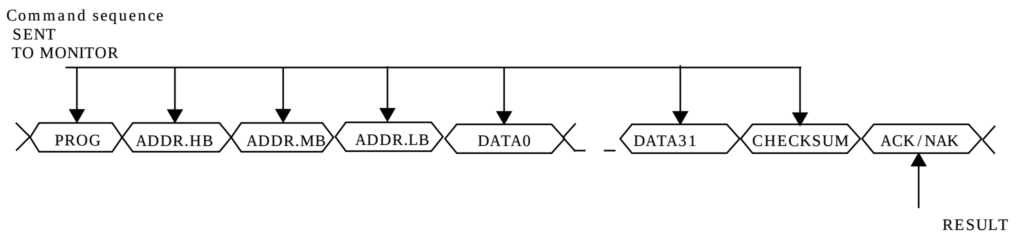

Figure 3-5. PROGRAM Command Transaction

3.6 PROGRAM FLASH (PROGRAM 128 BYTES TO MEMORY PAGE)

The PROGRAM command writes 128 bytes into Flash program memory. The complete 64k address range can be accessed. The command expects to receive a complete array of 128 bytes from the host, otherwise the command will not be acknowledged. The Flash programming sequence uses the default Flash program timing settings. The “Programming Busy” flag (MSTAT.PGMBUSY) is polled in order to guarantee that programming of one word is finished before writing to the next memory location. Each page is erased prior to programming. The user has to make sure that the transmitted data array is alligned to the program memory pages. Otherwise data corruption will occure.

Description	Programs 128 byted starting from specified address
Operand	3-byte memory address HB, MB, LB order followed by 128 data bytes
Data returned	None
Opcode	70_{16} (“p”)

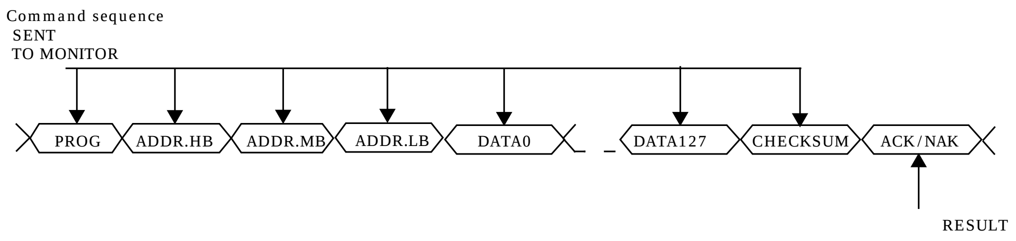

Figure 3-6. PROGRAM FLASH Command Transaction

-

3.7 LOAD (LOAD 32 BYTES INTO RAM)

The LOAD command writes 32 bytes into RAM. The complete 64k address range can be accessed. The command expects to receive a complete array of 32 bytes from the host. Otherwise the command will not be acknowledged.

Description	Load 32 bytes starting from specified address into RAM
Operand	3-byte memory address HB, MB, LB order followed by 32 data bytes
Data returned	None
Opcode	$4C_{16}$ ("L")

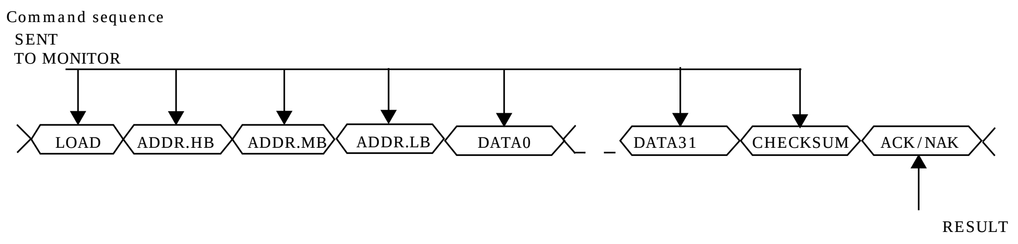

Figure 3-7. LOAD Command Transaction

3.8 VERIFY (READ 32 BYTES FROM MEMORY)

The VERIFY command reads 32 bytes from memory. Data is read and transferred to the host starting at a specified address and followed by the byte contents fo the next higher memory location. The complete 64k address range can be accessed.

Description	Reads 32 byte from memory starting at specified address.
Operand	3-byte memory address in HB, MB, LB order
Data returned	32 bytes of data
Opcode	56_{16} (“V”)

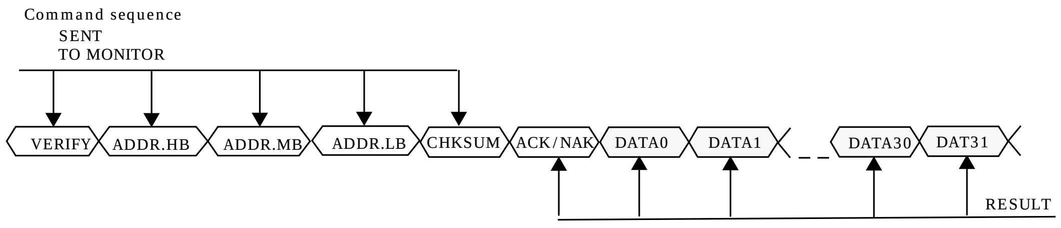

Figure 3-8. VERIFY Command Transaction

-

3.9 GO (EXECUTE USER PROGRAM)

The GO command starts the user progam execution at a specified address. The Debug Condition Enable flag of the CR16B Debug Control register (DCR.DEN) is set in order to enable the core to detect a breakpoint condition upon compare-address match or PC match (see also “Set Breakpoint command”). All CR16B core registers are stored prior to switching to the user program.

Description	Execute user program starting at specified address
Operand	3-byte memory address in HB, MB, LB order
Data returned	None
Opcode	47_{16} (“G”)

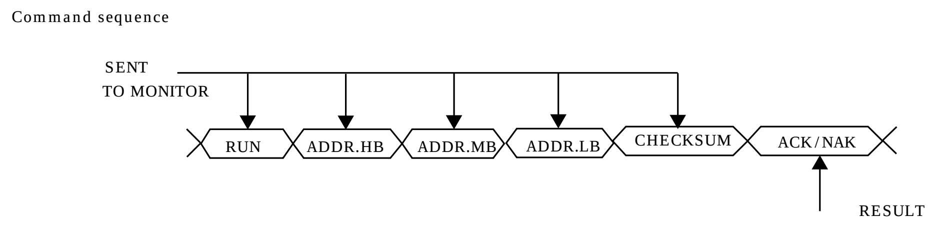

Figure 3-9. GO Command Transaction

3.10 STEP (SINGLE STEP USER PROGRAM)

The STEP command executes one assembler instruction of the user program at a specified address. All CR16B core registers are stored prior to switching to the user program in order to keep the current status of the ISP-Monitor. After one assembler instruction of the user program is finished, program control returns to the ISP-Monitor and the contents of the CR16B core registers are transmitted to the host. Subsequenty the core registers belonging to the user program are stored and the core register contents which belong to the ISP-Monitor are restored into the CR16B core.

Description	Executes one assembler instruction of the user program at a specified address and sends the contents of the CR16 core registers back to the host.
Operand	3-byte memory address of instruction to be executed in HB, MB, LB order
Data returned	next instruction address, CR16B core register contents
Opcode	53_{16} (“S”)

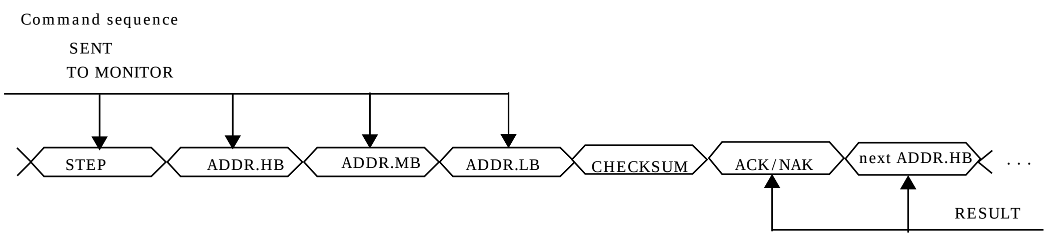

Figure 3-10. STEP Command Transaction

The address of the instruction to be executed next and the CR16B core register contents are transferred to the host PC in the following sequence.

1. next instruction address	<ADDR.HB>	<ADDR.MB>	<ADDR.LB>
2. ISP	<ISP.HB>	<ISP.MB>	<ISP.LB>
3. INTBASE	<INTB.HB>	<INTB.MB>	<INTB.LB>
4. PSR	<PSR.HB>	<PSR.LB>	
5. CFG	<CFG.HB>	<CFG.LB>	
6. R0, R1,..,RA, SP	<REG.HB>	<REG.LB>	

-

3.11 SET BREAKPOINT (SET A BREAKPOINT IN USER PROGRAM)

The SET BREAKPOINT command specifies a break condition within the user program. A complete breakpoint condition consists of the address and the break condition, which can be either Program Counter match (PC match) or address comparison for read and/or write operations. The break address is specified by a 3-byte address, the break condition is specified by setting the according flags in the Debug Control register (DCR). Please refer to “CR16B Programmer’s Reference Manual” for a more detailed description on the CR16B debugging support.

Description	Set a break address in the user program and specify break condition (read, write, read/write, execute (PC match))
Operand	3-byte memory break address in HB, MB, LB order; DCR register contents (break condition, bit DCR.DEN not set)
Data returned	CR16B core register contents
Opcode	42_{16} (“B”)

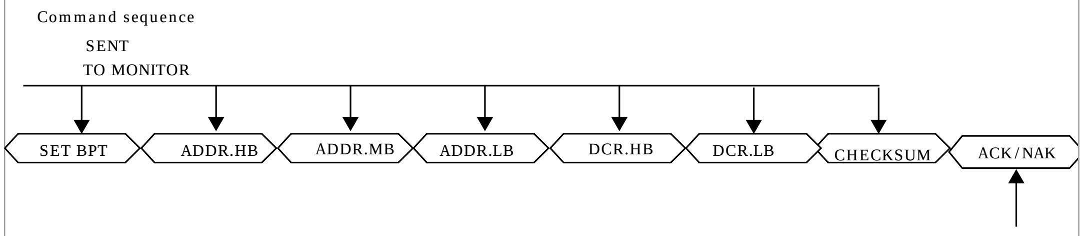

Figure 3-11. SET BREAKPOINT Command Transaction

After the break condition has been reached the address of the instruction to be executed next and the CR16B core register contents are transferred to the host PC in the following sequence.

1. next instruction address	<ADDR.HB>	<ADDR.MB>	<ADDR.LB>
2. ISP	<ISP.HB>	<ISP.MB>	<ISP.LB>
3. INTBASE	<INTB.HB>	<INTB.MB>	<INTB.LB>
4. PSR	<PSR.HB>	<PSR.LB>	
5. CFG	<CFG.HB>	<CFG.LB>	
6. R0, R1,..,RA, SP	<REG.HB>	<REG.LB>	

3.12 ABORT

The ABORT command stops the user program execution at the current location. The current CR16B core register contents are tranmitted to the host and program control returns to the ISP-Monitor.

Description	Stop program execution at the current location and return CR16B core register contents to the host.
Operand	none
Data returned	next instruction address, CR16B core register contents
Opcode	41_{16} ("A")

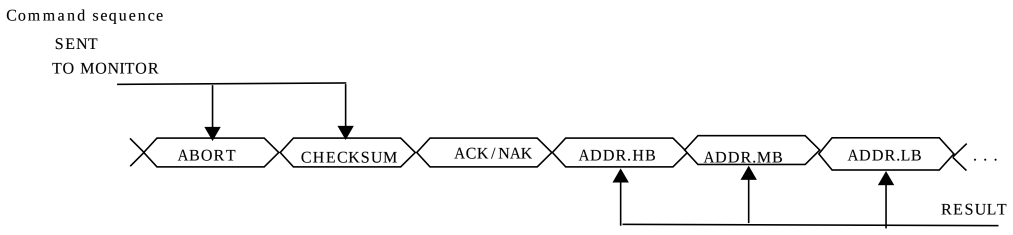

Figure 3-12. SET BREAKPOINT Command Transaction

The address of the instruction to be executed next and the CR16B core register contents are transferred to the host PC in the following sequence.

1. next instruction address	<ADDR.HB>	<ADDR.MB>	<ADDR.LB>
2. ISP	<ISP.HB>	<ISP.MB>	<ISP.LB>
3. INTBASE	<INTB.HB>	<INTB.MB>	<INTB.LB>
4. PSR	<PSR.HB>	<PSR.LB>	
5. CFG	<CFG.HB>	<CFG.LB>	
6. R0, R1,..,RA, SP	<REG.HB>	<REG.LB	

CHAPTER 4
FEATURES OF THE ISP-MONITOR

4.0.1 Watchdog Support

At the beginning of the ISP code the device checks whether a Power-on Reset (POR) occured or the device started execution after a hardware reset (RESET pin at low level). In case a POR occured 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 ciruitry connected to it. In a case where a hardware reset started the ISP-Monitor, or restarted the ISP-Monitor during the wait loop, all parameters needed to service the Watchdog correctly are read from a Watchdog info field in the Flash EEPROM data memory. This Watchdog info is then used to configure the Multi-Function Timer 2 and Watchdog output pin. Timer 2 is set into the Dual Independant Timer Mode and periodically generates a T2A interrupt. The specified output pin is toggled by the T2A interrupt handler. In case there are no valid data written into the Watchdog info field (checksum not correct) the default settings predefined within the ISP code will be activated. The default settings can be found in Table 4-2 on page 24.

Watchdog Info Field

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

The following table shows the addresses and functions of each byte of the info 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)	$EFFE_{16}$	$F27E_{16}$
Checksum (byte)	$EFFF_{16}$	$F27F_{16}$

Table 4-1. Watchdog info field

-

The value of the start delay is loaded into the Counter register T2CNT1, the value for the toggle rate is written into the Auto-reload register T2CRA.

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-Monitor itself as given in table Table 4-3 on page 24.

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 (byte)	20_{16}	20_{16}

Table 4-2. Watchdog Info Field - Default Settings

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 info field with the values given in table Table 4-3 on page 24.

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}
Pin Position (byte)	$EFFE_{16}$	20_{16}
Checksum (byte)	$EFFF_{16}$	$1E_{16}$

Table 4-3. Watchdog Info Field - Example for CR16MCS9

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 4-4 on page 25.

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)	$F27F_{16}$	$A6_{16}$

Table 4-4. Watchdog Info Field - Example for CR16MHS9

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 Timer Value divided by the System Clock, e.g. 0x1388/20MHz = 5000 / 20MHz = 250us. Figure 4-1. shows the reset sequence and the resulting output signal.

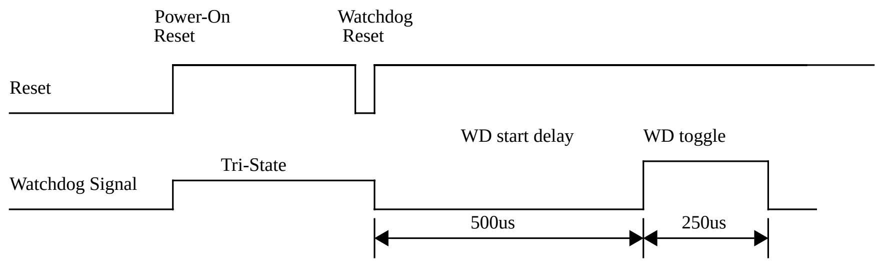

Figure 4-1. Watchdog Output Signal

4.1 ON-CHIP DEBUGGING USING THE ISP MONITOR

4.1.1 Resource Allocation

Since the user software and the ISP-Monitor need to share the on-chip resources during the debugging process , there are some constrains the user has to keep in mind when running an on-chip debugging session.

Memory Resources

The In-System-Programming and debugging routines do not reside entirely in the boot ROM section. There is also a number of routines which occupy on-chip RAM. Therefore the RAM memory range available to the user is restricted to the following address range.

CR16MCS9:	0xC000 ... 0xC4FF (1280 bytes)
CR16MHS9:	0xE000 ... 0xE1FF (512 bytes)

Overwriting of the other RAM location may crash the ISP-Monitor. See “Resource allocation” on page 5.

The program memory is entirely available to the user program.

On-Chip Peripherals

In order to maintain the communication between the device and the host PC the USART 1 interface must stay available to the ISP-Monitor program. Besides the USART1 itself, also the I/O-pins PG6 and PG5 must keep their alternate function as USART1 TX/RX-pins.

4.1.2 Interrupt Processing

The communication between the USART1 and the host PC is interrupt triggered, which means that these interrupts must remain active during both ISP-Monitor and user program execution. In case the user program also uses interrupts, it must not define a new dispatch table for these interrupt vectors, but rather has to load its interrupt vectors into the already existing dispatch table of the ISP-Monitor program. Also the interrupt stack pointer needs to remain unchanged in order to change back and forth between the ISP-Monitor and the user program without crashing the monitor. However, the user must globally enable interrupts to the CR16B core by setting the I-bit and E-bit in the PSR core register. This is because the monitor program keeps a copy of both, the user PSR register and the PSR register of the monitor itself, in order to maintain the current PSR settings of one program when switching to the other. When setting the E-bit and I-bit in the PSR register, care must be taken, that all other bits remain unaffected. For example, clearing the T-bit wil enhibit the detection of a trace trap and thus further stepping through the user program.

-

In order to make it easier for the user to copy the required interrupt vectors into the dispatch table, there are pre-defined include files for both C and assembly language, which can be utilized in the following way:

C-language (National Semiconductor's CompactRISC C-Compiler)

pre-defined interrupt vector location within the dispatch table:

```
typedef void *intvector (void);
#define T1A (*(intvector*) (INT_OFFSET+60))
```

Copy interrupt vector into dispatch table witin the user program:

```
T1A = T1Aint;
```

T1A interrupt handler:

```
#pragma interrupt(T1Aint)
void T1Aint(void){
            // user code
}
```

Assembly Language

pre-defined interrupt vector location within the dispatch table:

```
.set T1A, INT_OFFSET+60
```

Copy interrupt vector into dispatch table witin the user program:

```
.ORG T1A
.WORD   T1Aint
```

T1A interrupt handler:

```
.code_label T1Aint
T1Aint::
            # user code
    RETX
```

Enabled Interrupt Sources within the ISP-Monitor

The ISP-Monitor of the CR16MCS9 device disables all interrupt sources in the ICU by resetting the corresponding bit in the IENAM-registers. The only interrupts enabled are:

- UART1 RX
- UART1 TX
- T2A

Thus, all interrupts to be used within the user program must be enabled by the user software.

The ISP-Monitor of the CR16MHS9 remains all interrupt sources enabled as set by the reset value of the IENAM-register.

4.1.3 Setting Breakpoints

The “SET BREAKPOINT” command can be used to stop the execution of the user program under specified conditions. This debugging feature is supported by the CR16B core, as the core itself is able to trigger a breakpoint under the following conditions:

Breakpoint triggered by a PC Address Match

In this case the CR16B core forces a breakpoint trap as soon as the program counter (PC) is equal the content of a 21-bit Compare-Address Register (CAR). In order to enable a breakpoint upon PC match the PC Match-bit (DCR.PC) and the Debug Condition Enable-bit (DEN.DCR) of the Debug Control Register (DCR) must be set.

Breakpoint triggered by a Compare Address Match

A breakpoint trap may also result from a data access to a certain memory location. A Compare Address Match is detected when the address read or written by the CPU is equal the contents of the CAR register. In order to enable a breakpoint upon Compare Address Match the PC Match-bit (DCR.PC) must be reset and the Debug Condition Enable-bit (DEN.DCR) must be set. The status of the Compare-Address on Write-bit and Compare-Address on Read-bit determine the access type which forces a breakpoint trap execution.

In both cases always the whole CAR register, i.e. CAR[20:0] is used for comparison.

For a more detailed description on the CR16B Debugging Support please refer to the “CR16B Programmer’s Reference Manual”.

Breakpoints within Interrupt Service Routines

After the occurance of a breakpoint condition program control returns to the ISP-Monitor. The ISP-Monitor does not reset any interrupt pending flags (except for the ones it uses itself). Therefore it is necessary to reset the corresponding interrupt pending flag by the first instructions of the interrupt handler. The breakpoint address (PC match) should therefore be set after the pending flag has been cleared. Otherwise this interrupt will be triggered again from within the ISP-Monitor after an execution of a RETX instruction.

-

4.1.4 Watchdog Support during Debugging

If watchdog support has been activated during the initialization phase of the ISP-Monitor, it will remain active during the debug session. In a case where watchdog support is not desired during debugging, the Timer 2 functions have to be disabled as well as its associated output pin(s) at the beginning of the user program.