

[1] [1 [Generated on Wed Jan 31 11:09:59 2007 for DS80C400CLibraries by Doxygen]  
] [Generated on Wed Jan 31 11:09:59 2007 for DS80C400CLibraries by Doxygen

[1] [1 [Generated on Wed Jan 31 11:09:59 2007 for DS80C400CLibraries by Doxygen]  
] [Generated on Wed Jan 31 11:09:59 2007 for DS80C400CLibraries by Doxygen

# DS80C400CLibraries Reference Manual

1

Generated by Doxygen 1.4.3

Wed Jan 31 11:09:59 2007

## Contents

<b>1 DS80C400CLibraries Module Index</b>	<b>1</b>
<b>2 DS80C400CLibraries Directory Hierarchy</b>	<b>1</b>
<b>3 DS80C400CLibraries Data Structure Index</b>	<b>3</b>
<b>4 DS80C400CLibraries File Index</b>	<b>5</b>
<b>5 DS80C400CLibraries Module Documentation</b>	<b>6</b>
<b>6 DS80C400CLibraries Directory Documentation</b>	<b>25</b>
<b>7 DS80C400CLibraries Data Structure Documentation</b>	<b>31</b>
<b>8 DS80C400CLibraries File Documentation</b>	<b>60</b>

## 1 DS80C400CLibraries Module Index

### 1.1 DS80C400CLibraries Modules

Here is a list of all modules:

<b>Initialization module</b>	<b>6</b>
<b>Configuration module</b>	<b>7</b>
<b>Data Access module</b>	<b>21</b>

## 2 DS80C400CLibraries Directory Hierarchy

### 2.1 DS80C400CLibraries Directories

This directory hierarchy is sorted roughly, but not completely, alphabetically:

<b>canbus</b>	<b>25</b>
<b>crypt</b>	<b>25</b>

<b>debugport</b>	<b><a href="#">25</a></b>
<b>dhcp</b>	<b><a href="#">25</a></b>
<b>dirent</b>	<b><a href="#">25</a></b>
<b>dns</b>	<b><a href="#">26</a></b>
<b>err</b>	<b><a href="#">26</a></b>
<b>filesystem_lib</b>	<b><a href="#">26</a></b>
<b>flash</b>	<b><a href="#">26</a></b>
<b>ftplib</b>	<b><a href="#">26</a></b>
<b>http</b>	<b><a href="#">27</a></b>
<b>i2c</b>	<b><a href="#">27</a></b>
<b>isr</b>	<b><a href="#">27</a></b>
<b>kmem</b>	<b><a href="#">27</a></b>
<b>mem</b>	<b><a href="#">27</a></b>
<b>mime</b>	<b><a href="#">28</a></b>
<b>netif</b>	<b><a href="#">28</a></b>
<b>netstat</b>	<b><a href="#">28</a></b>
<b>ntlm</b>	<b><a href="#">28</a></b>
<b>onewire_raw</b>	<b><a href="#">28</a></b>
<b>pop3</b>	<b><a href="#">29</a></b>
<b>rarp</b>	<b><a href="#">29</a></b>
<b>rominit</b>	<b><a href="#">29</a></b>
<b>rtc</b>	<b><a href="#">29</a></b>
<b>smtp</b>	<b><a href="#">29</a></b>
<b>sock</b>	<b><a href="#">30</a></b>
<b>spi</b>	<b><a href="#">30</a></b>

<b>task</b>	<b>30</b>
<b>tftp</b>	<b>30</b>
<b>time</b>	<b>30</b>
<b>useriopoll</b>	<b>31</b>
<b>util</b>	<b>31</b>
<b>xnetboot</b>	<b>31</b>
<b>xnetstack</b>	<b>31</b>

### 3 DS80C400CLibraries Data Structure Index

#### 3.1 DS80C400CLibraries Data Structures

Here are the data structures with brief descriptions:

<b><a href="#">_hostinfo</a></b>	<b>31</b>
<b><a href="#">_http_request</a></b>	<b>32</b>
<b><a href="#">_http_response</a></b>	<b>33</b>
<b><a href="#">_http_session</a></b>	<b>34</b>
<b><a href="#">_http_variable</a></b>	<b>35</b>
<b><a href="#">_mail</a></b>	<b>35</b>
<b><a href="#">_mailheader</a></b>	<b>36</b>
<b><a href="#">_maillist</a></b>	<b>38</b>
<b><a href="#">_pop3_session</a></b>	<b>38</b>
<b><a href="#">_sbufhdr</a></b>	<b>39</b>
<b><a href="#">_type1msg</a></b>	<b>39</b>
<b><a href="#">_type1msghdr</a></b>	<b>40</b>
<b><a href="#">_type2msg</a></b>	<b>41</b>
<b><a href="#">_type2msghdr</a></b>	<b>41</b>

<code>_type3msg</code>	42
<code>_type3msghdr</code>	43
<code>_userheader</code>	44
<b>CanFrame</b> (CAN Frame structure. Denotes the structure of a Transmitted or received CAN frame )	45
<code>dirent</code>	46
<b>FARPTR</b>	46
<code>file_structure</code>	47
<code>hostent</code>	47
<code>in6_addr</code>	48
<code>in_addr</code>	49
<code>kmem_memory</code>	49
<code>mailhostent</code>	50
<b>MCConfig</b> (CAN Message center configuration structure. Used for configuration of receive parameters of Message Centers )	50
<code>netstat_arp_entry</code>	51
<code>netstat_tcp_socket</code>	52
<code>netstat_udp_entry</code>	55
<code>pingdata</code>	56
<code>sockaddr</code>	56
<code>sockaddr_in</code>	57
<b>TCB</b>	58
<b>TIME</b>	59
<code>tm</code>	59

## 4 DS80C400CLibraries File Index

### 4.1 DS80C400CLibraries File List

Here is a list of all documented files with brief descriptions:

<a href="#">dirent.h</a> (Functions for directory listing )	60
<a href="#">rom400_dhcp.h</a> (DHCP functions in the DS80C400 ROM )	65
<a href="#">rom400_err.h</a> (Error codes used by functions in the DS80C400 ROM )	73
<a href="#">rom400_flash.h</a> (Flash programming functions for the TINIm400 module )	75
<a href="#">rom400_http.h</a> (Http Server functions in the DS80C400 ROM )	77
<a href="#">rom400_init.h</a> (ROM Initialization functions in the DS80C400 ROM )	96
<a href="#">rom400_kmem.h</a> (Kernel Memory initialization functions for the DS80C400 ROM )	110
<a href="#">rom400_mem.h</a> (Memory management functions in the DS80C400 ROM )	114
<a href="#">rom400_netif.h</a> (Network interface library for the DS80C400 )	118
<a href="#">rom400_netstat.h</a> (Network statistics library for the DS80C400 )	121
<a href="#">rom400_ow.h</a> (Raw 1-Wire functions in the DS80C400 ROM )	127
<a href="#">rom400_rarp.h</a> (RARP library for the DS80C400 )	130
<a href="#">rom400_sock.h</a> (Socket functions in the DS80C400 ROM )	132
<a href="#">rom400_task.h</a> (Process scheduler functions in the DS80C400 ROM )	174
<a href="#">rom400_tftp.h</a> (TFTP Client functions in the DS80C400 ROM )	191
<a href="#">rom400_useriopoll.h</a> (User IO Poll registration routines for the DS80C400 ROM )	195
<a href="#">rom400_util.h</a> (Utility functions in the DS80C400 ROM )	199
<a href="#">rom400_xnetstack.h</a> (Enhanced network stack for the DS80C400 ROM )	209
<a href="#">stdio.h</a> (File and other IO functions )	216
<a href="#">tini400_canbus.h</a> (CAN Bus Interrupt Driver for DS80C390 / 400 )	244

<a href="#"><b>tini400_crypt.h</b></a> (SHA-1 and MD4 functions for the DS80C400 )	<b>251</b>
<a href="#"><b>tini400_debugport.h</b></a> (Functions supporting the debug port on the TI-NIs400 module )	<b>253</b>
<a href="#"><b>tini400_dns.h</b></a> (DNS Client functions for the DS80C400 ROM )	<b>255</b>
<a href="#"><b>tini400_ftplib.h</b></a> (FTP Client functions for DS80C400 )	<b>261</b>
<a href="#"><b>tini400_isr.h</b></a> (Interrupt Service Routine installation functions )	<b>271</b>
<a href="#"><b>tini400_mime.h</b></a> (MIME Library functions for DS80C400 processor )	<b>278</b>
<a href="#"><b>tini400_ntlm.h</b></a> (NTLM Library functions for DS80C400 processor )	<b>280</b>
<a href="#"><b>tini400_pop3.h</b></a> (Pop3 Library functions for DS80C400 processor )	<b>284</b>
<a href="#"><b>tini400_smtp.h</b></a> (SMTP Library functions for DS80C400 processor )	<b>292</b>
<a href="#"><b>tini400_spi.h</b></a> (SPI library for the TINIm400 module )	<b>300</b>
<a href="#"><b>tini400_time.h</b></a> (Date/Time utilities, tailored for the DS80C400 C Libraries )	<b>304</b>
<a href="#"><b>tini400_xnetboot.h</b></a> (External NetBoot library for the DS80C400 )	<b>307</b>
<a href="#"><b>tini_i2c.h</b></a> (I2C function library )	<b>309</b>
<a href="#"><b>tini_rtc.h</b></a> (RTC function library )	<b>313</b>

## 5 DS80C400CLibraries Module Documentation

### 5.1 Initialization module

#### Functions

- [\*\*uint16\\_t can\\_version\*\*](#) (void)  
*Returns the version number of this CAN library. this function is safe to be called from multiple processes at the same time.*
- void [\*\*can\\_init\*\*](#) (void)  
*Initializes CAN library.*



## 5.1.1 Function Documentation

### 5.1.1.1 void can\_init (void)

Initializes CAN library.

Initializes CAN library. This function has to be called as first function from application before calling other serial library functions. If this function is not called, all other APIs will return error [CAN\\_ERROR\\_NOT\\_INITIALIZED](#).

### 5.1.1.2 uint16\_t can\_version (void)

Returns the version number of this CAN library. this function is safe to be called from multiple processes at the same time.

#### Returns:

Version number of this CAN library.

## 5.2 Configuration module

### Functions

- [int8\\_t can\\_resetcontroller \(uint8\\_t CAN\\_No\)](#)  
*Resets CAN controller.*
- [int8\\_t can\\_setsiestamode \(uint8\\_t CAN\\_No\)](#)  
*Puts the CAN Controller in SIESTA (low power) mode.*
- [int8\\_t can\\_disablecontroller \(uint8\\_t CAN\\_No\)](#)  
*Disables the CAN controller.*
- [int8\\_t can\\_enablecontroller \(uint8\\_t CAN\\_No\)](#)  
*Enables the CAN controller.*
- [int8\\_t can\\_enablecontrollerpassive \(uint8\\_t CAN\\_No\)](#)  
*Enables the CAN controller, but doesn't connect CAN transmit to the bus.*
- [int8\\_t can\\_setrxwriteoverenable \(uint8\\_t CAN\\_No, boolean writeover\)](#)  
*Sets the state of write over in the receiver buffer.*
- [int8\\_t can\\_set11bitglobalidmask \(uint8\\_t CAN\\_No, uint32\\_t \\*mask\)](#)  
*Sets the 11 bit Standard Global Id Mask.*
- [int8\\_t can\\_set29bitglobalidmask \(uint8\\_t CAN\\_No, uint32\\_t \\*mask\)](#)

*Sets the 29 bit Standard Global Id Mask.*

- `int8_t can_set11bitmessagecenter15idmask (uint8_t CAN_No, uint32_t *mask)`

*Sets the global 11 Bit Message Center 15 ID Mask.*

- `int8_t can_set29bitmessagecenter15idmask (uint8_t CAN_No, uint32_t *mask)`

*Sets the global 29 Bit Message Center 15 ID Mask.*

- `int8_t can_setmediaidmask (uint8_t CAN_No, uint16_t mask)`

*Sets the global media ID mask.*

- `int8_t can_setmediaidarbitration (uint8_t CAN_No, uint16_t value)`

*Sets the global media ID arbitration.*

- `int8_t can_setbaudrateprescaler (uint8_t CAN_No, uint16_t prescaler)`

*Sets the basic time quantum (tqu) necessary for CAN communication.*

- `int8_t can_setsynchronizationjumpwidth (uint8_t CAN_No, uint8_t jump-Width)`

*Sets the Synchronization Jump Width necessary for adjusting TSEG1 and TSEG2.*

- `int8_t can_setsamplerate (uint8_t CAN_No, uint8_t sampleRate)`

*Sets the sample rate which is whether to use one or three samples per bit time during CAN communication.*

- `int8_t can_settseg1 (uint8_t CAN_No, uint8_t tseg1)`

*Sets Timing Segment 1 to a specified number of time quanta.*

- `int8_t can_settseg2 (uint8_t CAN_No, uint8_t tseg2)`

*Sets Timing Segment 2 to a specified number of time quanta.*

- `int8_t can_enablemessagecenter (uint8_t CAN_No, uint8_t messageCenter)`

*Puts the message center into Active mode if disabled.*

- `int8_t can_disablemessagecenter (uint8_t CAN_No, uint8_t messageCenter)`

*Puts the message center into Disabled mode if active.*

- `int8_t can_freemessagecenter (uint8_t CAN_No, uint8_t messageCenter)`

*Returns the message center to the free pool.*

- `int8_t can_setmessagecentertx (uint8_t CAN_No, uint8_t messageCenter)`

*Sets Tx/Rx bit of a specific message center to 1 (transmit).*

- `int8_t can_setmessagecenterrx (uint8_t CAN_No, uint8_t messageCenter)`  
*Sets Tx/Rx bit of a specific message center to 1 (receive).*
- `int8_t can_set11bitmessagecenterarbitrationid (uint8_t CAN_No, uint8_t messageCenter, uint32_t *ID)`  
*Sets the 11 bit Arbitration ID.*
- `int8_t can_set29bitmessagecenterarbitrationid (uint8_t CAN_No, uint8_t messageCenter, uint32_t *ID)`  
*Sets the 29 bit Arbitration ID.*
- `int8_t can_setmessagecentermessageidmaskenable (uint8_t CAN_No, uint8_t messageCenter, boolean maskEnable)`  
*Enables or disables Message ID Masking for a specific message center.*
- `int8_t can_setmessagecentermediaidmaskenable (uint8_t CAN_No, uint8_t messageCenter, boolean maskEnable)`  
*Enables or disables Media ID Masking for a specific message center.*

## 5.2.1 Function Documentation

### 5.2.1.1 `int8_t can_disablecontroller (uint8_t CAN_No)`

Disables the CAN controller.

Disables the CAN controller so that the controller is disconnected from the bus preventing any transmissions, or receptions. This is essential to change timing, global masks and other communication critical parameters. Message centers, transmit & receive buffers and message center allotments remain intact.

#### Parameters:

*CAN\_No* - CAN Port number (0 for DS80C400, 0 or 1 for DS80C390)

#### Returns:

`CAN_ERROR_NOERROR` - if successful.

`CAN_ERROR_ARGUMENT` - if improper argument.

`CAN_ERROR_NOT_INITIALIZED` - if CAN controller is not initialized (`can_init()` has not been called).

#### 5.2.1.2 **int8\_t can\_disablemessagecenter (uint8\_t CAN\_No, uint8\_t message-Center)**

Puts the message center into Disabled mode if active.

Puts the message center into Disabled mode if Active. i.e, If the message center is in MC\_RX\_ACTIVE mode, then it is put into MC\_RX\_DISABLED mode, and if the message center is in MC\_AARFR\_ACTIVE mode, then it is put into MC\_AARFR\_DISABLED mode. If the message center is in MC\_TX\_ACTIVE mode, then it is not put into MC\_TX\_DISABLED mode, but is made as MC\_IDLE.

##### **Parameters:**

*CAN\_No* - CAN Port number (0 for DS80C400, 0 or 1 for DS80C390)

*messageCenter* - Message Center number (1 to 15).

##### **Returns:**

[CAN\\_ERROR\\_NOERROR](#) - if successful.

[CAN\\_ERROR\\_ARGUMENT](#) - if improper argument.

[CAN\\_ERROR\\_NOT\\_INITIALIZED](#) - if CAN controller is not initialized ([can\\_init\(\)](#) has not been called).

#### 5.2.1.3 **int8\_t can\_enablecontroller (uint8\_t CAN\_No)**

Enables the CAN controller.

Starts up the CAN controller, and connects to the bus. All critical timing parameters and global masks must already be set. Message centers, transmit & receive buffers and message center allotments remain intact.

##### **Parameters:**

*CAN\_No* - CAN Port number (0 for DS80C400, 0 or 1 for DS80C390)

##### **Returns:**

[CAN\\_ERROR\\_NOERROR](#) - if successful.

[CAN\\_ERROR\\_ARGUMENT](#) - if improper argument.

[CAN\\_ERROR\\_NOT\\_INITIALIZED](#) - if CAN controller is not initialized ([can\\_init\(\)](#) has not been called).

[CAN\\_ERROR\\_INVALID\\_TSEG](#) - TSEG1 or TSEG2 not initialized

#### 5.2.1.4 **int8\_t can\_enablecontrollerpassive (uint8\_t CAN\_No)**

Enables the CAN controller, but doesn't connect CAN transmit to the bus.

Starts up the CAN controller, but doesn't connect CAN transmit to the bus Becomes a quiet listener on the bus. All critical timing parameters and global masks must already be set. Message centers, transmit & receive buffers and message centre allotments remain intact.

**Parameters:**

*CAN\_No* - CAN Port number (0 for DS80C400, 0 or 1 for DS80C390)

**Returns:**

[CAN\\_ERROR\\_NOERROR](#) - if successful.

[CAN\\_ERROR\\_ARGUMENT](#) - if improper argument.

[CAN\\_ERROR\\_NOT\\_INITIALIZED](#) - if CAN controller is not initialized ([can\\_init\(\)](#) has not been called).

[CAN\\_ERROR\\_INVALID\\_TSEG](#) - TSEG1 or TSEG2 not initialized

**5.2.1.5 [int8\\_t](#) can\_enablemessagecenter ([uint8\\_t](#) CAN\_No, [uint8\\_t](#) messageCenter)**

Puts the message center into Active mode if disabled.

Puts the message center into Active mode if disabled. i.e, If the message center is in MC\_RX\_DISABLED mode, then it is put into MC\_RX\_ACTIVE mode. If the message center is in MC\_TX\_DISABLED mode, then it is put into MC\_TX\_ACTIVE mode, and If the message center is in MC\_AARFR\_DISABLED mode, then it is put into MC\_AARFR\_ACTIVE mode. All Message Center settings & changes must be complete before this function is called.

**Parameters:**

*CAN\_No* - CAN Port number (0 for DS80C400, 0 or 1 for DS80C390)

*messageCenter* - Message Center number (1 to 15).

**Returns:**

[CAN\\_ERROR\\_NOERROR](#) - if successful.

[CAN\\_ERROR\\_ARGUMENT](#) - if improper argument.

[CAN\\_ERROR\\_NOT\\_INITIALIZED](#) - if CAN controller is not initialized ([can\\_init\(\)](#) has not been called).

[CAN\\_ERROR\\_GENERIC](#) - if message center is not in disabled mode.

**5.2.1.6 [int8\\_t](#) can\_freemessagecenter ([uint8\\_t](#) CAN\_No, [uint8\\_t](#) messageCenter)**

Returns the message center to the free pool.

Returns the message center to the free pool. The Message Centre will have to be disabled before it can be freed. Else it will return error [CAN\\_ERROR\\_MC\\_ACTIVE](#). In case of Transmit Message centres, it will get disabled as soon as the transmission (of one CAN frame) is completed. In case of Receive Message Centres, it will have to be explicitly disabled (to prevent further reception) using [can\\_disableMessageCenter](#) before it can be freed.

**Parameters:**

*CAN\_No* - CAN Port number (0 for DS80C400, 0 or 1 for DS80C390)

*messageCenter* - Message Center number (1 to 15).

**Returns:**

[CAN\\_ERROR\\_NOERROR](#) - if successful.

[CAN\\_ERROR\\_ARGUMENT](#) - if improper argument.

[CAN\\_ERROR\\_NOT\\_INITIALIZED](#) - if CAN controller is not initialized ([can\\_init\(\)](#) has not been called).

[CAN\\_ERROR\\_MC\\_ACTIVE](#) - if Message Centre is in active transmit or receive mode

**5.2.1.7 [int8\\_t can\\_resetcontroller \(uint8\\_t CAN\\_No\)](#)**

Resets CAN controller.

Resets the CAN controller to its power on default state. But it retains the Tx and Rx buffer contents.

**Parameters:**

*CAN\_No* - CAN Port number (0 for DS80C400, 0 or 1 for DS80C390)

**Returns:**

[CAN\\_ERROR\\_NOERROR](#) - if successful.

[CAN\\_ERROR\\_ARGUMENT](#) - if improper argument.

[CAN\\_ERROR\\_NOT\\_INITIALIZED](#) - if CAN controller is not initialized ([can\\_init\(\)](#) has not been called).

**5.2.1.8 [int8\\_t can\\_set11bitglobalidmask \(uint8\\_t CAN\\_No, uint32\\_t \\* mask\)](#)**

Sets the 11 bit Standard Global Id Mask.

Sets the 11 bit Standard Global Id Mask. The Global ID Mask is used to denote which bits to match in the incoming frame ID. This function requires the CAN port to be disabled else will return error [CAN\\_ERROR\\_PORT\\_ENABLED](#).

**Parameters:**

*CAN\_No* - CAN Port number (0 for DS80C400, 0 or 1 for DS80C390)

*mask* - pointer to mask value

**Returns:**

[CAN\\_ERROR\\_NOERROR](#) - if successful.

[CAN\\_ERROR\\_ARGUMENT](#) - if improper argument.

[CAN\\_ERROR\\_NOT\\_INITIALIZED](#) - if CAN controller is not initialized ([can\\_init\(\)](#) has not been called).

[CAN\\_ERROR\\_PORT\\_ENABLED](#) - if CAN port enabled (if SWINT is not set)

#### 5.2.1.9 `int8_t can_set11bitmessagecenter15idmask (uint8_t CAN_No, uint32_t *mask)`

Sets the global 11 Bit Message Center 15 ID Mask.

Sets the global 11 Bit Message Center 15 ID Mask. Message Center 15 has it's own ID Mask, which is used to denote which bits to match in the Message Center 15 ID. This function requires the CAN port to be disabled else will return error `CAN_ERROR_PORT_ENABLED`.

##### Parameters:

*CAN\_No* - CAN Port number (0 for DS80C400, 0 or 1 for DS80C390)

*mask* - pointer to mask value

##### Returns:

`CAN_ERROR_NOERROR` - if successful.

`CAN_ERROR_ARGUMENT` - if improper argument.

`CAN_ERROR_NOT_INITIALIZED` - if CAN controller is not initialized (`can_init()` has not been called).

`CAN_ERROR_PORT_ENABLED` - if CAN port enabled (if SWINT is not set)

#### 5.2.1.10 `int8_t can_set11bitmessagecenterarbitrationid (uint8_t CAN_No, uint8_t messageCenter, uint32_t *ID)`

Sets the 11 bit Arbitration ID.

Sets the 11 bit Arbitration ID. When this value matches an incoming frame ID subject to the Global ID Mask or Message Center 15 Mask, the incoming frame will be received. This function will also change the specified message center to standard mode, to only respond to 11 bit messages. The Message Centre must be either disabled or free before this function is called, else it will return error `CAN_ERROR_MC_ACTIVE`.

##### Parameters:

*CAN\_No* - CAN Port number (0 for DS80C400, 0 or 1 for DS80C390)

*messageCenter* - Message Center number (1 to 15).

*\*ID* - pointer to the 11 bit arbitration id.

##### Returns:

`CAN_ERROR_NOERROR` - if successful.

`CAN_ERROR_ARGUMENT` - if improper argument.

`CAN_ERROR_NOT_INITIALIZED` - if CAN controller is not initialized (`can_init()` has not been called).

`CAN_ERROR_MC_ACTIVE` - if Message Centre is not disabled.

#### 5.2.1.11 `int8_t can_set29bitglobalidmask (uint8_t CAN_No, uint32_t * mask)`

Sets the 29 bit Standard Global Id Mask.

Sets the 29 bit Standard Global Id Mask. The Global ID Mask is used to denote which bits to match in the incoming frame ID. This function requires the CAN port to be disabled else will return error [CAN\\_ERROR\\_PORT\\_ENABLED](#).

##### Parameters:

*CAN\_No* - CAN Port number (0 for DS80C400, 0 or 1 for DS80C390)

*mask* - pointer to mask value

##### Returns:

[CAN\\_ERROR\\_NOERROR](#) - if successful.

[CAN\\_ERROR\\_ARGUMENT](#) - if improper argument.

[CAN\\_ERROR\\_NOT\\_INITIALIZED](#) - if CAN controller is not initialized ([can\\_init\(\)](#) has not been called).

[CAN\\_ERROR\\_PORT\\_ENABLED](#) - if CAN port enabled (if SWINT is not set)

#### 5.2.1.12 `int8_t can_set29bitmessagecenter15idmask (uint8_t CAN_No, uint32_t * mask)`

Sets the global 29 Bit Message Center 15 ID Mask.

Sets the global 29 Bit Message Center 15 ID Mask. Message Center 15 has it's own ID Mask, which is used to denote which bits to match in the Message Center 15 ID. This function requires the CAN port to be disabled else will return error [CAN\\_ERROR\\_PORT\\_ENABLED](#).

##### Parameters:

*CAN\_No* - CAN Port number (0 for DS80C400, 0 or 1 for DS80C390)

*mask* - pointer to mask value

##### Returns:

[CAN\\_ERROR\\_NOERROR](#) - if successful.

[CAN\\_ERROR\\_ARGUMENT](#) - if improper argument.

[CAN\\_ERROR\\_NOT\\_INITIALIZED](#) - if CAN controller is not initialized ([can\\_init\(\)](#) has not been called).

[CAN\\_ERROR\\_PORT\\_ENABLED](#) - if CAN port enabled (if SWINT is not set)

#### 5.2.1.13 `int8_t can_set29bitmessagecenterarbitrationid (uint8_t CAN_No, uint8_t messageCenter, uint32_t * ID)`

Sets the 29 bit Arbitration ID.



Sets the 29 bit Arbitration ID. When this value matches an incoming frame ID subject to the Global ID Mask or Message Center 15 Mask, the incoming frame will be received. This function will also change the specified message center to extended mode, to only respond to 29 bit messages. The Message Centre must be either disabled or free before this function is called, else it will return error [CAN\\_ERROR\\_MC\\_ACTIVE](#).

**Parameters:**

*CAN\_No* - CAN Port number (0 for DS80C400, 0 or 1 for DS80C390)

*messageCenter* - Message Center number (1 to 15).

*\*ID* - pointer to the 29 bit arbitration id.

**Returns:**

[CAN\\_ERROR\\_NOERROR](#) - if successful.

[CAN\\_ERROR\\_ARGUMENT](#) - if improper argument.

[CAN\\_ERROR\\_NOT\\_INITIALIZED](#) - if CAN controller is not initialized ([can\\_init\(\)](#) has not been called).

[CAN\\_ERROR\\_MC\\_ACTIVE](#) - if Message Centre is in active transmit or receive mode.

**5.2.1.14 [int8\\_t](#) can\_setbaudrateprescaler ([uint8\\_t](#) CAN\_No, [uint16\\_t](#) prescaler)**

Sets the basic time quantum (tqu) necessary for CAN communication.

Sets the basic time quantum (tqu) necessary for CAN communication. It sets the baud rate prescaler from the CPU crystal. The divisor divides straight off the external crystal on the processor. For instance, at 18.432MHz, a divisor of 7 will give you a tqu of 379.774ns. i.e  $tqu = 7 * 1/18.432MHz$ . This function requires the CAN port to be disabled else will return error [CAN\\_ERROR\\_PORT\\_ENABLED](#).

**Parameters:**

*CAN\_No* - CAN Port number (0 for DS80C400, 0 or 1 for DS80C390)

*prescaler* - Prescaler divisor value. Valid range is 1 to 256.

**Returns:**

[CAN\\_ERROR\\_NOERROR](#) - if successful.

[CAN\\_ERROR\\_ARGUMENT](#) - if improper argument.

[CAN\\_ERROR\\_NOT\\_INITIALIZED](#) - if CAN controller is not initialized ([can\\_init\(\)](#) has not been called).

[CAN\\_ERROR\\_PORT\\_ENABLED](#) - if CAN port enabled (if SWINT is not set)

#### 5.2.1.15 `int8_t can_setmediaidarbiration (uint8_t CAN_No, uint16_t value)`

Sets the global media ID arbitration.

Sets the global media ID arbitration value which matches bits in the first two bytes of the incoming frame data area. MSB is First Data byte, and LSB is second data byte. This function requires the CAN port to be disabled else will return error `CAN_ERROR_PORT_ENABLED`.

##### Parameters:

*CAN\_No* - CAN Port number (0 for DS80C400, 0 or 1 for DS80C390)

*value* - media id arbitration value

##### Returns:

`CAN_ERROR_NOERROR` - if successful.

`CAN_ERROR_ARGUMENT` - if improper argument.

`CAN_ERROR_NOT_INITIALIZED` - if CAN controller is not initialized (`can_init()` has not been called).

`CAN_ERROR_PORT_ENABLED` - if CAN port enabled (if SWINT is not set)

#### 5.2.1.16 `int8_t can_setmediaidmask (uint8_t CAN_No, uint16_t mask)`

Sets the global media ID mask.

Sets the global media ID mask which determines what bits to match in the first two bytes of the incoming frame data area. MSB of mask is First Data byte, and LSB is second data byte. This function requires the CAN port to be disabled else will return error `CAN_ERROR_PORT_ENABLED`.

##### Parameters:

*CAN\_No* - CAN Port number (0 for DS80C400, 0 or 1 for DS80C390)

*mask* - mask value

##### Returns:

`CAN_ERROR_NOERROR` - if successful.

`CAN_ERROR_ARGUMENT` - if improper argument.

`CAN_ERROR_NOT_INITIALIZED` - if CAN controller is not initialized (`can_init()` has not been called).

`CAN_ERROR_PORT_ENABLED` - if CAN port enabled (if SWINT is not set)

#### 5.2.1.17 `int8_t can_setmessagecentermediaidmaskenable (uint8_t CAN_No, uint8_t messageCenter, boolean maskEnable)`

Enables or disables Media ID Masking for a specific message center.

Enables or disables Media ID Masking for a specific message center. If masking is disabled, no checks will occur on the first two data bytes. The Message Centre must be either disabled or free before this function is called, else it will return error [CAN\\_ERROR\\_MC\\_ACTIVE](#).

**Parameters:**

*CAN\_No* - CAN Port number (0 for DS80C400, 0 or 1 for DS80C390)

*messageCenter* - Message Center number (1 to 15).

*maskEnable* - 0 to require exact match of ID, non-zero to enable Media ID mask.

**Returns:**

[CAN\\_ERROR\\_NOERROR](#) - if successful.

[CAN\\_ERROR\\_ARGUMENT](#) - if improper argument.

[CAN\\_ERROR\\_NOT\\_INITIALIZED](#) - if CAN controller is not initialized ([can\\_init\(\)](#) has not been called).

[CAN\\_ERROR\\_MC\\_ACTIVE](#) - if Message Centre is in active transmit or receive mode.

**5.2.1.18 [int8\\_t](#) can\_setmessagecentermessageidmaskenable ([uint8\\_t](#) CAN\_No, [uint8\\_t](#) messageCenter, [boolean](#) maskEnable)**

Enables or disables Message ID Masking for a specific message center.

Enables or disables Message ID Masking for a specific message center. If masking is disabled, the message center ID must match ALL bits of incoming ID. The Message Centre must be either disabled or free before this function is called, else it will return error [CAN\\_ERROR\\_MC\\_ACTIVE](#).

**Parameters:**

*CAN\_No* - CAN Port number (0 for DS80C400, 0 or 1 for DS80C390)

*messageCenter* - Message Center number (1 to 15).

*maskEnable* - 0 to require exact match of ID, non-zero to enable global mask.

**Returns:**

[CAN\\_ERROR\\_NOERROR](#) - if successful.

[CAN\\_ERROR\\_ARGUMENT](#) - if improper argument.

[CAN\\_ERROR\\_NOT\\_INITIALIZED](#) - if CAN controller is not initialized ([can\\_init\(\)](#) has not been called).

[CAN\\_ERROR\\_MC\\_ACTIVE](#) - if Message Centre is in active transmit or receive mode.

#### 5.2.1.19 `int8_t can_setmessagecenterrx (uint8_t CAN_No, uint8_t messageCenter)`

Sets Tx/Rx bit of a specific message center to 1 (receive).

Sets Tx/Rx bit of a specific message center to 0 (receive). It doesn't affect the mode of the Message Center. The Message Centre must be either disabled or free before this function is called, else it will return the error `CAN_ERROR_MC_ACTIVE`.

##### Parameters:

*CAN\_No* - CAN Port number (0 for DS80C400, 0 or 1 for DS80C390)

*messageCenter* - Message Center number (1 to 15).

##### Returns:

`CAN_ERROR_NOERROR` - if successful.

`CAN_ERROR_ARGUMENT` - if improper argument.

`CAN_ERROR_NOT_INITIALIZED` - if CAN controller is not initialized (`can_init()` has not been called).

`CAN_ERROR_MC_ACTIVE` - if Message Centre is not disabled.

#### 5.2.1.20 `int8_t can_setmessagecentertx (uint8_t CAN_No, uint8_t messageCenter)`

Sets Tx/Rx bit of a specific message center to 1 (transmit).

Sets Tx/Rx bit of a specific message center to 1 (transmit). It doesn't affect the mode of the Message Center. The Message Centre must be either disabled or free before this function is called, else it will return the error `CAN_ERROR_MC_ACTIVE`. Message Center 15 can't be set to transmit mode.

##### Parameters:

*CAN\_No* - CAN Port number (0 for DS80C400, 0 or 1 for DS80C390)

*messageCenter* - Message Center number (1 to 15).

##### Returns:

`CAN_ERROR_NOERROR` - if successful.

`CAN_ERROR_ARGUMENT` - if improper argument.

`CAN_ERROR_NOT_INITIALIZED` - if CAN controller is not initialized (`can_init()` has not been called).

`CAN_ERROR_MC_ACTIVE` - if Message Centre is not disabled.

#### 5.2.1.21 `int8_t can_setrxwriteoverenable (uint8_t CAN_No, boolean writeover)`

Sets the state of write over in the receiver buffer.

Sets the state of write over in the receiver buffer. If writeover is set to 0, the latest message will be discarded in case of receive buffer overflow. If set to 1, the oldest message in the receive buffer will be discarded.

**Parameters:**

***CAN\_No*** - CAN Port number (0 for DS80C400, 0 or 1 for DS80C390)

***writeover*** - 1 for enable & 0 for disable

**Returns:**

[CAN\\_ERROR\\_NOERROR](#) - if successful.

[CAN\\_ERROR\\_ARGUMENT](#) - if improper argument.

[CAN\\_ERROR\\_NOT\\_INITIALIZED](#) - if CAN controller is not initialized ([can\\_init\(\)](#) has not been called).

**5.2.1.22 [int8\\_t](#) can\_setsamplerate ([uint8\\_t](#) CAN\_No, [uint8\\_t](#) sampleRate)**

Sets the sample rate which is whether to use one or three samples per bit time during CAN communication.

Sets SMP (Sample Rate) which is whether to use one or three samples per bit time during CAN communication. This function requires the CAN port to be disabled else will return error [CAN\\_ERROR\\_PORT\\_ENABLED](#).

**Parameters:**

***CAN\_No*** - CAN Port number (0 for DS80C400, 0 or 1 for DS80C390)

***sampleRate*** - Sample Rate. Valid values are 1 and 3.

**Returns:**

[CAN\\_ERROR\\_NOERROR](#) - if successful.

[CAN\\_ERROR\\_ARGUMENT](#) - if improper argument.

[CAN\\_ERROR\\_NOT\\_INITIALIZED](#) - if CAN controller is not initialized ([can\\_init\(\)](#) has not been called).

[CAN\\_ERROR\\_PORT\\_ENABLED](#) - if CAN port enabled (if SWINT is not set)

**5.2.1.23 [int8\\_t](#) can\_setsiestamode ([uint8\\_t](#) CAN\_No)**

Puts the CAN Controller in SIESTA (low power) mode.

Puts the CAN Controller in SIESTA (low power) mode. When Bus activity is detected, the controller will wake up and participate on the bus.

**Parameters:**

***CAN\_No*** - CAN Port number (0 for DS80C400, 0 or 1 for DS80C390)

**Returns:**

[CAN\\_ERROR\\_NOERROR](#) - if successful.  
[CAN\\_ERROR\\_ARGUMENT](#) - if improper argument.  
[CAN\\_ERROR\\_NOT\\_INITIALIZED](#) - if CAN controller is not initialized ([can\\_init\(\)](#) has not been called).

**5.2.1.24 [int8\\_t can\\_setsynchronizationjumpwidth \(uint8\\_t CAN\\_No, uint8\\_t jumpWidth\)](#)**

Sets the Synchronization Jump Width necessary for adjusting TSEG1 and TSEG2.

Sets the SJW (Synchronization Jump Width) necessary for adjusting TSEG1 and TSEG2 to compensate for sync problems during CAN communication. This function requires the CAN port to be disabled else will return error [CAN\\_ERROR\\_PORT\\_ENABLED](#).

**Parameters:**

*CAN\_No* - CAN Port number (0 for DS80C400, 0 or 1 for DS80C390)  
*jumpWidth* - SJW. valid range is 1 to 4 (1tqu to 4tqu).

**Returns:**

[CAN\\_ERROR\\_NOERROR](#) - if successful.  
[CAN\\_ERROR\\_ARGUMENT](#) - if improper argument.  
[CAN\\_ERROR\\_NOT\\_INITIALIZED](#) - if CAN controller is not initialized ([can\\_init\(\)](#) has not been called).  
[CAN\\_ERROR\\_PORT\\_ENABLED](#) - if CAN port enabled (if SWINT is not set)

**5.2.1.25 [int8\\_t can\\_settseg1 \(uint8\\_t CAN\\_No, uint8\\_t tseg1\)](#)**

Sets Timing Segment 1 to a specified number of time quanta.

Sets TSEG1 (Timing Segment 1 = PROP\_SEG + PHASE\_SEG1) to a specified number of time quanta. This is the timing segment before the bit sample. This function requires the CAN port to be disabled else will return error [CAN\\_ERROR\\_PORT\\_ENABLED](#).

**Parameters:**

*CAN\_No* - CAN Port number (0 for DS80C400, 0 or 1 for DS80C390)  
*tseg1* - Time quanta. Valid range is 2 to 16 (2tqu to 16tqu).

**Returns:**

[CAN\\_ERROR\\_NOERROR](#) - if successful.  
[CAN\\_ERROR\\_ARGUMENT](#) - if improper argument.  
[CAN\\_ERROR\\_NOT\\_INITIALIZED](#) - if CAN controller is not initialized ([can\\_init\(\)](#) has not been called).  
[CAN\\_ERROR\\_PORT\\_ENABLED](#) - if CAN port enabled (if SWINT is not set)

#### 5.2.1.26 `int8_t can_settseg2 (uint8_t CAN_No, uint8_t tseg2)`

Sets Timing Segment 1 to a specified number of time quanta.

Sets TSEG2 (Timing Segment 2 = PHASE\_SEG2) to a specified number of time quanta. This is the timing segment before the bit sample. This function requires the CAN port to be disabled else will return error `CAN_ERROR_PORT_ENABLED`.

##### Parameters:

`CAN_No` - CAN Port number (0 for DS80C400, 0 or 1 for DS80C390)

`tseg2` - Time quanta. Valid range is 2 to 16 (2tqu to 16tqu).

##### Returns:

`CAN_ERROR_NOERROR` - if successful.

`CAN_ERROR_ARGUMENT` - if improper argument.

`CAN_ERROR_NOT_INITIALIZED` - if CAN controller is not initialized (`can_init()` has not been called).

`CAN_ERROR_PORT_ENABLED` - if CAN port enabled (if SWINT is not set)

## 5.3 Data Access module

### Functions

- `int8_t can_sendframe (uint8_t CAN_No, CanFrame *frame)`  
*Transmits a data or RFR frame.*
- `int8_t can_getrxmessagecenter (uint8_t CAN_No, MConfig *config)`  
*Gets the first available message centre and configure it for reception.*
- `int8_t can_receiveframesavailable (uint8_t CAN_No)`  
*Gets the number of frames pending in the receive buffer.*
- `int8_t can_receiveframe (uint8_t CAN_No, CanFrame *frame)`  
*Gets a frame from the receive buffer.*
- `int8_t can_getautoanswerrfrmessagecenter (uint8_t CAN_No, CanFrame *frame)`  
*Gets the first available message centre and configure it for Auto-answering Remote RFRs.*
- `int16_t can_gettxerrorcount (uint8_t CAN_No)`  
*Gets the transmitter error count.*
- `int16_t can_getrxerrorcount (uint8_t CAN_No)`  
*Gets the receiver error count.*

### 5.3.1 Function Documentation

#### 5.3.1.1 `int8_t can_getautoanswerrfrmessagecenter (uint8_t CAN_No, Can-Frame *frame)`

Gets the first available message centre and configure it for Auto-answering Remote RFRs.

Gets the first available message centre and configure it for Auto-answering Remote RFRs. Once enabled, that message center keeps waiting for incoming messages matching the configuration and auto-responds. If the user needs to enable MEME or MDME, then it will have to be done manually after disabling the message center, doing the changes and then re-enabling it. This function returns the Message center number. If there is no free message center, then it returns error `CAN_ERROR_NOFREEMC`.

##### Parameters:

`CAN_No` - CAN Port number (0 for DS80C400, 0 or 1 for DS80C390)

`frame` - Pointer to frame that has to be sent as response.

##### Returns:

Message center number 1 to 14 - if successful.

`CAN_ERROR_ARGUMENT` - if improper argument.

`CAN_ERROR_NOT_INITIALIZED` - if CAN controller is not initialized (`can_init()` has not been called).

`CAN_ERROR_NOFREEMC` - if free message center is available.

#### 5.3.1.2 `int16_t can_getrxerrorcount (uint8_t CAN_No)`

Gets the receiver error count.

Gets the receiver error count.

##### Parameters:

`CAN_No` - CAN Port number (0 for DS80C400, 0 or 1 for DS80C390)

##### Returns:

`CAN_ERROR_NOERROR` - if successful.

`CAN_ERROR_ARGUMENT` - if improper argument.

`CAN_ERROR_NOT_INITIALIZED` - if CAN controller is not initialized (`can_init()` has not been called).

#### 5.3.1.3 `int8_t can_getrxmessagecenter (uint8_t CAN_No, MCConfig *config)`

Gets the first available message centre and configure it for reception.



Gets the first available message centre and configure it for reception according to the parameters specified by the config. Then that message center keeps waiting for incoming messages matching the configuration, which will then be put into the Receive buffer. This function returns the Message center number. If there is no free message center, then it returns error [CAN\\_ERROR\\_NOFREEMC](#).

**Parameters:**

*CAN\_No* - CAN Port number (0 for DS80C400, 0 or 1 for DS80C390)

*config* - Pointer to [MCConfig](#) structure.

**Returns:**

Message center number 1 to 14 - if successful.

[CAN\\_ERROR\\_ARGUMENT](#) - if improper argument.

[CAN\\_ERROR\\_NOT\\_INITIALIZED](#) - if CAN controller is not initialized ([can\\_init\(\)](#) has not been called).

[CAN\\_ERROR\\_NOFREEMC](#) - if free message center is available.

#### 5.3.1.4 [int16\\_t](#) can\_gettxerrorcount ([uint8\\_t](#) CAN\_No)

Gets the transmitter error count.

Gets the transmitter error count.

**Parameters:**

*CAN\_No* - CAN Port number (0 for DS80C400, 0 or 1 for DS80C390)

**Returns:**

[CAN\\_ERROR\\_NOERROR](#) - if successful.

[CAN\\_ERROR\\_ARGUMENT](#) - if improper argument.

[CAN\\_ERROR\\_NOT\\_INITIALIZED](#) - if CAN controller is not initialized ([can\\_init\(\)](#) has not been called).

#### 5.3.1.5 [int8\\_t](#) can\_receiveframe ([uint8\\_t](#) CAN\_No, [CanFrame](#) \*frame)

Gets a frame from the receive buffer.

Gets a frame from the receive buffer. Returns [CAN\\_ERROR\\_BUFEMPTY](#) if the receive buffer doesn't contain any frames. Even if it returns [CAN\\_ERROR\\_FRAMESDROPPED](#), it returns the valid frame from the receive buffer.

**Parameters:**

*CAN\_No* - CAN Port number (0 for DS80C400, 0 or 1 for DS80C390)

*frame* - Pointer to [CanFrame](#) structure to hold the received frame.

**Returns:**

[CAN\\_ERROR\\_NOERROR](#) - if successful.  
[CAN\\_ERROR\\_ARGUMENT](#) - if improper argument.  
[CAN\\_ERROR\\_NOT\\_INITIALIZED](#) - if CAN controller is not initialized ([can\\_init\(\)](#) has not been called).  
[CAN\\_ERROR\\_BUFEMPTY](#) - if receive buffer is empty.  
[CAN\\_ERROR\\_FRAMESDROPPED](#) - if one or more frames had been dropped.

**5.3.1.6 [int8\\_t can\\_receiveframesavailable \(uint8\\_t CAN\\_No\)](#)**

Gets the number of frames pending in the receive buffer.

Gets the number of frames pending in the receive buffer. Returns Zero if no message is pending.

**Parameters:**

*CAN\_No* - CAN Port number (0 for DS80C400, 0 or 1 for DS80C390)

**Returns:**

Number of frames in the receive queue - if successful.  
[CAN\\_ERROR\\_ARGUMENT](#) - if improper argument.  
[CAN\\_ERROR\\_NOT\\_INITIALIZED](#) - if CAN controller is not initialized ([can\\_init\(\)](#) has not been called).

**5.3.1.7 [int8\\_t can\\_sendframe \(uint8\\_t CAN\\_No, CanFrame \\*frame\)](#)**

Transmits a data or RFR frame.

Transmits a data or RFR frame through the first available message center. If no message center is available, it then enqueues the frame into the Transmit buffer. If buffer is full, then it returns [CAN\\_ERROR\\_BUFFERFULL](#). If a previous transmission has an error, then it returns the appropriate error codes.

**Parameters:**

*CAN\_No* - CAN Port number (0 for DS80C400, 0 or 1 for DS80C390)

*frame* - Pointer to [CanFrame](#) structure to be transmitted.

**Returns:**

[CAN\\_ERROR\\_NOERROR](#) - if successful.  
[CAN\\_ERROR\\_ARGUMENT](#) - if improper argument.  
[CAN\\_ERROR\\_NOT\\_INITIALIZED](#) - if CAN controller is not initialized ([can\\_init\(\)](#) has not been called).  
[CAN\\_ERROR\\_PORT\\_DISABLED](#) - If the CAN port is not enabled.  
[CAN\\_ERROR\\_BUFFERFULL](#) - if transmit buffer is full.  
transmit error code - if transmit error has occurred.

## 6 DS80C400CLibraries Directory Documentation

### 6.1 canbus/ Directory Reference

#### Files

- file [tini400\\_canbus.h](#)  
*CAN Bus Interrupt Driver for DS80C390 / 400.*

### 6.2 crypt/ Directory Reference

#### Files

- file [tini400\\_crypt.h](#)  
*SHA-1 and MD4 functions for the DS80C400.*

### 6.3 debugport/ Directory Reference

#### Files

- file [tini400\\_debugport.h](#)  
*Functions supporting the debug port on the TINIs400 module.*

### 6.4 dhcp/ Directory Reference

#### Files

- file [rom400\\_dhcp.h](#)  
*DHCP functions in the DS80C400 ROM.*

### 6.5 dirent/ Directory Reference

#### Files

- file [dirent.h](#)  
*Functions for directory listing.*

## 6.6 dns/ Directory Reference

### Files

- file [tini400\\_dns.h](#)  
*DNS Client functions for the DS80C400 ROM.*

## 6.7 err/ Directory Reference

### Files

- file [rom400\\_err.h](#)  
*Error codes used by functions in the DS80C400 ROM.*

## 6.8 filesystem\_lib/ Directory Reference

### Files

- file [stdio.h](#)  
*File and other IO functions.*

## 6.9 flash/ Directory Reference

### Files

- file [rom400\\_flash.h](#)  
*Flash programming functions for the TINIm400 module.*

## 6.10 ftpclient/ Directory Reference

### Files

- file [tini400\\_ftpclient.h](#)  
*FTP Client functions for DS80C400.*

## 6.11 http/ Directory Reference

### Files

- file [rom400\\_http.h](#)  
*Http Server functions in the DS80C400 ROM.*

## 6.12 i2c/ Directory Reference

### Files

- file [tini\\_i2c.h](#)  
*I2C function library.*

## 6.13 isr/ Directory Reference

### Files

- file [tini400\\_isr.h](#)  
*Interrupt Service Routine installation functions.*

## 6.14 kmem/ Directory Reference

### Files

- file [rom400\\_kmem.h](#)  
*Kernel Memory initialization functions for the DS80C400 ROM.*

## 6.15 mem/ Directory Reference

### Files

- file [rom400\\_mem.h](#)  
*Memory management functions in the DS80C400 ROM.*

## 6.16 mime/ Directory Reference

### Files

- file [tini400\\_mime.h](#)  
*MIME Library functions for DS80C400 processor.*

## 6.17 netif/ Directory Reference

### Files

- file [rom400\\_netif.h](#)  
*Network interface library for the DS80C400.*

## 6.18 netstat/ Directory Reference

### Files

- file [rom400\\_netstat.h](#)  
*Network statistics library for the DS80C400.*

## 6.19 ntlm/ Directory Reference

### Files

- file [tini400\\_ntlm.h](#)  
*NTLM Library functions for DS80C400 processor.*

## 6.20 onewire\_raw/ Directory Reference

### Files

- file [rom400\\_ow.h](#)  
*Raw 1-Wire functions in the DS80C400 ROM.*

## 6.21 pop3/ Directory Reference

### Files

- file [tini400\\_pop3.h](#)  
*Pop3 Library functions for DS80C400 processor.*

## 6.22 rarp/ Directory Reference

### Files

- file [rom400\\_rarp.h](#)  
*RARP library for the DS80C400.*

## 6.23 rominit/ Directory Reference

### Files

- file [rom400\\_init.h](#)  
*ROM Initialization functions in the DS80C400 ROM.*

## 6.24 rtc/ Directory Reference

### Files

- file [tini\\_rtc.h](#)  
*RTC function library.*

## 6.25 smtp/ Directory Reference

### Files

- file [tini400\\_smtp.h](#)  
*SMTP Library functions for DS80C400 processor.*

## 6.26 sock/ Directory Reference

### Files

- file [rom400\\_sock.h](#)  
*Socket functions in the DS80C400 ROM.*

## 6.27 spi/ Directory Reference

### Files

- file [tini400\\_spi.h](#)  
*SPI library for the TINIm400 module.*

## 6.28 task/ Directory Reference

### Files

- file [rom400\\_task.h](#)  
*Process scheduler functions in the DS80C400 ROM.*

## 6.29 tftp/ Directory Reference

### Files

- file [rom400\\_tftp.h](#)  
*TFTP Client functions in the DS80C400 ROM.*

## 6.30 time/ Directory Reference

### Files

- file [tini400\\_time.h](#)  
*Date/Time utilities, tailored for the DS80C400 C Libraries.*



### 6.31 useriopoll/ Directory Reference

#### Files

- file [rom400\\_useriopoll.h](#)

*User IO Poll registration routines for the DS80C400 ROM.*

### 6.32 util/ Directory Reference

#### Files

- file [rom400\\_util.h](#)

*Utility functions in the DS80C400 ROM.*

### 6.33 xnetboot/ Directory Reference

#### Files

- file [tini400\\_xnetboot.h](#)

*External NetBoot library for the DS80C400.*

### 6.34 xnetstack/ Directory Reference

#### Files

- file [rom400\\_xnetstack.h](#)

*Enhanced network stack for the DS80C400 ROM.*

## 7 DS80C400CLibraries Data Structure Documentation

### 7.1 \_hostinfo Struct Reference

```
#include <tini400_smtp.h>
```

### 7.1.1 Detailed Description

Structure for host configuration information that has to be registered with smtp library

#### Data Fields

- long [dns\\_primary\\_address](#)  
*primary dns server IP address*
- long [dns\\_secondary\\_address](#)  
*secondary dns server IP address*
- long [dns\\_timeout](#)  
*dns server response timeout*
- long [mailqueue\\_timeinterval](#)  
*interval time before resend queued mails*
- long [smtp\\_host](#)  
*IP address of SMTP host, if IP address is zero, smtp library look for IP address through DNS library calls.*
- char \* [localhostname](#)  
*char pointer that holds local host name value*

The documentation for this struct was generated from the following file:

- [tini400\\_smtp.h](#)

## 7.2 \_http\_request Struct Reference

```
#include <rom400_http.h>
```

### 7.2.1 Detailed Description

Structure for http request

## Data Fields

- char [path](#) [HTTP\_MAX\_URL]  
*URL path name.*
- char [request\\_method](#)  
*Request method flag.*
- char \* [query\\_string](#)  
*Query string value passed in http request.*
- char \* [req\\_headers](#)  
*String holds http request headers.*
- char \* [message\\_body](#)  
*Message body value passed in http request.*
- long [contentlength](#)  
*Contains content length value passed in http request.*
- [http\\_variable](#) \* [varlist](#)  
*Http variable list.*

The documentation for this struct was generated from the following file:

- [rom400\\_http.h](#)

## 7.3 \_http\_response Struct Reference

```
#include <rom400_http.h>
```

### 7.3.1 Detailed Description

Structure for http response

## Data Fields

- char \* [res\\_headers](#)  
*String holds http response headers.*

- char [response](#) [HTTP\_MAX\_BUFSIZE]  
*Response code and string.*
- char [content\\_type](#) [HTTP\_MAX\_BUFSIZE]  
*Content type of response message.*
- int [contentlength](#)  
*Length of response message body.*

The documentation for this struct was generated from the following file:

- [rom400\\_http.h](#)

## 7.4 [\\_http\\_session](#) Struct Reference

```
#include <rom400_http.h>
```

### 7.4.1 Detailed Description

Structure for http session

#### Data Fields

- int [sock\\_handler](#)  
*socket handler for client connection*
- [sockaddr](#) [address](#)  
*client socket address*
- [http\\_request](#) [request](#)  
*http request*
- [http\\_response](#) [response](#)  
*http response*

The documentation for this struct was generated from the following file:

- [rom400\\_http.h](#)

## 7.5 `_http_variable` Struct Reference

```
#include <rom400_http.h>
```

### 7.5.1 Detailed Description

Structure for http variable names and values

#### Data Fields

- `char * var\_name`  
*Http variable name.*
- `char * value`  
*Http variable value.*
- `http\_variable * next`  
*Next http variable node address, NULL value to indicate end of the list.*

The documentation for this struct was generated from the following file:

- [rom400\\_http.h](#)

## 7.6 `_mail` Struct Reference

```
#include <tini400_pop3.h>
```

### 7.6.1 Detailed Description

Structure for mail that contains standard mail header, user mail header, message and attachment filename list

#### Data Fields

- `\_mailheader mailhdr`  
*standard mailheader structure contains standard mail header values*
- `\_userheader userhdr`  
*user mailheader structure contains user defined mail header name list and value list*

- char \* [msg](#)  
*string holds mail message*
- char \* [attachmentlist](#) [POP3\_MAXATTACHMENTSIZ]  
*array of string contains attachment file list*

The documentation for this struct was generated from the following file:

- [tini400\\_pop3.h](#)

## 7.7 [\\_mailheader](#) Struct Reference

```
#include <tini400_pop3.h>
```

### 7.7.1 Detailed Description

Structure for standard mail header holds standard mail header values

#### Data Fields

- char \* [from\\_id](#)  
*string contains from id mailheader value*
- char \* [sendername](#)  
*string contains sendername mailheader value*
- char \* [to\\_id](#)  
*string contains to id mailheader value*
- char \* [recipientname](#)  
*string contains recipientname mailheader value*
- char \* [subject](#)  
*string contains subject mailheader value*
- char \* [reply\\_to\\_id](#)  
*string contains reply\_to\_id mailheader value*
- char \* [cc\\_id](#)  
*string contains cc\_id mailheader value*

- char \* [bcc\\_id](#)  
*string contains bcc\_id mailheader value*
- char \* [errors\\_to\\_id](#)  
*string contains errors\_to\_id mailheader value*
- char \* [date](#)  
*string contains date mailheader value*
- char \* [from\\_id](#)  
*string contains from id mailheader value*
- char \* [sendername](#)  
*string contains sendername mailheader value*
- char \* [to\\_id](#)  
*string contains to\_id mailheader value*
- char \* [recipientname](#)  
*string contains recipientname mailheader value*
- char \* [subject](#)  
*string contains subject mailheader value*
- char \* [reply\\_to\\_id](#)  
*string contains reply\_to\_id mailheader value*
- char \* [cc\\_id](#)  
*string contains cc\_id mailheader value*
- char \* [bcc\\_id](#)  
*string contains bcc\_id mailheader value*
- char \* [errors\\_to\\_id](#)  
*string contains errors\_to\_id mailheader value*
- char \* [date](#)  
*string contains date mailheader value*

The documentation for this struct was generated from the following files:

- [tini400\\_pop3.h](#)
- [tini400\\_smtp.h](#)

## 7.8 `_maillist` Struct Reference

```
#include <tini400_pop3.h>
```

### 7.8.1 Detailed Description

Structure for maillist

#### Data Fields

- int [numberofmails](#)  
*number of mails value*
- int \* [mailnumberlist](#)  
*array of integers with mail number list*
- int \* [mailsizelist](#)  
*array of integers with size of each mail*

The documentation for this struct was generated from the following file:

- [tini400\\_pop3.h](#)

## 7.9 `_pop3_session` Struct Reference

```
#include <tini400_pop3.h>
```

### 7.9.1 Detailed Description

Structure for pop3\_session

#### Data Fields

- unsigned int [handle](#)  
*socket handler*
- char \* [user](#)  
*username value*
- char \* [pass](#)



*password value*

- int [status](#)  
*status of pop3 session*
- int(\* [pop3\\_authentication](#) )()  
*address of pop3 authentication callback function*

The documentation for this struct was generated from the following file:

- [tini400\\_pop3.h](#)

## 7.10 [\\_sbufhdr](#) Struct Reference

```
#include <tini400_ntlm.h>
```

### 7.10.1 Detailed Description

Structure for security buffer header

#### Data Fields

- unsigned int [len](#)  
*length of the data*
- unsigned int [buflen](#)  
*length of the security buffer*
- unsigned long [start\\_loc](#)  
*starting address of the data*

The documentation for this struct was generated from the following file:

- [tini400\\_ntlm.h](#)

## 7.11 [\\_type1msg](#) Struct Reference

```
#include <tini400_ntlm.h>
```

### 7.11.1 Detailed Description

Structure for type1 message

#### Data Fields

- [type1msghdr t1hdr](#)  
*type 1 message header*
- unsigned char [buf](#) [1024]  
*security buffer*
- unsigned int [buf\\_index](#)  
*security buffer length*

The documentation for this struct was generated from the following file:

- [tini400\\_ntlm.h](#)

## 7.12 \_type1msghdr Struct Reference

```
#include <tini400_ntlm.h>
```

### 7.12.1 Detailed Description

Structure for type1 message header

#### Data Fields

- char [signature](#) [8]  
*char array to store NTLM signature*
- unsigned long [msgtype](#)  
*NTLM Message Type.*
- unsigned long [flags](#)  
*The NTLM flags.*
- [sbufhdr usr](#)  
*user name security buffer header*

- [sbufhdr domain](#)  
*domain name security buffer header*

The documentation for this struct was generated from the following file:

- [tini400\\_ntlm.h](#)

## 7.13 `_type2msg` Struct Reference

```
#include <tini400_ntlm.h>
```

### 7.13.1 Detailed Description

Structure for type2 message

#### Data Fields

- [type2msghdr t2hdr](#)  
*char array to store NTLM signature*
- unsigned char [buf](#) [1024]  
*security buffer*
- unsigned int [buf\\_index](#)  
*security buffer length*

The documentation for this struct was generated from the following file:

- [tini400\\_ntlm.h](#)

## 7.14 `_type2msghdr` Struct Reference

```
#include <tini400_ntlm.h>
```

### 7.14.1 Detailed Description

Structure for type2 message header

## Data Fields

- char [signature](#) [8]  
*char array to store NTLM signature*
- unsigned long [msgtype](#)  
*The NTLM message type.*
- [sbufhdr domain](#)  
*domain name security buffer header*
- unsigned long [flags](#)  
*The NTLM flags.*
- unsigned char [challenge](#) [8]  
*the 8 byte server challenge*
- unsigned char [context](#) [8]  
*reserved for future use*
- [sbufhdr targetinfo](#)  
*target information.*

The documentation for this struct was generated from the following file:

- [tini400\\_ntlm.h](#)

## 7.15 \_type3msg Struct Reference

```
#include <tini400_ntlm.h>
```

### 7.15.1 Detailed Description

Structure for type3 message

## Data Fields

- [type3msghdr t3hdr](#)  
*char array to store NTLM signature*

- unsigned char [buf](#) [1024]  
*security buffer*
- unsigned int [buf\\_index](#)  
*security buffer length*

The documentation for this struct was generated from the following file:

- [tini400\\_ntlm.h](#)

## 7.16 \_type3msghdr Struct Reference

```
#include <tini400_ntlm.h>
```

### 7.16.1 Detailed Description

Structure for type3 message header

#### Data Fields

- char [signature](#) [8]  
*char array to store NTLM signature*
- unsigned long [msgtype](#)  
*The NTLM message type.*
- [sbufhdr lmresponse](#)  
*lan manager response*
- [sbufhdr ntlmresponse](#)  
*network lan manager response*
- [sbufhdr domain](#)  
*domain name buffer header*
- [sbufhdr usr](#)  
*user name buffer header*
- [sbufhdr workstation](#)  
*workstation name buffer header*

- [sbufhdr session](#)  
*session buffer header.*
- unsigned long [flags](#)  
*The NTLM flags.*

The documentation for this struct was generated from the following file:

- [tini400\\_ntlm.h](#)

## 7.17 `_userheader` Struct Reference

```
#include <tini400_pop3.h>
```

### 7.17.1 Detailed Description

Structure for user defined mail header contains user header name list and user header value list

#### Data Fields

- char \* [headernamelist](#) [POP3\_MAXUSERHEADERSIZE]  
*array of string contains user mail header name list*
- char \* [headervalluelist](#) [POP3\_MAXUSERHEADERSIZE]  
*array of string contains user mail header value list*
- char \* [headernamelist](#) [SMTP\_MAXUSERHEADERSIZE]  
*array of string contains user mail header name list*
- char \* [headervalluelist](#) [SMTP\_MAXUSERHEADERSIZE]  
*array of string contains user mail header value list*

The documentation for this struct was generated from the following files:

- [tini400\\_pop3.h](#)
- [tini400\\_smtp.h](#)

## 7.18 CanFrame Struct Reference

```
#include <tini400_canbus.h>
```

### 7.18.1 Detailed Description

CAN Frame structure. Denotes the structure of a Transmitted or received CAN frame.

#### Data Fields

- [boolean RemoteFrameRequest](#)
- [boolean ExtendedID](#)
- [uint32\\_t ID](#)
- [uint8\\_t Length](#)
- [char Data](#) [8]

### 7.18.2 Field Documentation

#### 7.18.2.1 [char CanFrame::Data](#)[8]

Array containing the transmitted/received data

#### 7.18.2.2 [boolean CanFrame::ExtendedID](#)

Flag indicates whether the identifier is in Standard or Extended format

#### 7.18.2.3 [uint32\\_t CanFrame::ID](#)

Common for 11-bit (standard) and 29-bit (extended) Arbitration IDs, selectable by the ExtendedID flag (1 = Extended, 0 = Standard)

#### 7.18.2.4 [uint8\\_t CanFrame::Length](#)

Number of bytes contained in the frame

#### 7.18.2.5 [boolean CanFrame::RemoteFrameRequest](#)

Flag denotes to transmit a RFR in case of frame to be transmitted, and EXTRQ (RFR received) in case of Received frame

The documentation for this struct was generated from the following file:

- [tini400\\_canbus.h](#)

## 7.19 dirent Struct Reference

```
#include <dirent.h>
```

### 7.19.1 Detailed Description

Structure used to return the name of a directory listing entry.

#### Data Fields

- unsigned long [d\\_ino](#)  
*File serial number.*
- char [d\\_name](#) [256]  
*Name of the file.*

The documentation for this struct was generated from the following file:

- [dirent.h](#)

## 7.20 FARPTR Struct Reference

```
#include <rom400_task.h>
```

### 7.20.1 Detailed Description

Structure that defines a raw 24-bit memory pointer. Unlike void far\*, this pointer cannot be directly used in Keil C. To convert it into a far pointer, increase the highest byte by 1 and set the top bit depending on the memory space.

#### Data Fields

- unsigned char [msb](#)  
*Most significant (raw) byte of the memory address.*
- unsigned short [offset](#)  
*Offset within the 64KB segment of msb.*

The documentation for this struct was generated from the following file:

- [rom400\\_task.h](#)



## 7.21 file\_structure Struct Reference

```
#include <stdio.h>
```

### 7.21.1 Detailed Description

Structure for FILE object. Includes file flags, last error code, file type, and a pointer to the file descriptor.

#### Data Fields

- int [flags](#)  
*Flags for the file. Can denote the EOF is reached, or that file is temporary.*
- int [error](#)  
*Last error code for the file.*
- int [type](#)  
*File type. currently on the [FILE\\_TYPE\\_TINIFS](#) is supported.*
- void \* [fd](#)  
*Pointer to the file descriptor, used internally by the TINI File System.*
- unsigned char \* [fname\\_copy](#)  
*Copy of the name of the file used internally. Destroyed on [fclose](#).*

The documentation for this struct was generated from the following file:

- [stdio.h](#)

## 7.22 hostent Struct Reference

```
#include <tini400_dns.h>
```

### 7.22.1 Detailed Description

Structure for host information that will be returned by the DNS client functions.

## Data Fields

- char \* [h\\_name](#)  
*String with the official name of the host.*
- char \*\* [h\\_aliases](#)  
*String with alternative host names.*
- int [h\\_addrtype](#)  
*Address type ([AF\\_INET](#) or [AF\\_INET6](#)).*
- int [h\\_length](#)  
*Length of the address.*
- char \*\* [h\\_addr\\_list](#)  
*List of network addresses, each of [h\\_length](#) bytes. The list is null-terminated.*

The documentation for this struct was generated from the following file:

- [tini400\\_dns.h](#)

## 7.23 in6\_addr Struct Reference

```
#include <rom400_sock.h>
```

### 7.23.1 Detailed Description

Structure representing a 16 byte IPv6 address.

## Data Fields

- unsigned char [s6\\_addr](#) [16]  
*IPv6 compatible address.*

The documentation for this struct was generated from the following file:

- [rom400\\_sock.h](#)

## 7.24 in\_addr Struct Reference

```
#include <rom400_sock.h>
```

### 7.24.1 Detailed Description

Structure representing a 4 byte IPv4 address, for use with the [sockaddr\\_in](#) structure.

#### Data Fields

- unsigned long [s\\_addr](#)  
*Address as an unsigned long (32 bits).*

The documentation for this struct was generated from the following file:

- [rom400\\_sock.h](#)

## 7.25 kmem\_memory Struct Reference

```
#include <rom400_kmem.h>
```

### 7.25.1 Detailed Description

Structure for storing kernel memory size and count.

#### Data Fields

- unsigned char [size\\_90](#)  
*Count of blocks of size 90 bytes.*
- unsigned char [size\\_256](#)  
*Count of blocks of size 256 bytes.*
- unsigned char [size\\_512](#)  
*Count of blocks of size 512 bytes.*
- unsigned char [size\\_768](#)  
*Count of blocks of size 768 bytes.*
- unsigned char [size\\_1024](#)

*Count of blocks of size 1024 bytes.*

- unsigned char [size\\_1280](#)  
*Count of blocks of size 1280 bytes.*
- unsigned char [size\\_1600](#)  
*Count of blocks of size 1600 bytes.*

The documentation for this struct was generated from the following file:

- [rom400\\_kmem.h](#)

## 7.26 mailhostent Struct Reference

```
#include <tini400_dns.h>
```

### 7.26.1 Detailed Description

Structure for host information requested with an MX record type.

**See also:**

[dns\\_getmx](#)

#### Data Fields

- char \* [h\\_name](#)  
*String with the name of a mail host.*
- int [preference](#)  
*Preference value reported by the DNS query.*

The documentation for this struct was generated from the following file:

- [tini400\\_dns.h](#)

## 7.27 MConfig Struct Reference

```
#include <tini400_canbus.h>
```

### 7.27.1 Detailed Description

CAN Message center configuration structure. Used for configuration of receive parameters of Message Centers.

#### Data Fields

- [boolean ExtendedID](#)
- [uint32\\_t ID](#)
- [boolean MemeEnable](#)
- [boolean MdmeEnable](#)

### 7.27.2 Field Documentation

#### 7.27.2.1 [boolean MConfig::ExtendedID](#)

Flag indicates whether the identifier is in Standard or Extended format

#### 7.27.2.2 [uint32\\_t MConfig::ID](#)

Common for 11-bit (standard) and 29-bit (extended) Arbitration IDs, selectable by the ExtendedID flag (1 = Extended, 0 = Standard)

#### 7.27.2.3 [boolean MConfig::MdmeEnable](#)

Flag indicates whether Media identification masking is enabled or disabled

#### 7.27.2.4 [boolean MConfig::MemeEnable](#)

Flag indicates whether Message identification masking is enabled or disabled

The documentation for this struct was generated from the following file:

- [tini400\\_canbus.h](#)

## 7.28 netstat\_arp\_entry Struct Reference

```
#include <rom400_netstat.h>
```

### 7.28.1 Detailed Description

Structure for a single ARP entry. The `netstat_get_arp_table` function returns a pointer to a table that contains all system ARP entries. Each entry maps an Ethernet MAC address to an IPv4 address.

## Data Fields

- unsigned char [flags](#)  
*Flags: NETSTAT\_ARP\_USED, NETSTAT\_ARP\_REPLY\_PENDING or NETSTAT\_ARP\_STATIC.*
- unsigned char [ttl](#)  
*Time to live for this entry (in ticks).*
- unsigned char [mac](#) [6]  
*MAC address associated with this entry.*
- unsigned char [ip](#) [4]  
*IPv4 address for the MAC address.*

The documentation for this struct was generated from the following file:

- [rom400\\_netstat.h](#)

## 7.29 netstat\_tcp\_socket Struct Reference

```
#include <rom400_netstat.h>
```

### 7.29.1 Detailed Description

Structure for a TCP socket. The `netstat_get_tcp_socket` function returns a pointer to this structure for a given socket number (up to [NETSTAT\\_TCP\\_MAX\\_SOCKETS](#)).

## Data Fields

- unsigned char [flags](#)  
*Flags: NETSTAT\_TCP\_OUTPUT\_NEEDED\_MASK to NETSTAT\_TCP\_SEND\_FIN\_MASK.*
- unsigned char [state](#)  
*Socket state – see NETSTAT\_TCP\_STATE\_XXX (e.g. NETSTAT\_TCP\_STATE\_CLOSED).*
- unsigned char [server\\_sock](#)  
*Server socket number (only valid for server).*

- unsigned char `ack_timer`  
*Timer for delayed ACKs.*
- unsigned short `remote_port`  
*Remote port (if not a server socket).*
- unsigned char `remote_addr` [16]  
*Remote IP address (if not a server socket).*
- unsigned short `local_port`  
*Local port.*
- unsigned char `local_addr` [16]  
*Local IP address (may be the wildcard address 0).*
- unsigned long `sequence_num`  
*Current TCP sequence number.*
- unsigned long `ack_num`  
*Last ACK number.*
- unsigned short `input_retrieve_ptr`  
*Tail pointer to input queue.*
- unsigned short `input_store_ptr`  
*Head pointer to input queue.*
- unsigned char `input_buffer_hpp` [5]  
*Input queue.*
- unsigned short `output_retrieve_ptr`  
*Tail pointer to output queue.*
- unsigned short `output_store_ptr`  
*Head pointer to output queue.*
- unsigned char `output_buffer_hpp` [5]  
*Output queue.*
- unsigned short `receiver_win_size`  
*Receiver's TCP windows size.*

- unsigned short `sender_win_size`  
*Sender's TCP window size.*
- unsigned short `receiver_mss`  
*Maximum segment size of receiver.*
- unsigned short `sock`  
*Socket number.*
- unsigned long `last_ack_received`  
*Largest (usually last) ACK.*
- unsigned short `output_ack_ptr`  
*Pointer to last acknowledged byte.*
- unsigned char `reload_retry_min`  
*Lower bound on the retry timer reload.*
- unsigned char `retry_timer` [2]  
*Retry timer (one byte counter with overflow bit).*
- unsigned char `retry_flags`  
*(Reserved/unused)*
- unsigned char `retry_count`  
*Number of times the last segment has been retried.*
- unsigned char `retry_timer_reload`  
*Start value for the retry timer reload.*
- unsigned short `death_timer`  
*Time until a forced close of the connection.*
- unsigned char `options`  
*TCP option flags – see `NETSTAT_TCP_OPTION_XXX` (e.g. `NETSTAT_TCP_OPTION_NAGLE_ENABLED_MASK`).*
- unsigned char `unacked_segs`  
*Number of unacknowledged segments.*
- unsigned char `max_unacked_segs`  
*Maximum number of unacknowledged segments.*



- unsigned char [persist\\_timer](#)  
*TCP persist timer.*
- unsigned char [persist\\_timer\\_cap](#)  
*Current cap for TCP persist timer.*
- unsigned short [send\\_mss](#)  
*Maximum segment size for sending.*

The documentation for this struct was generated from the following file:

- [rom400\\_netstat.h](#)

## 7.30 netstat\_udp\_entry Struct Reference

```
#include <rom400_netstat.h>
```

### 7.30.1 Detailed Description

Structure for a single UDP port table entry. The `netstat_get_udp_table` function returns a pointer to a table that contains [NETSTAT\\_UDP\\_ENTRIES](#) of this structure.

#### Data Fields

- unsigned char [flags](#)  
*Flags: NETSTAT\_UDP\_USED.*
- unsigned short [port](#)  
*Port number for this entry.*
- unsigned char [queue\\_hpp](#) [5]  
*Incoming packet queue for this port.*
- unsigned char [reserved](#)  
*(Reserved)*

The documentation for this struct was generated from the following file:

- [rom400\\_netstat.h](#)

## 7.31 pingdata Struct Reference

```
#include <rom400_sock.h>
```

### 7.31.1 Detailed Description

The ping return data structure

#### Data Fields

- unsigned char [reserved](#) [3]  
*Reserved field.*
- unsigned char [ip\\_header](#) [20]  
*The IP header of the return packet.*
- unsigned char [icmp\\_header](#) [8]  
*icmp\_header - The ICMP header of the return packet*
- unsigned char [icmp\\_data](#) [32]  
*icmp\_data - The ICMP data portion of the return packet (should be 0x20,0x21,0x22,...,0x3f)*

The documentation for this struct was generated from the following file:

- [rom400\\_sock.h](#)

## 7.32 sockaddr Struct Reference

```
#include <rom400_sock.h>
```

### 7.32.1 Detailed Description

Structure for an IP address. For a normal, IPv4 (4 byte) address, set the address in `sin_addr[12,13,14,15]`, with the most significant byte at `sin_addr[12]`. Notice the 3 byte *reserved* to deal with the TNI native interface overhead.

## Data Fields

- unsigned char [reserved](#) [3]  
*Overhead for TNI native interface.*
- unsigned char [sin\\_addr](#) [16]  
*IP address. IPv4 address is in `sin_addr[12-15]` with MSB at `sin_addr[12]`.*
- unsigned short [sin\\_port](#)  
*16 bit port number for the socket.*
- unsigned char [sin\\_family](#)  
*Ignored by DS80C400 implementation.*

The documentation for this struct was generated from the following file:

- [rom400\\_sock.h](#)

## 7.33 sockaddr\_in Struct Reference

```
#include <rom400_sock.h>
```

### 7.33.1 Detailed Description

Alternate structure for an IP address. For a normal, IPv4 (4 byte) address, set the address in `sin_addr.s_addr`, and set `sin_zero` to all 0's. Notice the 3 byte *reserved* to deal with the TNI native interface overhead.

## Data Fields

- unsigned char [reserved](#) [3]  
*Overhead for TNI native interface.*
- unsigned char [sin\\_zero](#) [12]  
*Zeros in IP address due to IPv6 support.*
- [in\\_addr](#) [sin\\_addr](#)  
*IPv4 address structure.*
- unsigned short [sin\\_port](#)  
*16 bit port number for the socket.*

- unsigned char [sin\\_family](#)  
*Ignored by DS80C400 implementation.*

The documentation for this struct was generated from the following file:

- [rom400\\_sock.h](#)

## 7.34 TCB Struct Reference

```
#include <rom400_task.h>
```

### 7.34.1 Detailed Description

Task control buffer.

#### Data Fields

- unsigned char [Priority](#)  
*Priority of the task.*
- unsigned char [ID](#)  
*ID of the task.*
- [FARPTR Next](#)  
*Next task in the queue.*
- unsigned short [MemHandle](#)  
*KMalloc handle for this TCB.*
- unsigned char [Flags](#)  
*Flags for the task.*
- [TIME WakeupTime](#)  
*Time that the task is scheduled to wake from a sleep.*
- unsigned short [StateSize](#)  
*Size of the saved state for the task.*
- [FARPTR StatePtr](#)

*Pointer to the saved state for the task.*

The documentation for this struct was generated from the following file:

- [rom400\\_task.h](#)

## 7.35 TIME Struct Reference

```
#include <rom400_task.h>
```

### 7.35.1 Detailed Description

Structure to be used when handling the DS80C400's 5 byte time values.

See also:

[task\\_gettimemillis](#)

#### Data Fields

- unsigned char [msb](#)  
*Most significant byte of the time stamp. The Keil compiler does not have data types longer than 4 bytes.*
- unsigned long [millis](#)  
*The lower 4 bytes of a DS80C400 time stamp (in milliseconds). This will cover up to 49.7 days.*

The documentation for this struct was generated from the following file:

- [rom400\\_task.h](#)

## 7.36 tm Struct Reference

```
#include <tini400_time.h>
```

### 7.36.1 Detailed Description

Structure for calendar time. Note that the computation of these values depends on the time base year set by the [time\\_settimebase](#) function.

## Data Fields

- int [tm\\_sec](#)  
*Seconds after the minute (0..59).*
- int [tm\\_min](#)  
*Minutes after the hour (0..59).*
- int [tm\\_hour](#)  
*Hours since midnight (0..23).*
- int [tm\\_mday](#)  
*Day of the month (1..31).*
- int [tm\\_mon](#)  
*Months since January (0..11).*
- int [tm\\_year](#)  
*Year.*
- int [tm\\_wday](#)  
*Days since Sunday (0..6).*
- int [tm\\_yday](#)  
*Days since January 1 (0..365).*
- int [tm\\_isdst](#)  
*Daylight savings time flag, currently not supported.*

The documentation for this struct was generated from the following file:

- [tini400\\_time.h](#)

## 8 DS80C400CLibraries File Documentation

### 8.1 dirent.h File Reference

#### 8.1.1 Detailed Description

Functions for directory listing.

This library contains functions that allow applications to list the contents of a directory. To use this library, the file system must also be installed and initialized.

Note that not all of the traditional **dirent** functions are implemented.

For detailed information on the DS80C400 please see the [High-Speed Microcontroller User's Guide: DS80C400 Supplement](#).

#### Warning:

Some functions in this library are **NOT** multi-process safe—that is, if you call the same method from two different processes at the same time, the parameters to the function may be destroyed, yielding unpredictable results. Consult each individual function's documentation for details on which functions are multi-process safe.

#### Data Structures

- struct [dirent](#)

#### Defines

- #define [ROM400\\_DIRENT\\_VERSION](#) 2

#### Typedefs

- typedef unsigned char \* [DIR](#)

#### Functions

- int [closedir](#) ([DIR](#) \*dir)  
*Close a directory stream.*
- [DIR](#) \* [opendir](#) (const char \*name)  
*Open a directory stream.*
- [dirent](#) \* [readdir](#) ([DIR](#) \*dir)  
*Read a directory entry from a directory stream.*
- void [rewinddir](#) ([DIR](#) \*dir)  
*Resets the directory stream.*
- void [seekdir](#) ([DIR](#) \*dir, long int ptr)  
*Sets the directory stream location.*

- long int [telldir](#) ([DIR](#) \*dir)  
*Returns the current location in the directory stream.*
- unsigned int [dirent\\_version](#) (void)  
*Returns the version number of this DIRENT library.*

## 8.1.2 Define Documentation

### 8.1.2.1 #define ROM400\_DIRENT\_VERSION 2

Version number associated with this header file. Should be the same as the version number returned by the [dirent\\_version](#) function.

See also:

[dirent\\_version](#)

## 8.1.3 Typedef Documentation

### 8.1.3.1 typedef unsigned char\* [DIR](#)

Type definition for a directory structure. This object must not be altered by the application during use. Make sure to call the [closedir](#) function when finished with any DIR object.

## 8.1.4 Function Documentation

### 8.1.4.1 int [closedir](#) ([DIR](#) \* dir)

Close a directory stream.

Closes the directory stream **dir**, and frees the resources allocated to it.

**Parameters:**

*dir* Directory resource to free.

**Returns:**

0 on success, non-zero if the directory could not be closed.

See also:

[opendir](#)



#### 8.1.4.2 unsigned int dirent\_version (void)

Returns the version number of this DIRENT library.

**Returns:**

Version number of this DIRENT library.

#### 8.1.4.3 DIR\* opendir (const char \* name)

Open a directory stream.

Opens a directory stream for the directory **name**. The argument **name** should not have leading or trailing slashes. To open the root directory, use the empty string (`opendir("")`).

**Parameters:**

*name* Name of the directory to open

**Returns:**

Pointer to a directory stream object, or NULL if the directory could not be found.

**See also:**

[closedir](#)

#### 8.1.4.4 struct dirent\* readdir (DIR \* dir)

Read a directory entry from a directory stream.

Reads the current directory entry from the directory stream **dir**. This function also increments the internal stream counter, so the next call to *readdir* will read the next directory entry.

Before using the returned file name, call *file\_exists* to make sure the file still exists. It could have been deleted between the time the directory stream was opened and now, which would yield an invalid result.

**Parameters:**

*dir* Directory stream to read an entry from.

**Returns:**

Pointer to a directory entry, or NULL if the end of the directory stream has been reached.

**See also:**

[rewinddir](#)

[seekdir](#)

[telldir](#)

#### 8.1.4.5 void rewinddir (**DIR** \* *dir*)

Resets the directory stream.

Resets the directory stream to the beginning, so the first directory entry is read again.

##### Parameters:

*dir* Directory stream to be reset.

##### See also:

[seekdir](#)

[telldir](#)

#### 8.1.4.6 void seekdir (**DIR** \* *dir*, long int *ptr*)

Sets the directory stream location.

Sets the current 'pointer' into the directory stream to the value *ptr*. Internally, the directory stream is simply an array of file pointers. This function sets the current index into that array. If *ptr* is beyond the bounds of the array, the next call to [readdir](#) will return NULL;

##### Parameters:

*dir* Directory stream to set location

*ptr* Location to point to in stream

##### See also:

[readdir](#)

[telldir](#)

#### 8.1.4.7 long int telldir (**DIR** \* *dir*)

Returns the current location in the directory stream.

Returns the current location in the directory stream. Internally, the directory stream is simply an array of file pointers. This function returns the current index into that array.

##### Parameters:

*dir* Directory stream to get location

##### See also:

[readdir](#)

[seekdir](#)

## 8.2 rom400\_dhcp.h File Reference

### 8.2.1 Detailed Description

DHCP functions in the DS80C400 ROM.

This library contains functions that allow the DS80C400 to lease addresses from a DHCP server. Only Ipv4 addresses can be leased using DHCP. Ipv6 addresses are automatically configured. Once the DHCP client negotiates a lease on an address, functions from the socket libraries ([rom400\\_sock.h](#)) can be used to get the current IP address and communicate with other devices.

For detailed information on the DS80C400 please see the [High-Speed Microcontroller User's Guide: DS80C400 Supplement](#).

#### Warning:

The functions in this library are multi-process safe—that is, if you call the same method from two different processes at the same time, the parameters to the function will not be destroyed. However, it is recommended that only one process manage the DHCP client as it is a system-wide resource.

Note that the DHCP client requires a clock that is no slower than 133% of real time. Otherwise, DHCP lease renewal might be problematic. Avoid blocking interrupts for extended periods of time and use *init\_setfrequency* to set the correct crystal frequency.

#### Defines

- #define [ROM400\\_DHCP\\_VERSION](#) 12
- #define [DHCP\\_STATUS\\_INIT](#) 0
- #define [DHCP\\_STATUS\\_SELECTING](#) 1
- #define [DHCP\\_STATUS\\_REQUESTING](#) 2
- #define [DHCP\\_STATUS\\_INITREBOOT](#) 3
- #define [DHCP\\_STATUS\\_REBOOTING](#) 4
- #define [DHCP\\_STATUS\\_BOUND](#) 5
- #define [DHCP\\_STATUS\\_RENEWING](#) 6
- #define [DHCP\\_STATUS\\_REBINDING](#) 7
- #define [DHCP\\_MSG\\_DHCPDISCOVER](#) 1
- #define [DHCP\\_MSG\\_DHCPOFFER](#) 2
- #define [DHCP\\_MSG\\_DHCPREQUEST](#) 3
- #define [DHCP\\_MSG\\_DHCPDECLINE](#) 4
- #define [DHCP\\_MSG\\_DHCPACK](#) 5
- #define [DHCP\\_MSG\\_DHCPNAK](#) 6
- #define [DHCP\\_MSG\\_DHCPRELEASE](#) 7
- #define [DHCP\\_MSG\\_DHCPIFORM](#) 8

## Functions

- unsigned char **dhcp\_init** (void)  
*Initializes the DHCP client.*
- void **dhcp\_setrequestip** (struct **sockaddr** \*address, int len)  
*Sets the requested IP (and INITREBOOT state).*
- unsigned int **dhcp\_status** (void)  
*Gets the status of the DHCP client.*
- void **dhcp\_stop** (int releaseip)  
*Disabled the DHCP client.*
- void **dhcp\_registernotify** (void(\*functionptr)(unsigned int newstate, unsigned char far \*packet))  
*Register a function to be notified when DHCP acquires or loses an IP.*
- void **dhcp\_registerparseoption** (void(\*functionptr)(unsigned char far \*option))  
*Register a function to be called when an unknown or unhandled DHCP option is encountered.*
- void **dhcp\_registerbuildpacket** (unsigned char(\*functionptr)(unsigned char far \*option, unsigned char msgtype))  
*Register a function to be called when a DHCP packet is about to be sent.*
- unsigned int **dhcp\_version** (void)  
*Returns the version number of this DHCP library.*
- void **dhcp\_getserverip** (struct **sockaddr** \*address, int len)  
*Returns the IP address of the DHCP server.*
- void **dhcp\_getprimarydns** (struct **sockaddr** \*address)  
*Returns the IP address of the primary DNS server.*
- void **dhcp\_getsecondarydns** (struct **sockaddr** \*address)  
*Returns the IP address of the secondary DNS server.*
- unsigned int **dhcp\_gettaskid** ()  
*Returns task ID of the DHCP process.*
- int **dhcp\_getsocket** ()  
*Returns socket handle of the socket used in the DHCP process.*

## 8.2.2 Define Documentation

### 8.2.2.1 **#define DHCP\_MSG\_DHCPACK 5**

DHCP message type ACK

### 8.2.2.2 **#define DHCP\_MSG\_DHCPDECLINE 4**

DHCP message type DECLINE

### 8.2.2.3 **#define DHCP\_MSG\_DHCPDISCOVER 1**

DHCP message type DISCOVER

### 8.2.2.4 **#define DHCP\_MSG\_DHCPIFORM 8**

DHCP message type INFORM

### 8.2.2.5 **#define DHCP\_MSG\_DHCNNAK 6**

DHCP message type NAK

### 8.2.2.6 **#define DHCP\_MSG\_DHCPOFFER 2**

DHCP message type OFFER

### 8.2.2.7 **#define DHCP\_MSG\_DHCPRERELEASE 7**

DHCP message type RELEASE

### 8.2.2.8 **#define DHCP\_MSG\_DHCPREREQUEST 3**

DHCP message type REQUEST

### 8.2.2.9 **#define DHCP\_STATUS\_BOUND 5**

DHCP status code returned by [dhcp\\_status](#). The DHCP client is in the **BOUND** state: it has been configured with a valid address.

See also:

[dhcp\\_status](#)

#### 8.2.2.10 **#define DHCP\_STATUS\_INIT 0**

DHCP status code returned by [dhcp\\_status](#). The DHCP client is in the **INIT** state: it has not yet sent a DHCP\_DISCOVER message.

See also:

[dhcp\\_status](#)

#### 8.2.2.11 **#define DHCP\_STATUS\_INITREBOOT 3**

DHCP status code returned by [dhcp\\_status](#). The DHCP client is in the **INITREBOOT** state: it has rebooted, and is trying to acquire its old address.

See also:

[dhcp\\_status](#)

#### 8.2.2.12 **#define DHCP\_STATUS\_REBINDING 7**

DHCP status code returned by [dhcp\\_status](#). The DHCP client is in the **REBINDING** state: it is attempting to get a new lease after its current lease expired.

See also:

[dhcp\\_status](#)

#### 8.2.2.13 **#define DHCP\_STATUS\_REBOOTING 4**

DHCP status code returned by [dhcp\\_status](#). The DHCP client is in the **REBOOTING** state: after a reboot, it is waiting for permission to use its old address.

See also:

[dhcp\\_status](#)

#### 8.2.2.14 **#define DHCP\_STATUS\_RENEWING 6**

DHCP status code returned by [dhcp\\_status](#). The DHCP client is in the **RENEWING** state: it is attempting to extend the current, valid lease of its address.

See also:

[dhcp\\_status](#)

#### 8.2.2.15 **#define DHCP\_STATUS\_REQUESTING 2**

DHCP status code returned by [dhcp\\_status](#). The DHCP client is in the **REQUESTING** state: it has requested a DHCP address, and is awaiting a reply.

See also:

[dhcp\\_status](#)

#### 8.2.2.16 **#define DHCP\_STATUS\_SELECTING 1**

DHCP status code returned by [dhcp\\_status](#). The DHCP client is in the **SELECTING** state: it is collecting DHCP offers.

See also:

[dhcp\\_status](#)

#### 8.2.2.17 **#define ROM400\_DHCP\_VERSION 12**

Version number associated with this header file. Should be the same as the version number returned by the [dhcp\\_version](#) function.

See also:

[dhcp\\_version](#)

### 8.2.3 Function Documentation

#### 8.2.3.1 **void dhcp\_getprimarydns (struct [sockaddr](#) \* *address*)**

Returns the IP address of the primary DNS server.

Returns the IP address of the primary DNS server. The DNS server can be set by an option received from a DHCP response, or by setting it manually from the DNS library function [dns\\_setprimary](#). Note that this DNS server information entry is cleared out on initialization.

**Parameters:**

*address* will fill in the primary DNS server IP address

See also:

[dhcp\\_getsecondarydns](#)

[dns\\_setprimary](#)

[dns\\_getprimary](#)

#### 8.2.3.2 void dhcp\_getsecondarydns (struct sockaddr \* address)

Returns the IP address of the secondary DNS server.

Returns the IP address of the primary DNS server. The DNS server can be set by an option received from a DHCP response, or by setting it manually from the DNS library function [dns\\_setprimary](#). Note that this DNS server information entry is cleared out on initialization.

##### Parameters:

*address* will fill in the secondary DNS server IP address

##### See also:

[dhcp\\_getprimarydns](#)  
[dns\\_setsecondary](#)  
[dns\\_getsecondary](#)

#### 8.2.3.3 void dhcp\_getserverip (struct sockaddr \* address, int len)

Returns the IP address of the DHCP server.

##### Parameters:

*address* will fill in the DHCP server IP address

*len* length of the address structure (ignored)

#### 8.2.3.4 int dhcp\_getsocket ()

Returns socket handle of the socket used in the DHCP process.

Returns the socket handle of the socket used in the DHCP process. If the DHCP process is not running, the return value is invalid.

##### Returns:

Socket handle of DHCP socket

##### See also:

[dhcp\\_gettaskid](#)

#### 8.2.3.5 unsigned int dhcp\_gettaskid ()

Returns task ID of the DHCP process.

Returns the task ID of the DHCP process. If the DHCP process has not been initialized (but the [init\\_rom](#) function has been called), this function returns 0.

The value returned by this function is suitable to use with the task library—for instance, to alter the priority of the DHCP task.



**Returns:**

Task ID of the DHCP process.

**See also:**

[dhcp\\_init](#)  
[dhcp\\_getsocket](#)

**8.2.3.6 unsigned char dhcp\_init (void)**

Initializes the DHCP client.

Starts a DHCP Client task and returns to the caller. DHCP is implemented for IPv4 only. The IPv6 portion of the network stack uses neighbor discovery. To read the address that the DHCP client has leased, use the socket library function [getnetworkparams](#). DHCP tries to request a previously leased IP (INITREBOOT state), see [dhcp\\_setrequestip](#). This can cause dhcp\_init to take a long time. Set a zero IP to force the INIT state.

**Returns:**

0 for success, non-zero for failure.

**See also:**

[dhcp\\_stop](#)  
[getnetworkparams](#) [in the [socket](#) library]  
[dhcp\\_setrequestip](#)

**8.2.3.7 void dhcp\_registerbuildpacket (unsigned char\*)(unsigned char far \*option, unsigned char msgtype) functionptr)**

Register a function to be called when a DHCP packet is about to be sent.

The function passed as *functionptr* will be called when the DHCP client is about to send a DHCP packet. The function pointed to by *functionptr* should take two arguments (a pointer and a byte) and return a byte. Whenever the function at *functionptr* is called, the pointer will be pointing to the first byte after the default options. The user can fill in additional DHCP options, e.g. 0x0c,0x04,'T','I','N','I' would be a DHCP hostname option. The msgtype argument contains the current DHCP message type (DHCP\_MSG\_DISCOVER or DHCP\_MSG\_REQUEST). The return value is the number of bytes added to the DHCP options, 6 in the hostname example above.

The function does not need to save/restore any registers.

**Parameters:**

*functionptr* Pointer to a function with the signature unsigned char fn(unsigned char far\* option, unsigned char msgtype)

### 8.2.3.8 void dhcp\_registernotify (void(\*) (unsigned int newstate, unsigned char far \*packet) *functionptr*)

Register a function to be notified when DHCP acquires or loses an IP.

The function passed as *functionptr* will be called when the DHCP client acquires or loses an IP.

#### Parameters:

*functionptr* Pointer to a function with the signature void fn(unsigned int newstate, unsigned char far \*packet). The function will be provided with the new DHCP state and a pointer to the last DHCP packet received (the packet pointer points to the beginning of the BOOTP data structure).

#### See also:

[dhcp\\_status](#)

### 8.2.3.9 void dhcp\_registerparseoption (void(\*) (unsigned char far \*option) *functionptr*)

Register a function to be called when an unknown or unhandled DHCP option is encountered.

The function passed as *functionptr* will be called when the DHCP client is parsing an unknown DHCP option. The function pointed to by *functionptr* should take one argument (a pointer) and return void. Whenever the function at *functionptr* is called, the argument will be pointing to the current unhandled or vendor specific DHCP option.

The function does not need to save/restore any registers.

#### Parameters:

*functionptr* Pointer to a function with the signature void fn(unsigned char far\* option).

### 8.2.3.10 void dhcp\_setrequestip (struct [sockaddr](#) \* *address*, int *len*)

Sets the requested IP (and INITREBOOT state).

When [dhcp\\_init](#) is called and the requested IP is not zero, the DHCP state machine will start in INITREBOOT state and try to obtain a previously leased IP (if the DHCP server doesn't answer or the request is denied, it will ignore the requested IP). Setting a requested IP can delay the IP lease if the DHCP server doesn't respond (e.g. because it didn't retain the client record); set to zero to force the state machine to INIT state.

#### Parameters:

*address* fill in the requested IP address

*len* length of the address structure (ignored)

See also:

[dhcp\\_init](#)

#### 8.2.3.11 unsigned int dhcp\_status (void)

Gets the status of the DHCP client.

Returns the current state of the DHCP Client. DHCP Clients that have leased a valid address should return **DHCP\_STATUS\_BOUND**.

**Returns:**

Status of the DHCP client.

#### 8.2.3.12 void dhcp\_stop (int releaseip)

Disabled the DHCP client.

Kills the DHCP client task. Use [dhcp\\_init](#) to restart the DHCP client. If the "releaseip" argument is non-zero, a DHCP release message is sent. Some DHCP servers don't retain client records, and releasing the IP will make it impossible/slower to get the same IP after a reboot.

**Parameters:**

*releaseip* – 0: don't send a DHCP release message

See also:

[dhcp\\_init](#)

#### 8.2.3.13 unsigned int dhcp\_version (void)

Returns the version number of this DHCP library.

**Returns:**

Version number of this DHCP library.

### 8.3 rom400\_err.h File Reference

#### 8.3.1 Detailed Description

Error codes used by functions in the DS80C400 ROM.

This file contains error codes that might be returned by functions that call into the ROM.

For detailed information on the DS80C400 please see the [High-Speed Microcontroller User's Guide: DS80C400 Supplement](#).

## Defines

- #define ROM400\_ERR\_VERSION 1
- #define ROM400\_IOEXCEPTION 0x0B
- #define ROM400\_INTERRUPTEDIOEXCEPTION 0x36
- #define ROM400\_ARRAYINDEXOUTOFBOUNDSEXCEPTION 0x0D
- #define ROM400\_INTERNALERROR 0x2C
- #define ROM400\_NULLPOINTEREXCEPTION 0x08
- #define ROM400\_OUTOFMEMORYERROR 0x30
- #define ROM400\_BINDEXCEPTION 0x35
- #define ROM400\_CONNECTEXCEPTION 0x46
- #define ROM400\_SOCKETEXCEPTION 0x32

### 8.3.2 Define Documentation

#### 8.3.2.1 #define ROM400\_ARRAYINDEXOUTOFBOUNDSEXCEPTION 0x0D

Indicates that the index or offset to an array access was out of bounds.

#### 8.3.2.2 #define ROM400\_BINDEXCEPTION 0x35

Indicates that application cannot bind to address (interface unavailable, not a server socket or socket not bound).

#### 8.3.2.3 #define ROM400\_CONNECTEXCEPTION 0x46

Indicates that an error occurred trying to connect to a remote port. The connection was probably refused remotely.

#### 8.3.2.4 #define ROM400\_ERR\_VERSION 1

Version number associated with this header file.

#### 8.3.2.5 #define ROM400\_INTERNALERROR 0x2C

Indicates a problem with the network queue.

#### **8.3.2.6 #define ROM400\_INTERRUPTEDIOEXCEPTION 0x36**

Indicates that a sleep or wait was interrupted.

#### **8.3.2.7 #define ROM400\_IOEXCEPTION 0x0B**

General error that is returned when a resource (memory, port) is not available or an internal data structure (table) cannot hold more elements or is corrupted.

#### **8.3.2.8 #define ROM400\_NULLPOINTEREXCEPTION 0x08**

Indicates that a pointer was not able to be dereferenced.

#### **8.3.2.9 #define ROM400\_OUTOFMEMORYERROR 0x30**

Indicates that the system has run out of kernel or regular memory to allocate.

#### **8.3.2.10 #define ROM400\_SOCKETEXCEPTION 0x32**

Indicates that a socket is not available (port in use or socket closed).

## **8.4 rom400\_flash.h File Reference**

### **8.4.1 Detailed Description**

Flash programming functions for the TINIm400 module.

This library contains functions that allow applications to access the ROM's flash erasing and programming algorithms. Any flash that is compatible with the DS80C400 boot loader's functions will be compatible with this library.

For detailed information on the DS80C400 please see the [High-Speed Microcontroller User's Guide: DS80C400 Supplement](#).

The functions in this library are multi-process safe—that is, if you call the same method from two different processes at the same time, the parameters to the function will not be destroyed. However, multiple processes should not be performing flash altering operations without some kind of synchronization control.

### **Defines**

- #define [ROM400\\_FLASH\\_VERSION](#) 2

## Functions

- unsigned char [flash\\_eraseblock](#) (unsigned char blocknum)  
*Erase a flash block.*
- unsigned char [flash\\_programbyte](#) (void \*location, unsigned char b)  
*Program a byte of flash.*
- unsigned int [flash\\_version](#) (void)  
*Returns the version number of this flash library.*

### 8.4.2 Define Documentation

#### 8.4.2.1 #define ROM400\_FLASH\_VERSION 2

Version number associated with this header file. Should be the same as the version number returned by the [flash\\_version](#) function.

See also:

[flash\\_version](#)

### 8.4.3 Function Documentation

#### 8.4.3.1 unsigned char flash\_eraseblock (unsigned char blocknum)

Erase a flash block.

Erases the block of flash that begins at address *blocknum*:00:00. This operation checks to see if the block is RAM or is the ROM (*blocknum* equals FF), in which case the operation fails.

**Parameters:**

*blocknum* bank/block number of flash to erase

**Returns:**

0 if the erase was successful, 1 if the erase could not be performed.

#### 8.4.3.2 unsigned char flash\_programbyte (void \* location, unsigned char b)

Program a byte of flash.

Programs the byte *b* to the address *location*. If the location is unprogrammable (too many zero bits have already been set) the operation fails.

**Parameters:**

*location* The address to write the value *b* to

*b* The value to be programmed

**Returns:**

0 if the program is successful, 1 if the operation could not be performed.

**8.4.3.3 unsigned int flash\_version (void)**

Returns the version number of this flash library.

**Returns:**

Version number of this flash library.

**8.5 rom400\_http.h File Reference****8.5.1 Detailed Description**

Http Server functions in the DS80C400 ROM.

This library contains functions for implementing http server in DS80C400 ROM

For detailed information on the DS80C400 please see the [High-Speed Microcontroller User's Guide: DS80C400 Supplement](#).

**Warning:**

Some functions in this library are **NOT** multi-process safe—that is, if you call the same method from two different processes at the same time, the parameters to the function may be destroyed, yielding unpredictable results. Consult each individual function's documentation for details on which functions are multi-process safe.

**Data Structures**

- struct [\\_http\\_variable](#)
- struct [\\_http\\_request](#)
- struct [\\_http\\_response](#)
- struct [\\_http\\_session](#)

**Defines**

- #define [HTTP\\_VERSION](#) 8
- #define [HTTP\\_INSUFFICIENT\\_MEMORY](#) -1
- #define [HTTP\\_LOGFILE\\_ERROR](#) -2

- #define [HTTP\\_SOCKET\\_ERROR](#) -3
- #define [HTTP\\_REQUEST\\_NOT\\_PROCESSED](#) -4
- #define [HTTP\\_DENY\\_CONNECTION](#) -5
- #define [HTTP\\_TASK\\_ERROR](#) -6
- #define [HTTP\\_SERVER\\_ALREADY\\_RUNNING](#) -7
- #define [HTTP\\_NOSERVERTASK](#) -8
- #define [HTTP\\_STATUS\\_SUCCESS](#) 0
- #define [HTTP\\_DISABLE\\_LOG](#) 0
- #define [HTTP\\_ENABLE\\_LOG](#) 1
- #define [HTTP\\_ENABLE\\_VARIABLE\\_PARSING](#) 1
- #define [HTTP\\_DISABLE\\_VARIABLE\\_PARSING](#) 0
- #define [HTTP\\_ENABLE\\_MESSAGEBODY\\_READING](#) 2
- #define [HTTP\\_DISABLE\\_MESSAGEBODY\\_READING](#) 0
- #define [HTTP\\_STOP\\_SERVERTASK](#) 0
- #define [HTTP\\_RUN\\_SERVERTASK](#) 1
- #define [DEFAULT\\_BUF\\_SIZE](#) 400
- #define [DEFAULT\\_MAX\\_PENDING\\_CONNECTIONS](#) 5
- #define [HTTP\\_DEFAULT\\_PORT](#) 80
- #define [HTTP\\_MAX\\_URL](#) 400
- #define [HTTP\\_MAX\\_BUFSIZE](#) 400
- #define [HTTP\\_GET\\_METHOD](#) 1
- #define [HTTP\\_POST\\_METHOD](#) 2
- #define [HTTP\\_HEAD\\_METHOD](#) 3

## Typedefs

- typedef [\\_http\\_variable](#) [http\\_variable](#)
- typedef [\\_http\\_request](#) [http\\_request](#)
- typedef [\\_http\\_response](#) [http\\_response](#)
- typedef [\\_http\\_session](#) [http\\_session](#)

## Functions

- int [http\\_init](#) (struct [sockaddr](#) server\_addr)  
*Initializes http server library.*
- int [http\\_setrootdir](#) (char \*rootdir)  
*Sets http root directory.*
- char \* [http\\_getrootdir](#) (void)  
*Returns http root directory.*



- int [http\\_setindexpage](#) (char \*index)  
*Sets index page name.*
- char \* [http\\_getindexpage](#) (void)  
*Returns current index page.*
- void [http\\_setportnumber](#) (int portnumber)  
*Sets HTTP Server Port number.*
- int [http\\_getportnumber](#) (void)  
*Returns current http server port number.*
- void [http\\_setheaderbufsize](#) (int buffersize)  
*Sets header buffer size.*
- int [http\\_getheaderbufsize](#) (void)  
*Returns header buffer size.*
- void [http\\_set\\_req\\_linesize](#) (int buffersize)  
*Sets the maximum size value of each line of http request.*
- int [http\\_get\\_req\\_linesize](#) (void)  
*Returns the maximum size value of each line of http request.*
- void [http\\_set\\_req\\_process\\_options](#) (char flag)  
*Sets Http request Processing options configuration flag value.*
- char [http\\_get\\_req\\_process\\_options](#) (void)  
*Returns Http request Processing options configuration flag value.*
- int [http\\_getlogging](#) (void)  
*Returns logging status.*
- int [http\\_setlogging](#) (char logstatus)  
*Sets logging status.*
- char \* [http\\_getlogfilename](#) (void)  
*Returns log file name.*
- int [http\\_setlogfilename](#) (char \*logfilename)  
*Sets log file name.*

- void [http\\_setmaxconnections](#) (int max\_connection)  
*Sets maximum number of pending connections value.*
- int [http\\_getmaxconnections](#) (void)  
*Returns the current value for maximum number of pending connections.*
- void [http\\_setclientsocktimeout](#) (long milli\_sec)  
*Sets http client socket timeout value.*
- long [http\\_getclientsocktimeout](#) (void)  
*Returns the current value for http client socket timeout.*
- void [http\\_reg\\_req\\_callback](#) (int(\*func)())  
*Registers callback function to process http request.*
- void [http\\_reg\\_acl\\_callback](#) (int(\*func)())  
*Registers access control callback function.*
- void [http\\_sendheaders](#) ([http\\_session](#) \*https)  
*Sends http response headers to client.*
- int [http\\_start\\_server](#) (void)  
*Starts http server.*
- int [http\\_kill\\_server](#) (void)  
*Terminates http server.*
- void [http\\_decode\\_urlencodeddata](#) (char \*pathname)  
*Decodes the url path which was encoded in "application/x-www-form-urlencoded" format.*
- char [http\\_hex\\_from\\_ascii](#) (char c)  
*Returns hexadecimal value for input ascii digit.*
- int [http\\_change\\_server\\_state](#) (char server\_state)  
*Changes Http Server state.*

## 8.5.2 Define Documentation

### 8.5.2.1 #define DEFAULT\_BUF\_SIZE 400

Define for default header buffer size

**See also:**

[http\\_getheaderbufsize](#)

[http\\_setheaderbufsize](#)

#### **8.5.2.2 #define DEFAULT\_MAX\_PENDING\_CONNECTIONS 5**

Define for default maximum pending connections allowed

**See also:**

[http\\_setmaxconnections](#)

[http\\_getmaxconnections](#)

#### **8.5.2.3 #define HTTP\_DEFAULT\_PORT 80**

Define for default http port number

**See also:**

[http\\_getportnumber](#)

[http\\_setportnumber](#)

#### **8.5.2.4 #define HTTP\_DENY\_CONNECTION -5**

Error value to indicate that http client connection is denied by access control callback function

**See also:**

[http\\_reg\\_acl\\_callback](#)

#### **8.5.2.5 #define HTTP\_DISABLE\_LOG 0**

Define for disabling logging activity

**See also:**

[http\\_setlogging](#)

[http\\_getlogging](#)

#### **8.5.2.6 #define HTTP\_DISABLE\_MESSAGEBODY\_READING 0**

Define for disabling message body reading status flag

**See also:**

[http\\_set\\_req\\_process\\_options](#)

[http\\_get\\_req\\_process\\_options](#)

#### **8.5.2.7 #define HTTP\_DISABLE\_VARIABLE\_PARSING 0**

Define for disabling variable parsing status flag

See also:

[http\\_set\\_req\\_process\\_options](#)

[http\\_get\\_req\\_process\\_options](#)

#### **8.5.2.8 #define HTTP\_ENABLE\_LOG 1**

Define for enabling logging activity

See also:

[http\\_setlogging](#)

[http\\_getlogging](#)

#### **8.5.2.9 #define HTTP\_ENABLE\_MESSAGEBODY\_READING 2**

Define for enabling message body reading status flag

See also:

[http\\_set\\_req\\_process\\_options](#)

[http\\_get\\_req\\_process\\_options](#)

#### **8.5.2.10 #define HTTP\_ENABLE\_VARIABLE\_PARSING 1**

Define for enabling variable parsing status flag

See also:

[http\\_set\\_req\\_process\\_options](#)

[http\\_get\\_req\\_process\\_options](#)

#### **8.5.2.11 #define HTTP\_GET\_METHOD 1**

Get request method type

See also:

[http\\_request](#)

[http\\_session](#)

#### **8.5.2.12 #define HTTP\_HEAD\_METHOD 3**

Head request method type

See also:

[http\\_request](#)

[http\\_session](#)

#### **8.5.2.13 #define HTTP\_INSUFFICIENT\_MEMORY -1**

Insufficient memory error value

See also:

[http\\_setrootdir](#)

[http\\_setindexpage](#)

[http\\_setlogfilename](#)

#### **8.5.2.14 #define HTTP\_LOGFILE\_ERROR -2**

Error opening log file

See also:

[http\\_setlogging](#)

#### **8.5.2.15 #define HTTP\_MAX\_BUFSIZE 400**

Define for maximum buffer size

See also:

[http\\_request](#)

[http\\_session](#)

#### **8.5.2.16 #define HTTP\_MAX\_URL 400**

Define for maximum url path name size

See also:

[http\\_request](#)

[http\\_session](#)

#### **8.5.2.17 #define HTTP\_NOSERVERTASK -8**

Error value to indicate that http server task is not running

**See also:**

[http\\_start\\_server](#)  
[http\\_kill\\_server](#)  
[http\\_change\\_server\\_state](#)

#### **8.5.2.18 #define HTTP\_POST\_METHOD 2**

Post request method type

**See also:**

[http\\_request](#)  
[http\\_session](#)

#### **8.5.2.19 #define HTTP\_REQUEST\_NOT\_PROCESSED -4**

Error value to indicate that http request was not processed by callback function

**See also:**

[http\\_reg\\_req\\_callback](#)

#### **8.5.2.20 #define HTTP\_RUN\_SERVERTASK 1**

Define for running http server task

**See also:**

[http\\_change\\_server\\_state](#)

#### **8.5.2.21 #define HTTP\_SERVER\_ALREADY\_RUNNING -7**

Error value to indicate that server is already running

**See also:**

[http\\_start\\_server](#)

#### **8.5.2.22 #define HTTP\_SOCKET\_ERROR -3**

Socket error value

**See also:**

[http\\_start\\_server](#)

#### 8.5.2.23 **#define HTTP\_STATUS\_SUCCESS 0**

Http Status Success value, this value is returned when operation is completed successfully

See also:

[http\\_setrootdir](#)  
[http\\_setindexpage](#)  
[http\\_setlogfilename](#)  
[http\\_start\\_server](#)  
[http\\_kill\\_server](#)

#### 8.5.2.24 **#define HTTP\_STOP\_SERVERTASK 0**

Define for stopping http server task

See also:

[http\\_change\\_server\\_state](#)

#### 8.5.2.25 **#define HTTP\_TASK\_ERROR -6**

New task creation error

See also:

[http\\_start\\_server](#)

#### 8.5.2.26 **#define HTTP\_VERSION 8**

Version number associated with this header file. Should be the same as the version number returned by the [http\\_version](#) function.

See also:

[http\\_version](#)

### 8.5.3 Typedef Documentation

#### 8.5.3.1 **typedef struct [\\_http\\_request](#) http\_request**

Structure for http request

#### 8.5.3.2 **typedef struct [\\_http\\_response](#) http\_response**

Structure for http response

### 8.5.3.3 typedef struct [\\_http\\_session](#) [http\\_session](#)

Structure for http session

### 8.5.3.4 typedef struct [\\_http\\_variable](#) [http\\_variable](#)

Structure for http variable names and values

## 8.5.4 Function Documentation

### 8.5.4.1 int [http\\_change\\_server\\_state](#) (char *server\_state*)

Changes Http Server state.

This function sets http server state

This function is safe to be called from multiple processes at the same time.

#### Parameters:

*server\_state* Server state. Should be either [HTTP\\_RUN\\_SERVERTASK](#) or [HTTP\\_STOP\\_SERVERTASK](#)

#### Returns:

[HTTP\\_STATUS\\_SUCCESS](#) or [HTTP\\_NOSERVERTASK](#)

### 8.5.4.2 void [http\\_decode\\_urlencodeddata](#) (char \* *pathname*)

Decodes the url path which was encoded in "application/x-www-form-urlencoded" format.

This function decodes the url path which was encoded in "application/x-www-form-urlencoded" format.

This function is safe to be called from multiple processes at the same time.

#### Parameters:

*pathname* pointer to url path name

### 8.5.4.3 int [http\\_get\\_req\\_linesize](#) (void)

Returns the maximum size value of each line of http request.

This function returns the maximum size value of each line of http request

This function is safe to be called from multiple processes at the same time.

#### Returns:

bufferize the maximum size value of each http request line



#### **8.5.4.4 char http\_get\_req\_process\_options (void)**

Returns Http request Processing options configuration flag value.

This function is safe to be called from multiple processes at the same time.

##### **Returns:**

The variable parsing status flag value

#### **8.5.4.5 long http\_getclientsocktimeout (void)**

Returns the current value for http client socket timeout.

This function returns the current value for http client socket timeout

This function is safe to be called from multiple processes at the same time.

##### **Returns:**

The current value for http client socket timeout

#### **8.5.4.6 int http\_getheaderbufsize (void)**

Returns header buffer size.

This function returns buffer size value for http request and response headers.

This function is safe to be called from multiple processes at the same time.

##### **Returns:**

header buffer size

#### **8.5.4.7 char\* http\_getindexpage (void)**

Returns current index page.

This function returns the current index page name.

This function is safe to be called from multiple processes at the same time.

##### **Returns:**

The starting address of index page name. NULL will be returned if there is no index page set.

#### **8.5.4.8 char\* http\_getlogfilename (void)**

Returns log file name.

This function returns the log file name.

This function is safe to be called from multiple processes at the same time.

#### **Returns:**

The address of log file name. NULL will be returned if there is no log file set.

#### **8.5.4.9 int http\_getlogging (void)**

Returns logging status.

This function returns the current logging status

This function is safe to be called from multiple processes at the same time.

#### **Returns:**

*HTTP\_DISABLE\_LOG* or *HTTP\_ENABLE\_LOG*

#### **8.5.4.10 int http\_getmaxconnections (void)**

Returns the current value for maximum number of pending connections.

This function returns the current value for maximum number of pending connections

This function is safe to be called from multiple processes at the same time.

#### **Returns:**

The current value for maximum number of pending connections.

#### **8.5.4.11 int http\_getportnumber (void)**

Returns current http server port number.

This function returns the current http server port number

This function is safe to be called from multiple processes at the same time.

#### **Returns:**

http server port number

#### 8.5.4.12 `char* http_getrootdir (void)`

Returns http root directory.

This function returns the current root directory path name.

This function is safe to be called from multiple processes at the same time.

##### **Returns:**

The starting address of root directory path name. NULL will be returned if there is no root directory set.

#### 8.5.4.13 `char http_hex_from_ascii (char c)`

Returns hexadecimal value for input ascii digit.

This function returns hexadecimal value for input ascii digit

This function is safe to be called from multiple processes at the same time.

##### **Parameters:**

*c* ascii digit

##### **Returns:**

hexadecimal value for input ascii digit

#### 8.5.4.14 `int http_init (struct sockaddr server_addr)`

Initializes http server library.

This function initializes the internal data structures of http server library

##### **Warning:**

This function is not multi-process safe. If two processes try to call this function at the same time, its parameters may be destroyed, yielding unpredictable results.

##### **Parameters:**

*server\_addr* Address of http server. Should be initialized with the IP address of the TINI it is running on.

##### **Returns:**

Always returns [HTTP\\_STATUS\\_SUCCESS](#)

#### 8.5.4.15 `int http_kill_server (void)`

Terminates http server.

This function Terminates http server

This function is safe to be called from multiple processes at the same time.

##### **Returns:**

*HTTP\_STATUS\_SUCCESS*, *HTTP\_NOSERVERTASK*

#### 8.5.4.16 `void http_reg_acl_callback (int(*)() func)`

Registers access control callback function.

This function registers callback function to process http request

##### **Warning:**

This function is not multi-process safe. If two processes try to call this function at the same time, its parameters may be destroyed, yielding unpredictable results.

##### **Parameters:**

*func* - the function pointer to call back function

##### **NOTE:**

Access control callback function should have the following function prototype to receive the sockaddr pointer

`int << http access control call back function name >> (sockaddr *address)`

**NOTE:** Access control callback routine should return *HTTP\_STATUS\_SUCCESS* value to process the http request, *HTTP\_DENY\_CONNECTION* error value to deny the client connection

##### **Warning:**

this callback function should be multi-task safe.

#### 8.5.4.17 `void http_reg_req_callback (int(*)() func)`

Registers callback function to process http request.

This function registers callback function to process http request

##### **Warning:**

This function is not multi-process safe. If two processes try to call this function at the same time, its parameters may be destroyed, yielding unpredictable results.

**Parameters:**

*func* - the function pointer to call back function

**NOTE:** Http request process callback function should have the following function prototype to receive http session structure pointer.

int << http request process call back function name >> (http\_session \*https)

**NOTE:** Http request process callback routine should return *HTTP\_STATUS\_SUCCESS* value, if the request was processed. *HTTP\_REQUEST\_NOT\_PROCESSED* error value to let http library process the request.

**Warning:**

this callback function should be multi-task safe.

**8.5.4.18 void http\_sendheaders (http\_session \* https)**

Sends http response headers to client.

This function sends http response headers to client

This function is safe to be called from multiple processes at the same time.

**NOTE:** the response code, content type, content length, and response header values can be modified from application before sending response headers

**Parameters:**

*https* the pointer to http session object that contains response header value

**8.5.4.19 void http\_set\_req\_linesize (int buffersize)**

Sets the maximum size value of each line of http request.

This function sets the maximum size value of each line of http request

**Warning:**

This function is not multi-process safe. If two processes try to call this function at the same time, its parameters may be destroyed, yielding unpredictable results.

**Parameters:**

*buffersize* the maximum size value of each http request line

**8.5.4.20 void http\_set\_req\_process\_options (char flag)**

Sets Http request Processing options configuration flag value.

This function sets http request processing options configuration flag value. By default, The Http request Processing options flag contains ([HTTP\\_ENABLE\\_VARIABLE\\_PARSING|HTTP\\_ENABLE\\_MESSAGEBODY\\_READING](#)) value.

This function is safe to be called from multiple processes at the same time.

**Parameters:**

*flag* Http request Variable parsing status flag value. The following table shows the possible values for flag.

flag value	Description
<a href="#">HTTP_ENABLE_VARIABLE_PARSING HTTP_ENABLE_MESSAGEBODY_READING</a>	When the flag is set with this value, The http server parses the variables from "query string", and "message body" and stores it in variable list. The variable needs to be passed using standard convention(variablename=value) to make http server library to parse the variables successfully.
<a href="#">HTTP_DISABLE_VARIABLE_PARSING HTTP_ENABLE_MESSAGEBODY_READING</a>	This value makes http server library to disable variable parsing. But, library reads both "querystring" and "messagebody" part of http request, and user can access both "query string" and "messagebody" values from querystring,messagebody members of http session object.
<a href="#">HTTP_ENABLE_VARIABLE_PARSING HTTP_DISABLE_MESSAGEBODY_READING</a>	This value enables http server library to read "querystring" and parse querystring to extract http variables. But, http server does not read "messagebody" part of httprequest. It is user's responsibility to read and process messagebody.
<a href="#">HTTP_DISABLE_VARIABLE_PARSING HTTP_DISABLE_MESSAGEBODY_READING</a>	When the flag is set with this value, http server library reads "query string" value and keeps it in querystring member of http session object. Http server library neither parses "querystring" nor reads messagebody part of http request.

**8.5.4.21 void http\_setclientsockettimeout (long *milli\_sec*)**

Sets http client socket timeout value.

This function sets http client socket timeout value

**Warning:**

This function is not multi-process safe. If two processes try to call this function at the same time, its parameters may be destroyed, yielding unpredictable results.

**Parameters:**

*milli\_sec* timeout value in milliseconds

**8.5.4.22 void http\_setheaderbufsize (int *buffersize*)**

Sets header buffer size.

This function sets both http request and http response header buffer size

**Warning:**

This function is not multi-process safe. If two processes try to call this function at the same time, its parameters may be destroyed, yielding unpredictable results.

**Parameters:**

*buffersize* the input buffer size value

**8.5.4.23 int http\_setindexpage (char \* *index*)**

Sets index page name.

This function sets index page name in http library. Index page will be sent to http client if url request path is "/"

**Warning:**

This function is not multi-process safe. If two processes try to call this function at the same time, its parameters may be destroyed, yielding unpredictable results.

**Parameters:**

*index* Index page name

**Returns:**

- [\*HTTP\\_STATUS\\_SUCCESS\*](#) if the operation is completed successfully
- [\*HTTP\\_INSUFFICIENT\\_MEMORY\*](#) if memory can't be allocated for storing new index page name

**8.5.4.24 int http\_setlogfilename (char \* *logfilename*)**

Sets log file name.

This function sets log file name with http library.

**Warning:**

This function is not multi-process safe. If two processes try to call this function at the same time, its parameters may be destroyed, yielding unpredictable results.

**Parameters:**

*logfilename* - name of the log file

**Returns:**

- [\*HTTP\\_STATUS\\_SUCCESS\*](#) if the operation is completed successfully
- [\*HTTP\\_INSUFFICIENT\\_MEMORY\*](#) if memory can't be allocated for storing new log file name

NOTE: Logging activity has to be disabled and re-enabled in order to use new log file name for logging

**8.5.4.25 int http\_setlogging (char *logstatus*)**

Sets logging status.

This function sets the logging status with http server library

**Warning:**

This function is not multi-process safe. If two processes try to call this function at the same time, its parameters may be destroyed, yielding unpredictable results.

**Parameters:**

*logstatus* Logging status value. Either [\*HTTP\\_DISABLE\\_LOG\*](#) or [\*HTTP\\_ENABLE\\_LOG\*](#)

**Returns:**

[\*HTTP\\_STATUS\\_SUCCESS\*](#) or [\*HTTP\\_LOGFILE\\_ERROR\*](#)

**8.5.4.26 void http\_setmaxconnections (int *max\_connection*)**

Sets maximum number of pending connections value.

This function sets maximum number of pending connections allowed in the listen queue

**Warning:**

This function is not multi-process safe. If two processes try to call this function at the same time, its parameters may be destroyed, yielding unpredictable results.

**Parameters:**

*max\_connection* Value for maximum number of pending connections



#### 8.5.4.27 void http\_setportnumber (int *portnumber*)

Sets HTTP Server Port number.

This function sets http server port number

**Warning:**

This function is not multi-process safe. If two processes try to call this function at the same time, its parameters may be destroyed, yielding unpredictable results.

**Parameters:**

*portnumber* the input http server port number

NOTE: http server has to be stopped and re-started in order to use new http port number.

#### 8.5.4.28 int http\_setrootdir (char \* *rootdir*)

Sets http root directory.

This function sets http root directory in http library. url pathname for particular resource is "relative path name" to root directory.

**Warning:**

This function is not multi-process safe. If two processes try to call this function at the same time, its parameters may be destroyed, yielding unpredictable results.

**Parameters:**

*rootdir* root directory path

**Returns:**

- [\*HTTP\\_STATUS\\_SUCCESS\*](#) if the operation is completed successfully
- [\*HTTP\\_INSUFFICIENT\\_MEMORY\*](#) if memory can't be allocated for storing new root directory name

#### 8.5.4.29 int http\_start\_server (void)

Starts http server.

This function starts http server

This function is safe to be called from multiple processes at the same time.

**Returns:**

One of: [\*HTTP\\_STATUS\\_SUCCESS\*](#), [\*HTTP\\_TASK\\_ERROR\*](#), [\*HTTP\\_SOCKET\\_ERROR\*](#), or [\*HTTP\\_SERVER\\_ALREADY\\_RUNNING\*](#)

## 8.6 rom400\_init.h File Reference

### 8.6.1 Detailed Description

ROM Initialization functions in the DS80C400 ROM.

This library contains functions for initializing the functionality in the ROM. Note that the preferred way of initializing the ROM is to simply call the [init\\_rom](#) function. However, you can also initialize the various modules individually. To do this, call these functions in this order:

1. [init\\_clearXSEG](#)
2. [init\\_copyivt](#)
3. [init\\_redirect](#)
4. [init\\_sched](#)
5. [init\\_mm](#)
6. [init\\_km](#)
7. [init\\_ow](#)
8. [init\\_network](#)
9. [init\\_eth](#)
10. [init\\_sockets](#)
11. [init\\_tick](#)
12. [task\\_genesis](#) [in the process scheduler library]
13. [init\\_enableinterrupts](#)

For detailed information on the DS80C400 please see the [High-Speed Microcontroller User's Guide: DS80C400 Supplement](#).

Functions in this library should only be called once on startup. The safety of calling these functions from multiple processes at the same time is irrelevant.

#### Defines

- #define [ROM400\\_INIT\\_VERSION](#) 19
- #define [USE\\_KEIL\\_MONITOR](#)
- #define [INIT\\_DIVISOR\\_3MHZ](#) 0x01
- #define [INIT\\_DIVISOR\\_4MHZ](#) 0x08
- #define [INIT\\_DIVISOR\\_5MHZ](#) 0x02

- #define INIT\_DIVISOR\_6MHZ 0x05
  - #define INIT\_DIVISOR\_7MHZ 0x03
  - #define INIT\_DIVISOR\_8MHZ 0x0C
  - #define INIT\_DIVISOR\_10MHZ 0x06
  - #define INIT\_DIVISOR\_12MHZ 0x09
  - #define INIT\_DIVISOR\_14MHZ 0x07
  - #define INIT\_DIVISOR\_16MHZ 0x10
  - #define INIT\_DIVISOR\_20MHZ 0x0A
  - #define INIT\_DIVISOR\_24MHZ 0x0D
  - #define INIT\_DIVISOR\_28MHZ 0x0B
  - #define INIT\_DIVISOR\_32MHZ 0x14
  - #define INIT\_DIVISOR\_40MHZ 0x0E
  - #define INIT\_DIVISOR\_48MHZ 0x11
  - #define INIT\_DIVISOR\_56MHZ 0x0F
  - #define INIT\_DIVISOR\_64MHZ 0x18
  - #define INIT\_DIVISOR\_80MHZ 0x12
  - #define INIT\_DIVISOR\_96MHZ 0x15
  - #define INIT\_DIVISOR\_112MHZ 0x13
  - #define INIT\_DIVISOR\_128MHZ 0x1C
  - #define INIT\_POWERFAIL\_RESET 0x08
  - #define INIT\_WATCHDOG\_RESET 0x10
  - #define INIT\_CRYSTALFAIL\_RESET 0x20
  - #define DEFAULT\_HEAP\_START (((long)&HEAP\_START)&0x7ffffL)-0x10000L)
  - #define init\_setfrequency(clock) init\_setclock(((clock)\*5L)/60)
- Sets the crystal frequency.*

## Functions

- void **HEAP\_START** (void)
- void **init\_rom** (unsigned long mem\_start\_address, unsigned long mem\_end\_address)  
*Initializes the modules in the ROM.*
- void **init\_netboot** (void)  
*Starts the netboot functionality. Note that this will negate any initialization that has already been performed.*
- void **init\_copyivt** (void)  
*Copies the interrupt vector table into memory.*
- void **init\_redirect** (void)

*Sets up the redirect table for ROM redirected calls.*

- void `init_sched` (void)  
*Sets up the default task scheduler.*
- void `init_clearXSEG` (void)  
*Clears system variables in external RAM.*
- void `init_mm` (unsigned long mem\_start\_address, unsigned long mem\_end\_address)  
*Initializes the heap.*
- void `init_km` (void)  
*Initializes fast kernel memory.*
- void `init_ow` (unsigned char DIVISOR)  
*Initializes the internal I-Wire.*
- void `init_network` (void)  
*Initializes the network.*
- void `init_eth` (void)  
*Initializes the ethernet support.*
- void `init_sockets` (void)  
*Initializes the socket layer.*
- void `init_tick` (void)  
*Initializes the system timer.*
- void `init_enableinterrupts` (void)  
*Enables system interrupts.*
- void `init_usekeilmonitor` (void)  
*Performs initialization necessary for using the Keil Monitor.*
- unsigned int `init_version` (void)  
*Returns the version number of this initialization library.*
- unsigned char `init_getbootstate` (void)  
*Returns the boot status flags.*
- void `init_setclock` (unsigned int value)

*Sets the crystal frequency.*

## 8.6.2 Define Documentation

### 8.6.2.1 **#define DEFAULT\_HEAP\_START (((long)&HEAP\_START)&0x7ffff-L)-0x10000L)**

Defines the default start address for the heap.

See also:

[init\\_rom](#)

### 8.6.2.2 **#define INIT\_CRYSTALFAIL\_RESET 0x20**

Crystal failure reset status.

See also:

[init\\_getbootstate](#)

### 8.6.2.3 **#define INIT\_DIVISOR\_10MHZ 0x06**

1-Wire divisor value for operating frequencies greater than 10 MHz but less than 12 MHz.

See also:

[init\\_ow](#)

### 8.6.2.4 **#define INIT\_DIVISOR\_112MHZ 0x13**

1-Wire divisor value for operating frequencies greater than 112 MHz but less than 128 MHz.

See also:

[init\\_ow](#)

### 8.6.2.5 **#define INIT\_DIVISOR\_128MHZ 0x1C**

1-Wire divisor value for operating frequencies greater than 128 MHz.

See also:

[init\\_ow](#)

#### **8.6.2.6 #define INIT\_DIVISOR\_12MHZ 0x09**

1-Wire divisor value for operating frequencies greater than 12 MHz but less than 14 MHz.

**See also:**

[init\\_ow](#)

#### **8.6.2.7 #define INIT\_DIVISOR\_14MHZ 0x07**

1-Wire divisor value for operating frequencies greater than 14 MHz but less than 16 MHz.

**See also:**

[init\\_ow](#)

#### **8.6.2.8 #define INIT\_DIVISOR\_16MHZ 0x10**

1-Wire divisor value for operating frequencies greater than 16 MHz but less than 20 MHz.

**See also:**

[init\\_ow](#)

#### **8.6.2.9 #define INIT\_DIVISOR\_20MHZ 0x0A**

1-Wire divisor value for operating frequencies greater than 20 MHz but less than 24 MHz.

**See also:**

[init\\_ow](#)

#### **8.6.2.10 #define INIT\_DIVISOR\_24MHZ 0x0D**

1-Wire divisor value for operating frequencies greater than 24 MHz but less than 28 MHz.

**See also:**

[init\\_ow](#)

#### **8.6.2.11 #define INIT\_DIVISOR\_28MHZ 0x0B**

1-Wire divisor value for operating frequencies greater than 28 MHz but less than 32 MHz.

**See also:**

[init\\_ow](#)

#### **8.6.2.12 #define INIT\_DIVISOR\_32MHZ 0x14**

1-Wire divisor value for operating frequencies greater than 32 MHz but less than 40 MHz.

**See also:**

[init\\_ow](#)

#### **8.6.2.13 #define INIT\_DIVISOR\_3MHZ 0x01**

1-Wire divisor value for operating frequencies greater than 3 MHz but less than 4 MHz.

**See also:**

[init\\_ow](#)

#### **8.6.2.14 #define INIT\_DIVISOR\_40MHZ 0x0E**

1-Wire divisor value for operating frequencies greater than 40 MHz but less than 48 MHz.

**See also:**

[init\\_ow](#)

#### **8.6.2.15 #define INIT\_DIVISOR\_48MHZ 0x11**

1-Wire divisor value for operating frequencies greater than 48 MHz but less than 56 MHz.

**See also:**

[init\\_ow](#)

#### **8.6.2.16 #define INIT\_DIVISOR\_4MHZ 0x08**

1-Wire divisor value for operating frequencies greater than 4 MHz but less than 5 MHz.

**See also:**

[init\\_ow](#)

#### **8.6.2.17 #define INIT\_DIVISOR\_56MHZ 0x0F**

1-Wire divisor value for operating frequencies greater than 56 MHz but less than 64 MHz.

**See also:**

[init\\_ow](#)

#### **8.6.2.18 #define INIT\_DIVISOR\_5MHZ 0x02**

1-Wire divisor value for operating frequencies greater than 5 MHz but less than 6 MHz.

**See also:**

[init\\_ow](#)

#### **8.6.2.19 #define INIT\_DIVISOR\_64MHZ 0x18**

1-Wire divisor value for operating frequencies greater than 64 MHz but less than 80 MHz.

**See also:**

[init\\_ow](#)

#### **8.6.2.20 #define INIT\_DIVISOR\_6MHZ 0x05**

1-Wire divisor value for operating frequencies greater than 6 MHz but less than 7 MHz.

**See also:**

[init\\_ow](#)

#### **8.6.2.21 #define INIT\_DIVISOR\_7MHZ 0x03**

1-Wire divisor value for operating frequencies greater than 7 MHz but less than 8 MHz.

**See also:**

[init\\_ow](#)

#### **8.6.2.22 #define INIT\_DIVISOR\_80MHZ 0x12**

1-Wire divisor value for operating frequencies greater than 80 MHz but less than 96 MHz.

**See also:**

[init\\_ow](#)

#### **8.6.2.23 #define INIT\_DIVISOR\_8MHZ 0x0C**

1-Wire divisor value for operating frequencies greater than 8 MHz but less than 10 MHz.

**See also:**

[init\\_ow](#)



#### 8.6.2.24 **#define INIT\_DIVISOR\_96MHZ 0x15**

1-Wire divisor value for operating frequencies greater than 96 MHz but less than 112 MHz.

See also:

[init\\_ow](#)

#### 8.6.2.25 **#define INIT\_POWERFAIL\_RESET 0x08**

Power fail reset status.

See also:

[init\\_getbootstate](#)

#### 8.6.2.26 **#define init\_setfrequency(clock) init\_setclock(((clock)\*5L)/60)**

Sets the crystal frequency.

**Parameters:**

*clock* Clock frequency in kHz (e.g. 14746 for a 14.7456 MHz crystal). The operating frequency is the oscillator adjusted by any setting of the frequency multiplier (i.e. a 14 MHz oscillator with the clock doubler enabled should set 28 MHz)

Note that this macro has to be called before [init\\_rom](#).

See also:

[task\\_settickreload](#)

[init\\_rom](#)

[init\\_setclock](#)

#### 8.6.2.27 **#define INIT\_WATCHDOG\_RESET 0x10**

Watchdog reset status.

See also:

[init\\_getbootstate](#)

#### 8.6.2.28 **#define ROM400\_INIT\_VERSION 19**

Version number associated with this header file. Should be the same as the version number returned by the [init\\_version](#) function.

See also:

[init\\_version](#)

#### 8.6.2.29 **#define USE\_KEIL\_MONITOR**

Macro that allows the use of a define to determine whether or not to call the function *init\_usekeilmonitor*. This macro can be called after calling *init\_rom*, and will correct some monitor configuration details that are destroyed when *init\_rom* is called.

**See also:**

[init\\_rom](#)  
[init\\_usekeilmonitor](#)

### 8.6.3 **Function Documentation**

#### 8.6.3.1 **void HEAP\_START (void)**

Used to calculate the default start address for the heap.

**See also:**

[init\\_rom](#)

#### 8.6.3.2 **void init\_clearXSEG (void)**

Clears system variables in external RAM.

Note that calling *init\_rom* is the preferred way of initializing the ROM.

This function also sets the **EPFI** bit.

**See also:**

[init\\_rom](#)

#### 8.6.3.3 **void init\_copyivt (void)**

Copies the interrupt vector table into memory.

Note that calling *init\_rom* is the preferred way of initializing the ROM.

**See also:**

[init\\_rom](#)

#### 8.6.3.4 **void init\_enableinterrupts (void)**

Enables system interrupts.

Note that calling *init\_rom* is the preferred way of initializing the ROM.

**See also:**

[init\\_rom](#)

#### 8.6.3.5 void init\_eth (void)

Initializes the ethernet support.

Note that calling [init\\_rom](#) is the preferred way of initializing the ROM.

See also:

[init\\_rom](#)

#### 8.6.3.6 unsigned char init\_getbootstate (void)

Returns the boot status flags.

The status flags are defined as follows: Status.3 (0x08) - Power Fail Reset INIT\_POWERFAIL\_RESET Status.4 (0x10) - Watchdog Reset INIT\_WATCHDOG\_RESET Status.5 (0x20) - Crystal Oscillator Failure Reset INIT\_CRYSTALFAIL\_RESET All other bits are reserved, but not necessarily 0.

Returns:

Status flags

See also:

[INIT\\_POWERFAIL\\_RESET](#)  
[INIT\\_WATCHDOG\\_RESET](#)  
[INIT\\_CRYSTALFAIL\\_RESET](#)

#### 8.6.3.7 void init\_km (void)

Initializes fast kernel memory.

Note that calling [init\\_rom](#) is the preferred way of initializing the ROM.

See also:

[init\\_rom](#)

#### 8.6.3.8 void init\_mm (unsigned long mem\_start\_address, unsigned long mem\_end\_address)

Initializes the heap.

Note that calling [init\\_rom](#) is the preferred way of initializing the ROM.

Parameters:

*mem\_start\_address* The absolute beginning address for the heap (see [init\\_rom](#) for a detailed discussion of the input parameters). Unlike the [init\\_rom](#) function,

this function cannot accept **0** for default parameters. The start address must be specified. Use [DEFAULT\\_HEAP\\_START](#) to specify the default start address.

***mem\_end\_address*** The absolute ending address for the heap (see [init\\_rom](#) for a detailed discussion of the input parameters). Unlike the [init\\_rom](#) function, this function cannot accept **0** for default parameters. The end address must be specified.

**See also:**

[init\\_rom](#)  
[DEFAULT\\_HEAP\\_START](#)

#### **8.6.3.9 void init\_netboot (void)**

Starts the netboot functionality. Note that this will negate any initialization that has already been performed.

**See also:**

[init\\_rom](#)

#### **8.6.3.10 void init\_network (void)**

Initializes the network.

Note that calling [init\\_rom](#) is the preferred way of initializing the ROM.

**See also:**

[init\\_rom](#)

#### **8.6.3.11 void init\_ow (unsigned char *DIVISOR*)**

Initializes the internal 1-Wire.

Note that calling [init\\_rom](#) is the preferred way of initializing the ROM.

**Parameters:**

***DIVISOR*** Divisor value for given the DS80C400's operating frequency. The operating frequency is the oscillator adjusted by any setting of the frequency multiplier (i.e. a 14 MHz oscillator with the clock doubler enabled should look for a divisor for 28 MHz)

**See also:**

[init\\_rom](#)

INIT\_DIVISOR\_3MHZ  
INIT\_DIVISOR\_4MHZ  
INIT\_DIVISOR\_5MHZ  
INIT\_DIVISOR\_6MHZ  
INIT\_DIVISOR\_7MHZ  
INIT\_DIVISOR\_8MHZ  
INIT\_DIVISOR\_10MHZ  
INIT\_DIVISOR\_12MHZ  
INIT\_DIVISOR\_14MHZ  
INIT\_DIVISOR\_16MHZ  
INIT\_DIVISOR\_20MHZ  
INIT\_DIVISOR\_24MHZ  
INIT\_DIVISOR\_28MHZ  
INIT\_DIVISOR\_32MHZ  
INIT\_DIVISOR\_40MHZ  
INIT\_DIVISOR\_48MHZ  
INIT\_DIVISOR\_56MHZ  
INIT\_DIVISOR\_64MHZ  
INIT\_DIVISOR\_80MHZ  
INIT\_DIVISOR\_96MHZ  
INIT\_DIVISOR\_112MHZ  
INIT\_DIVISOR\_128MHZ

#### 8.6.3.12 void init\_redirect (void)

Sets up the redirect table for ROM redirected calls.

Note that calling [init\\_rom](#) is the preferred way of initializing the ROM.

See also:

[init\\_rom](#)

#### 8.6.3.13 void init\_rom (unsigned long *mem\_start\_address*, unsigned long *mem\_end\_address*)

Initializes the modules in the ROM.

Initializes the network stack, memory manager, process scheduler, and other modules in the DS80C400 Silicon Software. Calling this method is the preferred way of initializing the ROM.

Note that calling this function will cause the ROM to copy its own interrupt table into memory. If you have any interrupts installed before calling this function (for instance, you use the Keil compilers *interrupt* keyword to declare your function an interrupt handler), the entry in the interrupt table will be erased.

*init\_rom* prints status information to the serial port if serial 0 is set to use timer 2. If that is not desired, clear *tr2*. . . . TR2 = 0; *init\_rom*(...); TR2 = 1; . . .

*init\_rom* will probe all available 1-Wire devices for an approximate clock frequency and it will try to find a DS2502-E48 for an Ethernet MAC address. If no DS2502-E48 is present, you must use *init\_setfrequency* to specify a clock frequency, and you must modify *startup.a51* to manually set a MAC address.

#### Parameters:

***mem\_start\_address*** The absolute beginning address for the heap.

***mem\_end\_address*** The absolute ending address for the heap.

Use *mem\_start\_address*==0 to use the default settings for both start and end, or pass a value to *mem\_start\_address* and use *mem\_end\_address*==0 to use the remaining memory in the same bank, or use valid values for both addresses. Make sure the heap does not conflict with the XDATA segment (adjustable in project settings); you can examine the MAP file to determine the size of XDATA. Also note that the reentrant stack starts at the top of the XDATA segment (start address adjustable in *startup400.a51*).

Start address examples...

mem_start_ - address	mem_end_ - address	actual start	actual end	size of heap
0x000000	0x000000	0x002900	0x00FFFF	55040
<i>DEFAULT_ HEAP_ START</i>	0x07FFFF	0x002900	0x07FFFF	513792
0x010440	0x000000	0x010440	0x01FFFF	64448
0x010440	0x07FFFF	0x010440	0x07FFFF	457663

#### See also:

[DEFAULT\\_HEAP\\_START](#)

#### 8.6.3.14 void init\_sched (void)

Sets up the default task scheduler.

Note that calling *init\_rom* is the preferred way of initializing the ROM.

#### See also:

[init\\_rom](#)

### 8.6.3.15 void init\_setclock (unsigned int *value*)

Sets the crystal frequency.

#### Parameters:

*value* Clock frequency in kHz \* 5/60 (e.g. 1229 for a 14.7456 MHz crystal). The operating frequency is the oscillator adjusted by any setting of the frequency multiplier (i.e. a 14 MHz oscillator with the clock doubler enabled should set 28 MHz)

Note that this function has to be called before [init\\_rom](#). Users should call the more friendly macro [init\\_setfrequency](#).

#### See also:

[task\\_settickreload](#)  
[init\\_rom](#)  
[init\\_setfrequency](#)

### 8.6.3.16 void init\_sockets (void)

Initializes the socket layer.

Note that calling [init\\_rom](#) is the preferred way of initializing the ROM.

#### See also:

[init\\_rom](#)

### 8.6.3.17 void init\_tick (void)

Initializes the system timer.

Note that calling [init\\_rom](#) is the preferred way of initializing the ROM.

#### See also:

[init\\_rom](#)

### 8.6.3.18 void init\_usekeilmonitor (void)

Performs initialization necessary for using the Keil Monitor.

Performs initialization needed when using the Keil MON390 Monitor to debug programs that access the DS80C400's ROM. This function should be called after calling [init\\_rom](#), and only if the monitor will be used.

This file includes a macro [USE\\_KEIL\\_MONITOR](#) which is defined to call this function if *MONITOR* is defined. Use the following code to make use of this macro:

```
init_rom();  
USE_KEIL_MONITOR
```

**See also:**

[init\\_rom](#)  
[USE\\_KEIL\\_MONITOR](#)

### 8.6.3.19 unsigned int init\_version (void)

Returns the version number of this initialization library.

**Returns:**

Version number of this INIT library.

## 8.7 rom400\_kmem.h File Reference

### 8.7.1 Detailed Description

Kernel Memory initialization functions for the DS80C400 ROM.

This library allows users to allocate different amounts of memory as fast kernel buffers for use as ethernet buffers and as task control structures. The default allocation by the ROM may not be sufficient, and the use of multiple processes and multiple sockets might combine to drain all kernel memory. This library allows you to increase that amount for more complex applications.

There are two ways to use this library. 1) When using [init\\_rom](#): Call [kmem\\_install](#) before calling [init\\_rom](#).

2) When using the individual initialization functions: The function [kmem\\_init](#) is meant to replace the function [init\\_km](#) from the initialization library.

For detailed information on the DS80C400 please see the [High-Speed Microcontroller User's Guide: DS80C400 Supplement](#).

The functions in this library are multi-process safe—that is, if you call the same method from two different processes at the same time, the parameters to the function will not be destroyed. However, the function [kmem\\_init](#) is a system initialization function and should only be called once before the process scheduler is active.

### Data Structures

- struct [kmem\\_memory](#)



## Defines

- #define [ROM400\\_KMEM\\_VERSION](#) 7
- #define [ROM400\\_KMEM\\_MODEL\\_SMALLEST](#) 1
- #define [ROM400\\_KMEM\\_MODEL\\_LARGEST](#) 11

## Functions

- unsigned char [kmem\\_init](#) (unsigned char MODEL)  
*Initializes the kernel buffers.*
- unsigned int [kmem\\_version](#) (void)  
*Returns the version number of this KMEM library.*
- void [kmem\\_install](#) (unsigned char MODEL)  
*Installs the kmem library.*
- void [kmem\\_getfree](#) (struct [kmem\\_memory](#) \*freeblocks)  
*Returns a count of free kmem blocks.*
- unsigned char [kmem\\_getblockcount](#) (unsigned char offset)  
*Returns a count of free kmem blocks at specified offset.*

### 8.7.2 Define Documentation

#### 8.7.2.1 #define [ROM400\\_KMEM\\_MODEL\\_LARGEST](#) 11

The largest value of the argument to be passed to [kmem\\_init](#).

See also:

[ROM400\\_KMEM\\_MODEL\\_SMALLEST](#)  
[kmem\\_init](#)

#### 8.7.2.2 #define [ROM400\\_KMEM\\_MODEL\\_SMALLEST](#) 1

The smallest value of the argument to be passed to [kmem\\_init](#).

See also:

[ROM400\\_KMEM\\_MODEL\\_LARGEST](#)  
[kmem\\_init](#)

### 8.7.2.3 #define ROM400\_KMEM\_VERSION 7

Version number associated with this header file. Should be the same as the version number returned by the [kmem\\_version](#) function.

See also:

[kmem\\_version](#)

## 8.7.3 Function Documentation

### 8.7.3.1 unsigned char kmem\_getblockcount (unsigned char *offset*)

Returns a count of free kmem blocks at specified offset.

Allows user applications to query the kernel memory allocator for detailed free block information.

**Parameters:**

*offset* offset of requested block count (range of 0 to 6)

### 8.7.3.2 void kmem\_getfree (struct [kmem\\_memory](#) \* *freeblocks*)

Returns a count of free kmem blocks.

Allows user applications to query the kernel memory allocator for detailed free block information.

**Parameters:**

*freeblocks* pointer to structure to hold the current free block counts

### 8.7.3.3 unsigned char kmem\_init (unsigned char *MODEL*)

Initializes the kernel buffers.

Allows user applications to specify the amount of kernel memory that will be available to the system. Kernel memory is used internally for Ethernet buffers and task control structures, and as such can limit the number of processes or sockets an application can use concurrently if there is not enough kernel buffer space. The default kernel buffer allocation given by the ROM is:

- 90 byte buffers (20 count)
- 256 byte buffers (2 count)
- 512 byte buffers (1 count)
- 768 byte buffers (1 count)

- 1024 byte buffers (1 count)
- 1280 byte buffers (1 count)
- 1600 byte buffers (2 count)

By calling this function, the count of kernel buffers is multiplied by the value *MODEL*. Note that while **ROM400\_KMEM\_MODEL\_LARGEST** is the largest amount of kernel memory that the system can support, few applications will need to go beyond **ROM400\_KMEM\_MODEL\_SMALLEST** + 2.

**Parameters:**

*MODEL* specifies how much kernel memory will be allocated for the system

**Returns:**

0 for success, non-zero for failure.

**See also:**

[init\\_rom](#)

#### 8.7.3.4 void kmem\_install (unsigned char *MODEL*)

Installs the kmem library.

This function must be called before [init\\_rom](#).

Allows user applications to specify the amount of kernel memory that will be available to the system. Kernel memory is used internally for Ethernet buffers and task control structures, and as such can limit the number of processes or sockets an application can use concurrently if there is not enough kernel buffer space. The default kernel buffer allocation given by the ROM is:

- 90 byte buffers (20 count)
- 256 byte buffers (2 count)
- 512 byte buffers (1 count)
- 768 byte buffers (1 count)
- 1024 byte buffers (1 count)
- 1280 byte buffers (1 count)
- 1600 byte buffers (2 count)

By calling this function, the count of kernel buffers is multiplied by the value *MODEL*. Note that while **ROM400\_KMEM\_MODEL\_LARGEST** is the largest amount of kernel memory that the system can support, few applications will need to go beyond **ROM400\_KMEM\_MODEL\_SMALLEST** + 2.

**Parameters:**

**MODEL** specifies how much kernel memory will be allocated for the system

**See also:**

[init\\_rom](#)

### 8.7.3.5 unsigned int kmem\_version (void)

Returns the version number of this KMEM library.

**Returns:**

Version number of this KMEM library.

## 8.8 rom400\_mem.h File Reference

### 8.8.1 Detailed Description

Memory management functions in the DS80C400 ROM.

This library contains functions for allocating and de-allocating blocks of memory through the ROM's memory manager.

For detailed information on the DS80C400 please see the [High-Speed Microcontroller User's Guide: DS80C400 Supplement](#).

The methods in this library are all multi-process safe. That is, a function can be called by more than one process at the same time and its parameters won't be destroyed.

**Defines**

- #define [ROM400\\_MEM\\_VERSION](#) 6

**Functions**

- void \* [mem\\_malloc](#) (unsigned int size)  
*Requests a block of memory to be allocated.*
- void \* [mem\\_mallocdirty](#) (unsigned int size)  
*Requests a block of memory to be allocated.*
- unsigned char [mem\\_free](#) (void \*ptr)  
*De-allocates a block of memory.*
- unsigned long [mem\\_getfreeram](#) (void)

*Returns the amount of memory available for allocation.*

- unsigned int [mem\\_sizeof](#) (void \*ptr)  
*Gets the size of an allocated block of memory.*
- unsigned int [mem\\_version](#) (void)  
*Returns the version number of this memory management library.*
- void [mem\\_coalesce](#) (void)  
*Join adjacent chunks of freed memory.*

## 8.8.2 Define Documentation

### 8.8.2.1 #define ROM400\_MEM\_VERSION 6

Version number associated with this header file. Should be the same as the version number returned by the [mem\\_version](#) function.

See also:

[mem\\_version](#)

## 8.8.3 Function Documentation

### 8.8.3.1 void mem\_coalesce (void)

Join adjacent chunks of freed memory.

When the memory manager frees allocated memory, it makes no attempt to rejoin adjacent pieces of memory. Therefore, the memory becomes fragmented over time unless the allocation calls are very careful. This function will join adjacent pieces of memory and make the larger piece available for allocation.

### 8.8.3.2 unsigned char mem\_free (void \*ptr)

De-allocates a block of memory.

Deallocates a block of memory that was previously allocated by calling [mem\\_malloc](#) or [mem\\_mallocdirty](#), making this block available for re-allocation. Use the function [mem\\_getfreeram](#) to determine how much memory is available for allocation.

This is a redirected function. The ROM includes a default memory manager implementation. See the [DS80C400 User's Guide](#) for information on replacing the default memory manager with your own memory manager.

**Parameters:**

*ptr* pointer to the beginning of the previously allocated memory

**Returns:**

0 for success, non-zero for failure

**See also:**

[mem\\_malloc](#)  
[mem\\_mallocdirty](#)  
[mem\\_getfreeram](#)  
[mem\\_sizeof](#)

**8.8.3.3 unsigned long mem\_getfreeram (void)**

Returns the amount of memory available for allocation.

Returns the total amount of memory available for allocation. Memory is allocated in increments of 32 bytes. Due to fragmentation, large memory allocations may not be possible.

Note that the size returned by this function includes the memory manager overhead for this particular block. For example, if you request 512 bytes in a call to [mem\\_malloc](#), this function will report the amount 512 plus overhead size, rounded up to the next 32-byte block (thus returning 544).

This is a redirected function. The ROM includes a default memory manager implementation. See the [DS80C400 User's Guide](#) for information on replacing the default memory manager with your own memory manager.

**See also:**

[mem\\_sizeof](#)

**Returns:**

The amount of memory available for allocation from the memory manager.

**8.8.3.4 void\* mem\_malloc (unsigned int size)**

Requests a block of memory to be allocated.

Tries to allocate a block of memory of the requested size (maximum size of 64K). The data allocated is filled with 0's (similar to the traditional "calloc" library function). To request non-cleared memory (and save the extra time) use [mem\\_mallocdirty](#). To de-allocate the memory block, use [mem\\_free](#).

This is a redirected function. The ROM includes a default memory manager implementation. See the [DS80C400 User's Guide](#) for information on replacing the default memory manager with your own memory manager.

**Parameters:**

*size* amount of data requested for allocation

**Returns:**

pointer to the newly allocated memory, or NULL (0) if the operation failed

**See also:**

[mem\\_mallocdirty](#)  
[mem\\_free](#)  
[mem\\_sizeof](#)

**8.8.3.5 void\* mem\_mallocdirty (unsigned int size)**

Requests a block of memory to be allocated.

Tries to allocate a block of memory of the requested size (maximum size of 64K). The data allocated is NOT filled with 0's, and is likely to be filled with unpredictable values. To request cleared memory, use [mem\\_malloc](#). To de-allocate the memory block, use [mem\\_free](#).

This is a redirected function. The ROM includes a default memory manager implementation. See the [DS80C400 User's Guide](#) for information on replacing the default memory manager with your own memory manager.

**Parameters:**

*size* amount of data requested for allocation

**Returns:**

pointer to the newly allocated memory, or NULL (0) if the operation failed

**See also:**

[mem\\_malloc](#)  
[mem\\_free](#)  
[mem\\_sizeof](#)

**8.8.3.6 unsigned int mem\_sizeof (void \* ptr)**

Gets the size of an allocated block of memory.

Returns the size of a block of memory that was allocated by the ROM's default memory manager. If the input pointer is not a valid pointer that was created by an earlier call to [mem\\_malloc](#) or [mem\\_mallocdirty](#), the value returned has no meaning.

This is **NOT** a redirected function, and only functions if the ROM's default memory manager is used.

**Parameters:**

*ptr* pointer to the beginning of the previously allocated memory

**Returns:**

size of the memory allocated for a valid input pointer

**See also:**

[mem\\_malloc](#)  
[mem\\_mallocdirty](#)  
[mem\\_getfreeram](#)

**8.8.3.7 unsigned int mem\_version (void)**

Returns the version number of this memory management library.

**Returns:**

Version number of this memory management library.

**8.9 rom400\_netif.h File Reference****8.9.1 Detailed Description**

Network interface library for the DS80C400.

This library allows a user to add network interface drivers to the network stack.

For detailed information on the DS80C400 please see the [High-Speed Microcontroller User's Guide: DS80C400 Supplement](#).

**Defines**

- #define [ROM400\\_NETIF\\_VERSION](#) 2

**Functions**

- unsigned int [netif\\_version](#) (void)  
*Returns the version number of this NETIF library.*
- int [netif\\_packetreceived](#) (unsigned char \*packet, int len)  
*Submit an inbound packet to the network stack.*
- int [netif\\_addinterface](#) (char \*name, unsigned long ip, unsigned long subnet, unsigned long gateway, unsigned char flags, int(\*transmitter)(unsigned char \*packet, int len), int mtu, unsigned char timeout)  
*Add an interface to the network interface list.*



- int [netif\\_removeinterface](#) (char \*name)  
*Remove specified interface from the network interface list.*
- int [netif\\_setdefaultinterface](#) (char \*name)  
*Set the specified interface as default interface.*

## 8.9.2 Define Documentation

### 8.9.2.1 #define ROM400\_NETIF\_VERSION 2

Version number associated with this header file. Should be the same as the version number returned by the [netif\\_version](#) function.

See also:

[netif\\_version](#)

## 8.9.3 Function Documentation

### 8.9.3.1 int netif\_addinterface (char \* *name*, unsigned long *ip*, unsigned long *subnet*, unsigned long *gateway*, unsigned char *flags*, int(\*) (unsigned char \*packet, int len) *transmitter*, int *mtu*, unsigned char *timeout*)

Add an interface to the network interface list.

**Parameters:**

***name*** name of network interface (e.g. "ppp0")  
***ip*** IP address of the new interface (MSB first, e.g. 0x0a000002L for 10.0.0.2)  
***subnet*** subnet mask of the new interface (e.g. 0xff000000L for 255.0.0.0)  
***gateway*** gateway IP address of the new interface (e.g. 0x0a000001L for 10.0.0.1)  
***flags*** set to 1 if new interface should be the default interface (else 0)  
***transmitter*** address of the user supplied transmit function (see below)  
***mtu*** maximum transmission unit  
***timeout*** initial tcp timeout period based on an 8Hz tick; must be 8, 16, 32, 64 or 128 (default: 128)

**Returns:**

1 for success, 0 for failure

See also:

[netif\\_removeinterface](#)

The transmitter function *int transmitter(unsigned char \*packet, int len)* should return 1 when the packet was successfully sent (or dropped) and the packet memory should be freed. If the packet couldn't be sent and the packet should be retried, the transmitter should return 0. The argument *packet* points to the IP packet data to be transmitted and *length* is the length of the IP packet. Note that the transmit function runs under interrupt. Registers are saved, but only thread-safe functions can be called.

### 8.9.3.2 int netif\_packetreceived (unsigned char \* packet, int len)

Submit an inbound packet to the network stack.

#### Parameters:

*packet* IP packet  
*len* length of the packet

#### Returns:

1 for success, 0 for failure

Causes for failure are:

- The ROM IP\_CHECKHEADER routine doesn't like the packet.
- KMalloc can't allocate the memory required for the packet.

IP\_CHECKHEADER would decline a packet if any of the following were true:

- not an IPv4 packet
- fragment offset not 0
- smaller than 28 bytes
- source IP is 0.x.x.x, or 127.x.x.x, or 255.x.x.x
- destination IP is 0.x.x.x, or 127.x.x.x
- destination IP is not ours
- packet not IGMP, ICMP, UDP, or TCP
- multicast packet we're not interested in (not joined to this group)

### 8.9.3.3 `int netif_removeinterface (char * name)`

Remove specified interface from the network interface list.

**Parameters:**

*name* name of network interface to remove

**Returns:**

1 for success, 0 for failure

**See also:**

[netif\\_addinterface](#)

**NOTE:** The behavior of this function is not guaranteed if a network interface is removed while output traffic for the interface is still pending. It is recommended to close all sockets and delay for a few seconds before removing any network interface.

### 8.9.3.4 `int netif_setdefaultinterface (char * name)`

Set the specified interface as default interface.

**Parameters:**

*name* name of network interface

**Returns:**

1 for success, 0 for failure

### 8.9.3.5 `unsigned int netif_version (void)`

Returns the version number of this NETIF library.

**Returns:**

Version number of this NETIF library.

## 8.10 `rom400_netstat.h` File Reference

### 8.10.1 Detailed Description

Network statistics library for the DS80C400.

This library contains functions that return pointers to network information tables in the socket library.

Note that the tables and structures returned by these functions are the actual, physical tables used by the network stack and should not be modified by user applications.

Since these are the actual network structures, it is possible they might change while an application is processing them. Any critical analysis of these structures should be protected from interruption.

For detailed information on the DS80C400 please see the [High-Speed Microcontroller User's Guide: DS80C400 Supplement](#).

## Data Structures

- struct [netstat\\_arp\\_entry](#)
- struct [netstat\\_udp\\_entry](#)
- struct [netstat\\_tcp\\_socket](#)

## Defines

- #define [ROM400\\_NETSTAT\\_VERSION](#) 2
- #define [NETSTAT\\_ROM\\_ARP\\_ENTRIES](#) 8
- #define [NETSTAT\\_ARP\\_USED](#) 1
- #define [NETSTAT\\_ARP\\_REPLY\\_PENDING](#) 2
- #define [NETSTAT\\_ARP\\_STATIC](#) 4
- #define [NETSTAT\\_UDP\\_ENTRIES](#) 16  
*Number of entries in the UDP port table.*
- #define [NETSTAT\\_UDP\\_USED](#) 1  
*Values for [netstat\\_udp\\_entry.flags](#). Table entry is used.*
- #define [NETSTAT\\_TCP\\_MAX\\_SOCKETS](#) 25  
*Maximum number of sockets supported.*
- #define [NETSTAT\\_TCP\\_OUTPUT\\_NEEDED\\_MASK](#) 2  
*Value for [netstat\\_tcp\\_socket.flags](#). Either ACK or data or both.*
- #define [NETSTAT\\_TCP\\_ACK\\_NEEDED\\_MASK](#) 4  
*Value for [netstat\\_tcp\\_socket.flags](#). Need an ACK.*
- #define [NETSTAT\\_TCP\\_SERVER\\_MASK](#) 8  
*Value for [netstat\\_tcp\\_socket.flags](#). This is a server connection.*
- #define [NETSTAT\\_TCP\\_RESERVED\\_MASK](#) 16  
*Value for [netstat\\_tcp\\_socket.flags](#). (Reserved).*
- #define [NETSTAT\\_TCP\\_HAVE\\_OUTPUT\\_DATA\\_MASK](#) 32  
*Value for [netstat\\_tcp\\_socket.flags](#). Have data in output buffer.*

- #define `NETSTAT_TCP_HAVE_FIN_MASK` 64  
Value for `netstat_tcp_socket.flags`. Set when we receive a *FIN*.
- #define `NETSTAT_TCP_SEND_FIN_MASK` 128  
Value for `netstat_tcp_socket.flags`. Send a *FIN* after all data sent.
- #define `NETSTAT_TCP_OPTION_NAGLE_ENABLED_MASK` 1  
Value for `netstat_tcp_socket.options`. Set when Nagle's algorithm enabled.
- #define `NETSTAT_TCP_OPTION_IPV6_MASK` 2  
Value for `netstat_tcp_socket.options`. Set when we should talk IPv6 on the socket.
- #define `NETSTAT_TCP_OPTION_SOCKET_ASSIGNED` 4  
Value for `netstat_tcp_socket.options`. Assigned an application socket for this *TCB*.
- #define `NETSTAT_TCP_STATE_CLOSED` 0  
Value for `netstat_tcp_socket.state`. The socket is closed.
- #define `NETSTAT_TCP_STATE_LISTEN` 1  
Value for `netstat_tcp_socket.state`. The socket is listening.
- #define `NETSTAT_TCP_STATE_SYN_SENT` 2  
Value for `netstat_tcp_socket.state`. The socket has sent a *SYN*.
- #define `NETSTAT_TCP_STATE_SYN_RECEIVED` 3  
Value for `netstat_tcp_socket.state`. The socket had received a *SYN*.
- #define `NETSTAT_TCP_STATE_ESTABLISHED` 4  
Value for `netstat_tcp_socket.state`. The socket connection has been established.
- #define `NETSTAT_TCP_STATE_FIN_WAIT_1` 5  
Value for `netstat_tcp_socket.state`. The socket has been closed, and is waiting for its peer to close.
- #define `NETSTAT_TCP_STATE_FIN_WAIT_2` 6  
Value for `netstat_tcp_socket.state`. The socket's peer has *ACKed* a *FIN*.
- #define `NETSTAT_TCP_STATE_CLOSE_WAIT` 7  
Value for `netstat_tcp_socket.state`. The socket's peer has sent a *FIN*, the application should now close the socket.
- #define `NETSTAT_TCP_STATE_LAST_ACK` 8

Value for `netstat_tcp_socket.state`. The socket has closed, and is waiting for it's peer to ACK.

- `#define NETSTAT_TCP_STATE_CLOSING 9`  
Value for `netstat_tcp_socket.state`. Both ends have closed the socket.
- `#define NETSTAT_TCP_STATE_TIME_WAIT 10`  
Value for `netstat_tcp_socket.state`. Timeout wait before returning to closed state.

## Functions

- unsigned int `netstat_version` (void)  
Returns the version number of this NETSTAT library.
- `netstat_arp_entry` far \* `netstat_get_arp_table` (void)  
Returns a pointer to the ARP cache table.
- unsigned int `netstat_num_arp_entries` (void)  
Returns the number of entries in the ARP cache table.
- `netstat_udp_entry` far \* `netstat_get_udp_table` (void)  
Returns a pointer to the UDP port table.
- unsigned int `netstat_num_udp_entries` (void)  
Returns the number of entries in the UDP port table.
- `netstat_tcp_socket` far \* `netstat_get_tcp_socket` (unsigned int conn)  
Returns a pointer to a TCP socket information block.
- unsigned int `netstat_num_tcp_sockets` (void)  
Returns the number of entries in the TCP socket table.

## 8.10.2 Define Documentation

### 8.10.2.1 `#define NETSTAT_ARP_REPLYPENDING 2`

Value for `netstat_arp_entry.flags`. Table entry is not yet valid, request has been sent out

### 8.10.2.2 `#define NETSTAT_ARP_STATIC 4`

Value for `netstat_arp_entry.flags`. Table entry does not expire

#### 8.10.2.3 **#define NETSTAT\_ARP\_USED 1**

Value for [netstat\\_arp\\_entry.flags](#). Table entry is used

#### 8.10.2.4 **#define NETSTAT\_ROM\_ARP\_ENTRIES 8**

Maximum number of ARP table entries. Only valid when not using Enhanced Network Stack. Use [xnetstack\\_get\\_arptablesizes](#) if using Enhanced Network Stack

#### 8.10.2.5 **#define ROM400\_NETSTAT\_VERSION 2**

Version number associated with this header file. Should be the same as the version number returned by the [netstat\\_version](#) function.

See also:

[netstat\\_version](#)

### 8.10.3 **Function Documentation**

#### 8.10.3.1 **[netstat\\_arp\\_entry](#) far\* [netstat\\_get\\_arp\\_table](#) (void)**

Returns a pointer to the ARP cache table.

This function returns a pointer to the ARP cache table. Each entry is a [netstat\\_arp\\_entry](#). The entry is used when its "flags" has the [NETSTAT\\_ARP\\_USED](#) bit set.

**Returns:**

Far pointer to the ARP cache table

#### 8.10.3.2 **[netstat\\_tcp\\_socket](#) far\* [netstat\\_get\\_tcp\\_socket](#) (unsigned int *conn*)**

Returns a pointer to a TCP socket information block.

This function returns a pointer to a specific TCP socket information block of type [netstat\\_tcp\\_socket](#). There are at most [NETSTAT\\_TCP\\_MAXSOCKETS](#), the function returns NULL when a given socket number doesn't exist. Note that the actual number of sockets in the socket table might change at any time. Table entries are not guaranteed to be contiguous. A user application \* should therefore call this function for all values from 0 to [NETSTAT\\_TCP\\_MAXSOCKETS](#) - 1 and discard non-existent entries.

**Parameters:**

*conn* Socket number

**Returns:**

Far pointer to the socket's information block (or NULL if the socket doesn't exist).

#### **8.10.3.3   `netstat_udp_entry` far\* `netstat_get_udp_table` (void)**

Returns a pointer to the UDP port table.

This function returns a pointer to the UDP port table. There are `NETSTAT_UDP_ENTRIES` in the UDP port table. Each entry is a `netstat_udp_entry`. The entry is used when its "flags" has the `NETSTAT_UDP_USED` bit set.

##### **Returns:**

Far pointer to the UDP port table

#### **8.10.3.4   `unsigned int` `netstat_num_arp_entries` (void)**

Returns the number of entries in the ARP cache table.

This function returns the number of used entries in the ARP cache table (entries with the `NETSTAT_ARP_USED` flag set).

##### **Returns:**

Number of entries in the ARP cache table

#### **8.10.3.5   `unsigned int` `netstat_num_tcp_sockets` (void)**

Returns the number of entries in the TCP socket table.

This function returns the number of used entries in the TCP socket table.

##### **Returns:**

Number of entries in the TCP socket table

#### **8.10.3.6   `unsigned int` `netstat_num_udp_entries` (void)**

Returns the number of entries in the UDP port table.

This function returns the number of used entries in the UDP port table (entries with the `NETSTAT_UDP_USED` flag set).

##### **Returns:**

Number of entries in the UDP port table

#### **8.10.3.7   `unsigned int` `netstat_version` (void)**

Returns the version number of this NETSTAT library.

##### **Returns:**

Version number of this NETSTAT library.



## 8.11 rom400\_ow.h File Reference

### 8.11.1 Detailed Description

Raw 1-Wire functions in the DS80C400 ROM.

This library contains functions for finding and communicating with devices on the internal 1-Wire. These functions use the DS80C400's 1-Wire master, applications do not need to worry about protecting the ROM 1-Wire routines from interruption.

For detailed information on the DS80C400 please see the [High-Speed Microcontroller User's Guide: DS80C400 Supplement](#).

These functions are all safe to be called from multiple processes simultaneously. That is, if two processes call one of these functions at the same time, the function parameters will not be destroyed. However, two processes attempting 1-Wire communications at the same time will surely cause communications problems. In addition, the memory space that ROM ID's are stored in is global for the system. Therefore, processes should synchronize around all 1-Wire communication sessions.

#### Defines

- #define [ROM400\\_OW\\_VERSION](#) 4
- #define [OW\\_RESET\\_SHORT](#) 0
- #define [OW\\_RESET\\_PRESENCE](#) 1
- #define [OW\\_RESET\\_ALARM](#) 2
- #define [OW\\_RESET\\_NO\\_PRESENCE](#) 3

#### Functions

- unsigned char [ow\\_first](#) (void)  
*Searches for the first device on the 1-Wire bus.*
- unsigned char [ow\\_next](#) (void)  
*Searches the 1-Wire for subsequent devices.*
- unsigned char [ow\\_reset](#) (void)  
*Sends a reset signal to the 1-Wire bus.*
- unsigned char [ow\\_byte](#) (unsigned char x)  
*Sends/receives a byte to/from the 1-Wire bus.*
- unsigned char \* [ow\\_getcurrentid](#) (void)  
*Returns a pointer to the address of the current device in a 1-Wire bus search.*

- unsigned int [ow\\_version](#) (void)  
*Returns the version number of this 1-Wire library.*

## 8.11.2 Define Documentation

### 8.11.2.1 #define OW\_RESET\_ALARM 2

Result of a [ow\\_reset](#) operation. There is an alarming device on the 1-Wire bus.

See also:

[ow\\_reset](#)

### 8.11.2.2 #define OW\_RESET\_NO\_PRESENCE 3

Result of a [ow\\_reset](#) operation. There is no device on the 1-Wire bus.

See also:

[ow\\_reset](#)

### 8.11.2.3 #define OW\_RESET\_PRESENCE 1

Result of a [ow\\_reset](#) operation. There is a device on the 1-Wire bus.

See also:

[ow\\_reset](#)

### 8.11.2.4 #define OW\_RESET\_SHORT 0

Result of a [ow\\_reset](#) operation. The 1-Wire bus is shorted.

See also:

[ow\\_reset](#)

### 8.11.2.5 #define ROM400\_OW\_VERSION 4

Version number associated with this header file. Should be the same as the version number returned by the [ow\\_version](#) function.

See also:

[ow\\_version](#)

### 8.11.3 Function Documentation

#### 8.11.3.1 `unsigned char ow_byte (unsigned char x)`

Sends/receives a byte to/from the 1-Wire bus.

Sends the input byte to the 1-Wire bus, and returns any byte transmitted from the 1-Wire bus. Send the byte 0xFF to return the result of a transmission by the slave (the device or iButton).

**Parameters:**

*x* byte to write to the 1-Wire bus

**Returns:**

Byte read from the 1-Wire bus

#### 8.11.3.2 `unsigned char ow_first (void)`

Searches for the first device on the 1-Wire bus.

Tries to access the first device on the 1-Wire bus. After a call to [\*ow\\_first\*](#), use the address returned by [\*ow\\_getcurrentid\*](#) to read the 8 byte Address of the device. To read all the devices present, call this method only once, and then call [\*ow\\_next\*](#) to read all subsequent devices.

**Returns:**

Non-zero if a device is found, 0 if no devices are found.

**See also:**

[\*ow\\_next\*](#)  
[\*ow\\_getcurrentid\*](#)

#### 8.11.3.3 `unsigned char* ow_getcurrentid (void)`

Returns a pointer to the address of the current device in a 1-Wire bus search.

Use the pointer returned by this method after every call to [\*ow\\_first\*](#) and [\*ow\\_next\*](#). Note that calls to these functions destroy what was previously held at this address. Programs that need to remember all the devices found should copy the addresses one at a time as the 1-Wire bus is searched.

**Returns:**

Pointer to the 8-byte device address.

**See also:**

[\*ow\\_first\*](#)  
[\*ow\\_next\*](#)

#### 8.11.3.4 unsigned char ow\_next (void)

Searches the 1-Wire for subsequent devices.

Call [ow\\_first](#) once before making subsequent calls to [ow\\_next](#) to find the second, third, and so on devices. After a successful call to [ow\\_next](#), call the function [ow\\_getcurrentid](#) to get the unique 64-bit address of the device found.

##### Returns:

Non-zero if a device is found, 0 if no more devices are found.

##### See also:

[ow\\_first](#)

[ow\\_getcurrentid](#)

#### 8.11.3.5 unsigned char ow\_reset (void)

Sends a reset signal to the 1-Wire bus.

The result of a reset tells you if the bus is shorted, if a device is present, if an alarming device is present, or if no device is present.

##### Returns:

Result of reset (i.e. [OW\\_RESET\\_SHORT](#))

##### See also:

[OW\\_RESET\\_SHORT](#)

[OW\\_RESET\\_PRESENCE](#)

[OW\\_RESET\\_ALARM](#)

[OW\\_RESET\\_NO\\_PRESENCE](#)

#### 8.11.3.6 unsigned int ow\_version (void)

Returns the version number of this 1-Wire library.

##### Returns:

Version number of this 1-Wire library.

## 8.12 rom400\_rarp.h File Reference

### 8.12.1 Detailed Description

RARP library for the DS80C400.

This library allows a user to send a RARP request to the network.

For detailed information on the DS80C400 please see the [High-Speed Microcontroller User's Guide: DS80C400 Supplement](#).

## Defines

- `#define ROM400_RARP_VERSION 1`

## Functions

- unsigned int `rarp_version` (void)  
*Returns the version number of this RARP library.*
- void `rarp_send` (void(\*callback)(unsigned long))  
*Send a RARP request.*
- void `rarp_stop` (void)  
*Disable reception of RARP packets (in the event of a timeout).*

### 8.12.2 Define Documentation

#### 8.12.2.1 `#define ROM400_RARP_VERSION 1`

Version number associated with this header file. Should be the same as the version number returned by the `rarp_version` function.

#### See also:

`rarp_version`

### 8.12.3 Function Documentation

#### 8.12.3.1 void `rarp_send` (void(\*) (unsigned long) *callback*)

Send a RARP request.

#### Parameters:

*callback* function that gets called when RARP receives an IP address (the IP address will be supplied to callback MSB first)

#### 8.12.3.2 void `rarp_stop` (void)

Disable reception of RARP packets (in the event of a timeout).

If RARP receives an IP address, it is not necessary to call this function. This function is only necessary if the callback from `rarp_send` was never called.

### 8.12.3.3 unsigned int rarp\_version (void)

Returns the version number of this RARP library.

#### Returns:

Version number of this RARP library.

## 8.13 rom400\_sock.h File Reference

### 8.13.1 Detailed Description

Socket functions in the DS80C400 ROM.

This library contains functions for TCP, UDP and Multicast sockets, as well as network configuration. The functions in this library **are** safe to be called from multiple processes at the same time, with the exception of the function *ping*. Both the traditional Berkeley style socket API and the *synchronized* socket functions are supported (the Berkeley style API is supported through macros implemented by the synchronized functions).

It is recommended that new applications use the Berkeley style API for portability.

Note that in order to run at 100 Mbs, the DS80C400 must be running at least 25MHz. This can be accomplished on the TINIm400 module by enabling the clock doubler.

For detailed information on the DS80C400 please see the [High-Speed Microcontroller User's Guide: DS80C400 Supplement](#).

```
#include <stdlib.h>
```

#### Data Structures

- struct [sockaddr](#)
- struct [in\\_addr](#)
- struct [in6\\_addr](#)
- struct [sockaddr\\_in](#)
- struct [pingdata](#)

#### Defines

- #define [ROM400\\_SOCKET\\_VERSION](#) 12
- #define [ROM400\\_SOCKET\\_SYNCH\\_VERSION](#) ROM400\_SOCKET\_VERSION
- #define [SOCKET\\_TYPE\\_DATAGRAM](#) 0
- #define [SOCKET\\_TYPE\\_STREAM](#) 1
- #define [SOCK\\_DGRAM](#) SOCKET\_TYPE\_DATAGRAM
- #define [SOCK\\_STREAM](#) SOCKET\_TYPE\_STREAM

- #define `PF_INET` 4
- #define `AF_INET` 4
- #define `AF_INET6` 6
- #define `IPPROTO_UDP` 0
- #define `TCP_NODELAY` 0
- #define `SO_LINGER` 1
- #define `SO_TIMEOUT` 2
- #define `SO_BINDADDR` 3
- #define `ETH_STATUS_LINK` 1
- #define `htons(x)` (x)  
*Convert a number to network byte order.*
- #define `ntohs(x)` (x)  
*Convert a number to host byte order.*
- #define `socket(domain, type, protocol)` `syn_socket((type))`  
*Create a network socket for TCP or UDP communication.*
- #define `sendto(socket_num, buffer, length, flags, address, address_length)` `syn_sendto(syn_setDatagramAddress((socket_num),1,(address)),(length),(buffer))`  
*Sends a UDP datagram to a specified address.*
- #define `recvfrom(socket_num, buffer, length, flags, address, address_length)` `syn_recvfrom(syn_setDatagramAddress((socket_num),0,(address)),(length),(buffer))`  
*Receive a UDP datagram.*
- #define `connect(socket_num, address, address_length)` `syn_connect((socket_num),(address))`  
*Connects a TCP socket to a specified address.*
- #define `bind(socket_num, address, address_length)` `syn_bind((socket_num),(address))`  
*Binds a socket to a specified address.*
- #define `syn_listen(socket_num, backlog)` `listen((socket_num),(backlog))`  
*Tells a socket to listen for incoming connections.*
- #define `accept(socket_num, address, address_length)` `syn_accept((socket_num),(address))`  
*Accepts TCP connections on the specified socket.*

- #define `recv(socket_num, buffer, length, flags) syn_recv((socket_num),(length),(buffer))`  
*Reads data from a TCP socket.*
- #define `send(socket_num, buffer, length, flags) syn_send((socket_num),(length),(buffer))`  
*Sends data to a TCP socket.*
- #define `getsockopt(socket_num, level, name, buffer, length) syn_getsockopt((socket_num),(name),(buffer))`  
*Get various socket options.*
- #define `setsockopt(socket_num, level, name, buffer, length) syn_setsockopt((socket_num),(name),(buffer))`  
*Set various socket options.*
- #define `getsockname(socket_num, address, address_length) syn_getsockname((socket_num),(address))`  
*Gets the local IP and port of a socket.*
- #define `getpeername(socket_num, address, address_length) syn_getpeername((socket_num),(address))`  
*Gets the remote address of a connection-based (TCP socket).*
- #define `syn_cleanup(process_id) cleanup((process_id))`  
*Close all sockets and free the parameter buffer associated with a task.*
- #define `syn_avail(socket_num) avail((socket_num))`  
*Reports number of bytes available on a TCP socket.*
- #define `join(socket_num, address, address_length) syn_join((socket_num),(address))`  
*Adds a socket to a specified multicast group.*
- #define `leave(socket_num, address, address_length) syn_leave((socket_num),(address))`  
*Removes a socket from the specified multicast group.*
- #define `syn_getnetworkparams(param_buffer) getnetworkparams((param_buffer))`  
*Get the IPv4 configuration parameters.*



- #define `syn_setnetworkparams`(param\_buffer) setnetworkparams((param\_buffer))  
*Set the IPv4 configuration parameters.*
- #define `syn_getipv6params`(param\_buffer) getipv6params((param\_buffer))  
*Get the IPv6 address.*
- #define `syn_getethernetstatus`() getethernetstatus()  
*Get the ethernet status.*
- #define `gettftpserver`(address, address\_length) syn\_gettftpserver((address))  
*Get the address of the TFTP server.*
- #define `settftpserver`(address, address\_length) syn\_settftpserver((address))  
*Set the address of the TFTP server.*
- #define `syn_version`() sock\_version()  
*Returns the version number of this socket library.*
- #define `arp_generaterequest`(address, address\_length) syn\_arp\_generaterequest((address))  
*Unconditionally generate an ARP request for a given IPv4 address.*
- #define `arp_cacherequest`(address, address\_length) syn\_arp\_cacherequest((address))  
*Generate an ARP request for a given IPv4 address and add to the ARP cache.*
- #define `syn_closesocket`(socket\_num) closesocket((socket\_num))  
*Closes a specific socket.*
- #define `syn_getmacid`() getmacid()  
*Get the pointer to the MAC ID storage area.*
- #define `syn_setmacid`() setmacid()  
*Stores the MAC ID into the MAC ID storage area.*

## Functions

- char \* `inet_ntop` (int family, void \*addr, char \*strptr, `size_t` len)  
*Converts a numeric address to a string.*

- unsigned int [inet\\_pton](#) (int family, char \*str, void \*addr)  
*Converts a string to a numeric IP address.*
- unsigned long [inet\\_addr](#) (char \*inet\_string)  
*Converts a string representing an IPv4 address to numeric form.*
- int [syn\\_socket](#) (unsigned int type)  
*Create a network socket for TCP or UDP communication.*
- int [syn\\_setDatagramAddress](#) (int socket\_num, unsigned char sending, struct [sockaddr](#) \*addr)  
*Set the IP address parameter for future datagram calls.*
- int [syn\\_sendto](#) (int socket\_num, unsigned int length, void \*buffer)  
*Sends a UDP datagram to an address earlier specified by a call to [syn\\_setDatagramAddress](#).*
- int [syn\\_recvfrom](#) (int socket\_num, unsigned int length, void \*buffer)  
*Receive a UDP datagram.*
- int [syn\\_connect](#) (int socket\_num, struct [sockaddr](#) \*address)  
*Connects a TCP socket to a specified address.*
- int [syn\\_bind](#) (int socket\_num, struct [sockaddr](#) \*address)  
*Binds a socket to a specified address.*
- int [listen](#) (int socket\_num, unsigned int backlog)  
*Tells a socket to listen for incoming connections.*
- int [syn\\_accept](#) (int socket\_num, struct [sockaddr](#) \*address)  
*Accepts TCP connections on the specified socket.*
- int [syn\\_recv](#) (int socket\_num, unsigned int length, void \*buffer)  
*Reads data from a TCP socket.*
- int [syn\\_send](#) (int socket\_num, unsigned int length, void \*buffer)  
*Sends data to a TCP socket.*
- int [syn\\_getsockopt](#) (int socket\_num, unsigned int name, void \*buffer)  
*Get various socket options.*
- int [syn\\_setsockopt](#) (int socket\_num, unsigned int name, void \*buffer)  
*Set various socket options.*

- int [syn\\_getsockname](#) (int socket\_num, struct [sockaddr](#) \*address)  
*Gets the local IP and port of a socket.*
- int [syn\\_getpeername](#) (int socket\_num, struct [sockaddr](#) \*address)  
*Gets the remote address of a connection-based (TCP socket).*
- int [cleanup](#) (unsigned int process\_id)  
*Close all sockets and free the parameter buffer associated with a task.*
- int [avail](#) (int socket\_num)  
*Reports number of bytes available on a TCP socket.*
- int [syn\\_join](#) (int socket\_num, struct [sockaddr](#) \*address)  
*Adds a socket to a specified multicast group.*
- int [syn\\_leave](#) (int socket\_num, struct [sockaddr](#) \*address)  
*Removes a socket from the specified multicast group.*
- int [getnetworkparams](#) (void \*param\_buffer)  
*Get the IPv4 configuration parameters.*
- int [setnetworkparams](#) (void \*param\_buffer)  
*Set the IPv4 configuration parameters.*
- int [getipv6params](#) (void \*param\_buffer)  
*Get the IPv6 address.*
- unsigned int [getethernetstatus](#) (void)  
*Get the ethernet status.*
- int [syn\\_gettftpserver](#) (struct [sockaddr](#) \*address)  
*Get the address of the TFTP server.*
- int [syn\\_settftpserver](#) (struct [sockaddr](#) \*address)  
*Set the address of the TFTP server.*
- void [clear\\_param\\_buffers](#) (void)  
*Clears the parameter buffers used by the socket library.*
- unsigned int [sock\\_version](#) (void)  
*Returns the version number of this socket library.*

- int [syn\\_arp\\_generaterequest](#) (struct [sockaddr](#) \*address)  
*Generate an ARP request for a given IPv4 address.*
- int [syn\\_arp\\_cacherequest](#) (struct [sockaddr](#) \*address)  
*Generate an ARP request for a given IPv4 address and add to the ARP cache.*
- int [acceptqueue](#) (int socket\_handle, struct [sockaddr](#) \*address)  
*Returns the number of sockets in the wait queue for this listening socket.*
- int [udpavailable](#) (int socket\_handle, struct [sockaddr](#) \*address)  
*Returns whether or not data is available to be read on a datagram socket.*
- int [closesocket](#) (int socket\_num)  
*Closes a specific socket.*
- unsigned char \* [getmacid](#) (void)  
*Get the pointer to the MAC ID storage area.*
- void [setmacid](#) (void)  
*Stores the MAC ID into the MAC ID storage area.*
- long [ping](#) (struct [sockaddr](#) \*address, unsigned int address\_length, unsigned int time\_to\_live, struct [pingdata](#) \*response)  
*Pings the specified address.*
- unsigned int [eth\\_readmii](#) (unsigned int phy, unsigned int reg)  
*Read a PHY register via MII.*
- void [eth\\_writemii](#) (unsigned int phy, unsigned int reg, unsigned int val)  
*Write a PHY register via MII.*
- void [eth\\_disablemulticastreceiver](#) (void)  
*Disable multicast hardware receiver.*
- int [unbind](#) (int socket\_num)  
*Unbind a bound socket.*
- int [setsockowner](#) (int socket\_num, unsigned int process\_id)  
*Sets the socket's owner to a different task ID.*
- unsigned long [eth\\_readcsr](#) (unsigned int reg)

*Read a MAC CSR register.*

- void [eth\\_writecsr](#) (unsigned int reg, unsigned long val)

*Write a MAC CSR register.*

### 8.13.2 Define Documentation

#### 8.13.2.1 **#define accept(socket\_num, address, address\_length) syn\_accept((socket\_num),(address))**

Accepts TCP connections on the specified socket.

Accepts a TCP connection on the specified socket. This function moves the first pending connection request from the listen queue into the established state, assigning a new local socket to the connection for communication. [accept](#) blocks if there are no pending incoming requests. The socket *socket\_num* must have been created with type [SOCKET\\_TYPE\\_STREAM](#), bound to an address using [bind](#), and given a listen queue by calling [listen](#).

#### Parameters:

*socket\_num* the handle of the socket that will wait for connections

*address* location to write remote address

*address\_length* the length of the address structure (ignored)

#### Returns:

New socket handle for communicating with remote socket, or 0xFFFF for failure

#### See also:

[socket](#)

[bind](#)

[listen](#)

#### 8.13.2.2 **#define AF\_INET 4**

IPv4 family define, ignored by DS80C400 Silicon Software, but included for compatibility

#### 8.13.2.3 **#define AF\_INET6 6**

IPv6 family define, ignored by DS80C400 Silicon Software, but included for compatibility

#### 8.13.2.4 **#define arp\_cacherequest(address, address\_length) syn\_arp\_cacherequest((address))**

Generate an ARP request for a given IPv4 address and add to the ARP cache.

If the given IP address is not in the ARP cache, generate an ARP request and add it to the cache.

##### **Parameters:**

*address* structure to store the address

*address\_length* the length of the address structure (ignored)

##### **Returns:**

0 for success, non-zero for failure

#### 8.13.2.5 **#define arp\_generaterequest(address, address\_length) syn\_arp\_generaterequest((address))**

Unconditionally generate an ARP request for a given IPv4 address.

Unconditionally generate an ARP request for a given IPv4 address. This functionality can be used to implement Zeroconf protocols.

##### **Parameters:**

*address* structure to store the address

*address\_length* the length of the address structure (ignored)

##### **Returns:**

0 for success, non-zero for failure

#### 8.13.2.6 **#define bind(socket\_num, address, address\_length) syn\_bind((socket\_num),(address))**

Binds a socket to a specified address.

Assigns a local address and port (stored in the *address* parameter) to a socket. Binding a socket is necessary for server sockets. For client sockets, use *bind* if a specific source port is desirable.

Fill *address* with 0's (for *sin\_addr* and *sin\_port*) to bind to any available local port. Use *getsockname* to discover which port the socket was bound to.

**NOTE:** When binding a UDP socket, matching inbound UDP packets will be queued up for the socket. Call *recvfrom* periodically to avoid the risk of running out of kernel memory.

**Parameters:**

*socket\_num* socket handle to bind to a local port number  
*address* contains the local address (including port number)  
*address\_length* the length of the address structure (ignored)

**Returns:**

0 for success, non-zero for failure.

**See also:**

[listen](#)  
[getsockname](#)  
[recvfrom](#)

**8.13.2.7 #define connect(socket\_num, address, address\_length) syn-connect((socket\_num),(address))**

Connects a TCP socket to a specified address.

Connects to a specified address with a streaming socket. This function can only be used once with each socket. The socket *socket\_num* must have been created with type [SOCKET\\_TYPE\\_STREAM](#).

**Parameters:**

*socket\_num* the socket handle to use to wait for and read a UDP packet  
*address* IP address and port number to create a streaming connection to  
*address\_length* the length of the address structure (ignored)

**Returns:**

0 for success, non-zero for failure.

**See also:**

[socket](#)

**8.13.2.8 #define ETH\_STATUS\_LINK 1**

Flag for analyzing ethernet status.

**See also:**

[getethernetstatus](#)

**8.13.2.9 #define getpeername(socket\_num, address, address\_length) syn-getpeername((socket\_num),(address))**

Gets the remote address of a connection-based (TCP socket).

Stores the IP address of the remote socket communicating with the socket specified by *socket\_num*. Use [getsockname](#) to get the local port's information.

**Parameters:**

*socket\_num* handle of the socket to get remote IP and port for

*address* structure where IP and port will be stored

*address\_length* the length of the address structure (ignored)

**Returns:**

0 for success, non-zero for failure

**See also:**

[getsockname](#)

**8.13.2.10 #define getsockname(socket\_num, address, address\_length) syn-getsockname((socket\_num),(address))**

Gets the local IP and port of a socket.

Stores the local IP and port number of the specified socket in the *address* parameter. Use [getpeername](#) to get the remote port's information for a connection-based (TCP) socket.

**Parameters:**

*socket\_num* handle of the socket to get local IP and port for

*address* structure where IP and port will be stored

*address\_length* the length of the address structure (ignored)

**Returns:**

0 for success, non-zero for failure

**See also:**

[getpeername](#)

**8.13.2.11 #define getsockopt(socket\_num, level, name, buffer, length) syn-getsockopt((socket\_num),(name),(buffer))**

Get various socket options.



Reads a number of supported socket options. Data written into the buffer depends on the requested socket option.

Name	Description	Data in buffer
TCP_NODELAY	TCP Nagle Enable	1 byte
SO_LINGER	Ignored	N/A
SO_TIMEOUT	Inactivity timeout	4 bytes (milliseconds, MSB first)
SO_BINDADDR	Local socket IP	16 bytes

**Parameters:**

*socket\_num* socket to get option information for  
*level* ignored  
*name* option to get  
*buffer* location where option data will be written  
*length* length of the buffer

**Returns:**

0 for success, non-zero for failure

**See also:**

[setsockopt](#)

**8.13.2.12 #define gettftpserver(address, address\_length) syn - gettftpserver((address))**

Get the address of the TFTP server.

Returns the address of the server accessed by the TFTP functions. To communicate with a TFTP server, use the functions listed in [rom400\\_tftp.h](#), the TFTP library.

**Parameters:**

*address* structure to store the address of the TFTP server  
*address\_length* the length of the address structure (ignored)

**Returns:**

0 for success, non-zero for failure

**See also:**

[settftpserver](#)

#### 8.13.2.13 **#define htons(x) (x)**

Convert a number to network byte order.

Converts a word from host byte order to network byte order. On the DS80C400, the orders are the same, so this function does not alter the input data. This function is included for compatibility.

##### **Parameters:**

*x* Input data to convert to network byte order

##### **Returns:**

Input data converted to network byte order

#### 8.13.2.14 **#define IPPROTO\_UDP 0**

Protocol ID define, ignored by DS80C400 Silicon Software, but included for compatibility

#### 8.13.2.15 **#define join(socket\_num, address, address\_length) syn\_join((socket\_num),(address))**

Adds a socket to a specified multicast group.

Adds a UDP socket to a specified multicast group. In order to receive multicasts from a group, first [bind](#) the socket to the port number that the multicast group is using (it is not sufficient to include it here in order to receive).

Use the [leave](#) function to leave a multicast group.

##### **Warning:**

IPv6 multicasting is not supported

##### **Parameters:**

*socket\_num* handle for the datagram socket that will join a multicast group

*address* IP address of the multicast group to join

*address\_length* the length of the address structure (ignored)

##### **Returns:**

0 for success, non-zero for failure.

##### **See also:**

[leave](#)

**8.13.2.16 #define leave(socket\_num, address, address\_length) syn\_ - leave((socket\_num),(address))**

Removes a socket from the specified multicast group.

Removes a UDP socket from the specified multicast group.

**Parameters:**

*socket\_num* handle for the datagram socket that will leave a multicast group

*address* IP address of the multicast group to leave

*address\_length* the length of the address structure (ignored)

**Returns:**

0 for success, non-zero for failure.

**See also:**

[join](#)

**8.13.2.17 #define ntohs(x) (x)**

Convert a number to host byte order.

Converts a word from network byte order to host byte order. On the DS80C400, the orders are the same, so this function does not alter the input data. This function is included for compatibility.

**Parameters:**

*x* Input data to convert to network byte order

**Returns:**

Input data converted to network byte order

**8.13.2.18 #define PF\_INET 4**

IPv4 protocol family define

**8.13.2.19 #define recv(socket\_num, buffer, length, flags) syn\_recv((socket\_num),(length),(buffer))**

Reads data from a TCP socket.

Reads data from a TCP socket. If there is no data available, [recv](#) blocks until there is data, subject to the value of [SO\\_TIMEOUT](#). **NOTE:** This function reads **up to** *length* bytes. Call this function repeatedly if you need to read a minimum number of bytes.

**Parameters:**

*socket\_num* handle of the streaming socket that will read data

*buffer* location to write any data read

*length* maximum amount of data to read

*flags* ignored

**Returns:**

The number of bytes read. If the operation times out according to the *SO\_TIMEOUT*, a value of -2 is returned. If another error occurs, -1 is returned. If the socket was closed by the other side, 0 is returned.

**See also:**

[connect](#)

[send](#)

**8.13.2.20** `#define recvfrom(socket_num, buffer, length, flags, address, address_length) syn_recvfrom(syn_setDatagramAddress((socket_num),0,(address)),(length),(buffer))`

Receive a UDP datagram.

Receives a message on the specified socket, and stores the address that sent it. If no data is available, [recvfrom](#) blocks subject to the *SO\_TIMEOUT* value. The socket *socket\_num* must have been created with a type *SOCKET\_TYPE\_DATAGRAM*. It is required to use [bind](#) to assign a local port to the socket, before receiving data. **NOTE:** This function reads **up to** *length* bytes of a datagram. Any data not read in the datagram will be discarded.

**Parameters:**

*socket\_num* the socket handle to use to wait for and read a UDP packet

*buffer* the location to write any data read from the datagram socket

*length* the maximum number of bytes to read from a datagram socket

*flags* ignored

*address* location to fill in the address and port of the sender

*address\_length* the length of the address structure (ignored)

**Returns:**

The number of bytes read. If the operation times out according to the *SO\_TIMEOUT*, a value of -2 is returned. If another error occurs, -1 is returned.

**See also:**

[sendto](#)

[socket](#)

[bind](#)

#### 8.13.2.21 **#define ROM400 SOCK SYNCH\_VERSION ROM400 SOCK - VERSION**

Version number associated with this header file. Should be the same as the version number returned by the [sock\\_version](#) function.

See also:

[syn\\_version](#)

#### 8.13.2.22 **#define ROM400 SOCK\_VERSION 12**

Version number associated with this header file. Should be the same as the version number returned by the [sock\\_version](#) function.

See also:

[sock\\_version](#)

#### 8.13.2.23 **#define send(socket\_num, buffer, length, flags) syn\_send((socket\_num),(length),(buffer))**

Sends data to a TCP socket.

Writes data to a TCP socket. The return value of this function is only a local success/failure code, and may not necessarily detect transmission errors.

**Parameters:**

*socket\_num* handle of the streaming socket that will write data

*buffer* location of data to write

*length* number of bytes to write

*flags* ignored

**Returns:**

0 for success, non-zero for failure.

See also:

[connect](#)

[recv](#)

#### 8.13.2.24 **#define sendto(socket\_num, buffer, length, flags, address, address\_length) syn\_sendto(syn\_setDatagramAddress((socket\_num),1,(address)),(length),(buffer))**

Sends a UDP datagram to a specified address.

Sends a UDP datagram to a specified address. The success/failure code this function returns says nothing of if the packet was recieved by the target, only that the socket layer was able to push the data out. The socket *socket\_num* must have been created with a type [SOCKET\\_TYPE\\_DATAGRAM](#).

**Parameters:**

*socket\_num* the socket handle to use to send a UDP packet  
*buffer* the data to send in the datagram packet  
*length* the number of bytes to send in the datagram packet  
*flags* ignored  
*address* the destination address and port  
*address\_length* the length of the address structure (ignored)

**Returns:**

0 for success, non-zero for failure.

**See also:**

[recvfrom](#)  
[socket](#)

**8.13.2.25 #define setsockopt(socket\_num, level, name, buffer, length) syn-setsockopt((socket\_num),(name),(buffer))**

Set various socket options.

Sets a number of supported socket options. Input data in the buffer depends on the desired socket option.

Name	Description	Data in buffer
TCP_NODELAY	TCP Nagle Enable	1 byte
SO_LINGER	Ignored	N/A
SO_TIMEOUT	Inactivity timeout	4 bytes (milliseconds, MSB first)
SO_BINDADDR	Read only	N/A

**Parameters:**

*socket\_num* socket to set option information for  
*level* ignored  
*name* option to set  
*buffer* location of option data that will be written  
*length* length of the buffer

**Returns:**

0 for success, non-zero for failure

**See also:**

[getsockopt](#)

**8.13.2.26** `#define settftpserver(address, address_length) syn - settftpserver((address))`

Set the address of the TFTP server.

Set the address of the server that the TFTP functions will use. The [settftpserver](#) function must be used if the address of the TFTP server is not acquired by DHCP or 1-Wire. Once the TFTP server's address is set, use the functions listed in [rom400\\_tftp.h](#) to begin receiving files.

**Parameters:**

*address* structure to store the address of the TFTP server

*address\_length* the length of the address structure (ignored)

**Returns:**

0 for success, non-zero for failure

**See also:**

[gettftpserver](#)

**8.13.2.27** `#define SO_BINDADDR 3`

Argument for socket option. Local binding address.

**See also:**

[getsockopt](#)

[setsockopt](#)

**8.13.2.28** `#define SO_LINGER 1`

Argument for socket option. Ignored by DS80C400 ROM.

**See also:**

[getsockopt](#)

[setsockopt](#)

#### 8.13.2.29 **#define SO\_TIMEOUT 2**

Argument for socket option. Socket inactivity timeout.

**See also:**

[getsockopt](#)  
[setsockopt](#)

#### 8.13.2.30 **#define SOCK\_DGRAM SOCKET\_TYPE\_DATAGRAM**

Argument to function [socket](#) to create a UDP socket (same as [SOCKET\\_TYPE\\_DATAGRAM](#))

**See also:**

[socket](#)

#### 8.13.2.31 **#define SOCK\_STREAM SOCKET\_TYPE\_STREAM**

Argument to function [socket](#) to create a TCP socket (same as [SOCKET\\_TYPE\\_STREAM](#))

**See also:**

[socket](#)

#### 8.13.2.32 **#define socket(domain, type, protocol) syn\_socket((type))**

Create a network socket for TCP or UDP communication.

Creates a socket for network communication. This function returns a socket handle, but has not specific local address assigned to it. Note that this function calls [task\\_gettaskid](#) through the function redirect table.

**Parameters:**

*domain* ignored

*type* [SOCKET\\_TYPE\\_DATAGRAM](#) or [SOCK\\_DGRAM](#) for UDP, [SOCKET\\_TYPE\\_STREAM](#) or [SOCK\\_STREAM](#) for TCP

*protocol* ignored

**Returns:**

0xFFFF for failure, or the socket handle (socket number)

**See also:**

[bind](#)  
[connect](#)  
[closesocket](#)



#### 8.13.2.33 **#define SOCKET\_TYPE\_DATAGRAM 0**

Argument to function [socket](#) to create a UDP socket (same as [SOCK\\_DGRAM](#))

**See also:**

[socket](#)

#### 8.13.2.34 **#define SOCKET\_TYPE\_STREAM 1**

Argument to function [socket](#) to create a TCP socket (same as [SOCK\\_STREAM](#))

**See also:**

[socket](#)

#### 8.13.2.35 **#define syn\_avail(socket\_num) avail((socket\_num))**

Reports number of bytes available on a TCP socket.

Reports the number of bytes available on a TCP socket. This is the number of bytes that can currently be read using the [recv](#) function without blocking.

**Parameters:**

*socket\_num* the handle of the socket to check for available data

**Returns:**

The number of bytes available for a [recv](#) function call on this socket, or 0xFFFF on failure.

**See also:**

[recv](#)

#### 8.13.2.36 **#define syn\_cleanup(process\_id) cleanup((process\_id))**

Close all sockets and free the parameter buffer associated with a task.

Close all sockets associated with a process ID and free the parameter buffer.

User applications should call this function whenever a task dies or is killed to ensure all associated resources are freed by the socket layer.

**Warning:**

The DS80C400 Silicon Software task scheduler does **NOT** call this function. User applications should call [cleanup](#) after each call to [task\\_kill](#).

**Parameters:**

*process\_id* Task PID to clean up sockets associated with, 0 for current process.

**Returns:**

0 for success, non-zero for failure.

**8.13.2.37 #define syn\_closesocket(socket\_num) closesocket((socket\_num))**

Closes a specific socket.

Closes the specified socket that was created using the *socket* function.

**Parameters:**

*socket\_num* the socket handle to close

**Returns:**

0 for success, non-zero for failure.

**See also:**

[socket](#)

**8.13.2.38 #define syn\_getethernetstatus() getethernetstatus()**

Get the ethernet status.

Returns the ethernet status byte. This is a bit-wise OR of the following flags:

Flag	Value	Description
ETH_STATUS_LINK	01h	Ethernet link status

Currently, no other flags are defined.

**Returns:**

Bitmapped ethernet status byte.

**8.13.2.39 #define syn\_getipv6params(param\_buffer) getipv6params((param\_buffer))**

Get the IPv6 address.

Gets the IPv6 address of the ethernet interface. The format for the buffer after this function returns is:

Parameter	Offset	Length	Description
IP6ADDR	0	16	IP address
IP6PREFIX	16	1	IP prefix length

**Parameters:**

*param\_buffer* pointer to buffer to store IPv6 configuration data

**Returns:**

0 for success, non-zero for failure

**See also:**

[getnetworkparams](#)  
[setnetworkparams](#)

**8.13.2.40 #define syn\_getmacid() getmacid()**

Get the pointer to the MAC ID storage area.

Returns the pointer to the MAC ID storage area. This area will store the MAC ID after a successful call to [setmacid](#).

**Returns:**

Pointer to the 400's MAC ID (6 bytes stored at this location)

**See also:**

[setmacid](#)

**8.13.2.41 #define syn\_getnetworkparams(param\_buffer) getnetworkparams((param\_buffer))**

Get the IPv4 configuration parameters.

Get the IPv4 configuration parameters, including IP address, subnet mask, and gateway. The parameters are returned in a buffer in the following form:

Parameter	Offset	Length	Description
(zero)	0	12	Must be 0
IP4ADDR	12	4	IP address
IP4SUBNET	16	4	Subnet mask
IP4PREFIX	20	1	Number of 1 bits in subnet mask
(zero)	21	12	Must be 0
IP4GATEWAY	33	4	Gateway IP address

IPv6 addresses are autoconfigured. To retrieve the IPv6 address, use the [getipv6params](#) function.

**Parameters:**

*param\_buffer* pointer to buffer to store IP configuration data

**Returns:**

0 for success, non-zero for failure

See also:

[setnetworkparams](#)  
[getipv6params](#)

**8.13.2.42 #define syn\_listen(socket\_num, backlog) listen((socket\_num),(backlog))**

Tells a socket to listen for incoming connections.

Tells the socket to listen for connections. A queue of length *backlog* is created for pending (un-*accepted* connections). It is required to use *bind* to assign a local port before calling *listen*. Use *accept* to move an incoming request to an established state, or wait for incoming connections.

**Parameters:**

*socket\_num* socket handle that will listen for connections

*backlog* the maximum number of pending connections (max 16 for the DS80C400)

**Returns:**

0 for success, non-zero for failure.

See also:

[bind](#)  
[accept](#)

**8.13.2.43 #define syn\_setmacid() setmacid()**

Stores the MAC ID into the MAC ID storage area.

This is a redirected function. The DS80C400's default implementation of this function searches the 1-Wire for a DS2502U-E48 1-Wire chip which contains a MAC ID. This MAC ID is then stored into the MAC ID storage area, the location of which is stored in a pointer in the export table. Use the *getmacid* function to return a pointer to the MAC ID storage area.

See also:

[getmacid](#)

**8.13.2.44 #define syn\_setnetworkparams(param\_buffer) setnetworkparams((param\_buffer))**

Set the IPv4 configuration parameters.

Set the IPv4 configuration parameters, including IP address, subnet mask, and gateway. Input parameters should be formatted in the following form:

Parameter	Offset	Length	Description
(zero)	0	12	Must be 0
IP4ADDR	12	4	IP address
IP4SUBNET	16	4	Subnet mask
IP4PREFIX	20	1	Number of 1 bits in subnet mask
(zero)	21	12	Must be 0
IP4GATEWAY	33	4	Gateway IP address

Use this method to give the DS80C400 a static IP address. To dynamically configure an IP address, use methods from the DHCP library in [rom400\\_dhcp.h](#) (IP addresses leased by the DHCP client can still be retrieved by calling [getnetworkparams](#)).

IPv6 addresses are autoconfigured. To retrieve the IPv6 address, use the [getipv6params](#) function.

**Parameters:**

*param\_buffer* pointer to buffer with IP configuration data

**Returns:**

0 for success, non-zero for failure

**See also:**

[getnetworkparams](#)  
[getipv6params](#)

**8.13.2.45 #define syn\_version() sock\_version()**

Returns the version number of this socket library.

**Returns:**

Version number of this SOCK library.

**8.13.2.46 #define TCP\_NODELAY 0**

Argument for socket option. Enables/disables Nagle algorithm.

**See also:**

[getsockopt](#)  
[setsockopt](#)

### 8.13.3 Function Documentation

#### 8.13.3.1 `int acceptqueue (int socket_handle, struct sockaddr * address)`

Returns the number of sockets in the wait queue for this listening socket.

Returns the number of sockets in the queue attempting to connect to this server socket.

**Parameters:**

*socket\_handle* handle to socket to check for waiting connections

*address* location where the IP and port number will be written

**Returns:**

-1 if the socket is not a streaming socket set up to listen 0 or greater for the number of sockets waiting

The IP and port of the socket are returned in *address*.

#### 8.13.3.2 `int avail (int socket_num)`

Reports number of bytes available on a TCP socket.

Reports the number of bytes available on a TCP socket. This is the number of bytes that can currently be read using the *recv* function without blocking.

**Parameters:**

*socket\_num* the handle of the socket to check for available data

**Returns:**

The number of bytes available for a *recv* function call on this socket, or -1 on failure.

**See also:**

*recv*

#### 8.13.3.3 `int cleanup (unsigned int process_id)`

Close all sockets and free the parameter buffer associated with a task.

Close all sockets associated with a process ID and free the parameter buffer.

User applications should call this function whenever a task dies or is killed to ensure all associated resources are freed by the socket layer.

**Warning:**

The DS80C400 Silicon Software task scheduler does **NOT** call this function. User applications should call *cleanup* after each call to *task\_kill*.

**Parameters:**

*process\_id* Task PID to clean up sockets associated with, 0 for current process.

**Returns:**

0 for success, non-zero for failure.

**See also:**

[setsockowner](#)

**8.13.3.4 void clear\_param\_buffers (void)**

Clears the parameter buffers used by the socket library.

Clears buffers used to store parameters for the socket library. This function should be called immediately after calling the [init\\_rom](#) function, and before any socket library functions are called.

**See also:**

[init\\_rom](#)

**8.13.3.5 int closesocket (int socket\_num)**

Closes a specific socket.

Closes the specified socket that was created using the *socket* function.

**Parameters:**

*socket\_num* the socket handle to close

**Returns:**

0 for success, non-zero for failure.

**See also:**

[socket](#)

**8.13.3.6 void eth\_disablemulticastreceiver (void)**

Disable multicast hardware receiver.

This function disables the "pass multicast" (PM) bit in the DS80C400 MAC control register. This improves performance if the application doesn't use multicast. This function must be called after initialization of the Ethernet. **WARNING:** IPv6 requires multicast. Disabling the receiver disables IPv6 address resolution.

#### 8.13.3.7 unsigned long eth\_readcsr (unsigned int *reg*)

Read a MAC CSR register.

This function reads a MAC CSR register from the DS80C400. See the data sheet and user's guide for more information about the CSR registers.

##### Parameters:

*reg* register address (0 to 0x2c in steps of 4)

##### Returns:

value read from the register specified

##### See also:

[eth\\_writecsr](#)

#### 8.13.3.8 unsigned int eth\_readmii (unsigned int *phy*, unsigned int *reg*)

Read a PHY register via MII.

This function reads a PHY register via the MII interface. See the IEEE 802.3 specification (22.2.4) for a description of the MII management register set.

##### Parameters:

*phy* PHY address (0 to 31)

*reg* register address (0 to 31, 16 through 31 are vendor specific)

##### Returns:

value read from the register specified

#### 8.13.3.9 void eth\_writecsr (unsigned int *reg*, unsigned long *val*)

Write a MAC CSR register.

This function writes a MAC CSR register. See the DS80C400 data sheet and user's guide for more information about the CSR registers.

##### Parameters:

*reg* register address (0 to 0x2c in steps of 4)

*val* value to write to the specified register

##### See also:

[eth\\_readcsr](#)



### 8.13.3.10 void eth\_writemii (unsigned int *phy*, unsigned int *reg*, unsigned int *val*)

Write a PHY register via MII.

This function writes a PHY register via the MII interface. See the IEEE 802.3 specification (22.2.4) for a description of the MII management register set.

#### Parameters:

*phy* PHY address (0 to 31)

*reg* register address (0 to 31, 16 through 31 are vendor specific)

*val* value to write to the specified register

### 8.13.3.11 unsigned int getethernetstatus (void)

Get the ethernet status.

Returns the ethernet status byte. This is a bit-wise OR of the following flags:

Flag	Value	Description
ETH_STATUS_LINK	01h	Ethernet link status

Currently, no other flags are defined.

#### Returns:

Bitmapped ethernet status byte.

### 8.13.3.12 int getipv6params (void \* *param\_buffer*)

Get the IPv6 address.

Gets the IPv6 address of the ethernet interface. The format for the buffer after this function returns is:

Parameter	Offset	Length	Description
IP6ADDR	0	16	IP address
IP6PREFIX	16	1	IP prefix length

#### Parameters:

*param\_buffer* pointer to buffer to store IPv6 configuration data

#### Returns:

0 for success, non-zero for failure

#### See also:

[getnetworkparams](#)

[setnetworkparams](#)

#### 8.13.3.13 unsigned char\* getmacid (void)

Get the pointer to the MAC ID storage area.

Returns the pointer to the MAC ID storage area. This area will store the MAC ID after a successful call to [setmacid](#).

##### Returns:

Pointer to the 400's MAC ID (6 bytes stored at this location)

##### See also:

[setmacid](#)

#### 8.13.3.14 int getnetworkparams (void \* param\_buffer)

Get the IPv4 configuration parameters.

Get the IPv4 configuration parameters, including IP address, subnet mask, and gateway. The parameters are returned in a buffer in the following form:

Parameter	Offset	Length	Description
(zero)	0	12	Must be 0
IP4ADDR	12	4	IP address
IP4SUBNET	16	4	Subnet mask
IP4PREFIX	20	1	Number of 1 bits in subnet mask
(zero)	21	12	Must be 0
IP4GATEWAY	33	4	Gateway IP address

IPv6 addresses are autoconfigured. To retrieve the IPv6 address, use the [getipv6params](#) function.

##### Parameters:

*param\_buffer* pointer to buffer to store IP configuration data

##### Returns:

0 for success, non-zero for failure

##### See also:

[setnetworkparams](#)  
[getipv6params](#)

#### 8.13.3.15 unsigned long inet\_addr (char \* inet\_string)

Converts a string representing an IPv4 address to numeric form.

Converts the input string into an IPv4 address suitable for setting in a [sockaddr\\_in](#) structure.

**Parameters:**

*inet\_string* IPv4 address in string form

**Returns:**

Numeric IPv4 address

**See also:**

[sockaddr\\_in](#)

**8.13.3.16 char\* inet\_ntop (int *family*, void \* *addr*, char \* *strptr*, [size\\_t](#) *len*)**

Converts a numeric address to a string.

Converts a numeric IP address to a presentable format as a null terminated string. IPv4 addresses are formatted such as in "192.0.1.1". IPv6 addresses are formatted such as in "b803:8a11:0000:2121:fec5:0601:aa01:0102". Note that the '::' shortcut is **not** supported—a '0000' must be fully specified.

**Parameters:**

*family* AF\_INET or AF\_INET6

*addr* pointer to numeric representation of IP address

*strptr* storage location for presentation string

*len* size of storage area for strptr

**Returns:**

Reference to strptr, or NULL if the *family* is not recognized or if there is not enough space as declared by *len*

**See also:**

[inet\\_pton](#)

**8.13.3.17 unsigned int inet\_pton (int *family*, char \* *str*, void \* *addr*)**

Converts a string to a numeric IP address.

Converts a string representation of an IP address into numeric format. IPv4 addresses are expected to be input in a format such as in "192.0.1.1". IPv6 addresses are expected to be formatted such as in "b8:03:8a:11:00:00:21:21:fe:c5:06:01:aa:01:01:02".

**Parameters:**

*family* AF\_INET or AF\_INET6

*str* address string to translate

*addr* pointer to storage for numeric representation of IP address

**Returns:**

1 for successful translation. 0 if the format was invalid, or the *family* was not recognized.

**See also:**

[inet\\_ntop](#)

**8.13.3.18 int listen (int *socket\_num*, unsigned int *backlog*)**

Tells a socket to listen for incoming connections.

Tells the socket to listen for connections. A queue of length *backlog* is created for pending (un-*accepted* connections). It is required to use [bind](#) to assign a local port before calling *listen*. Use [accept](#) to move an incoming request to an established state, or wait for incoming connections.

**Parameters:**

*socket\_num* socket handle that will listen for connections

*backlog* the maximum number of pending connections (max 16 for the DS80C400)

**Returns:**

0 for success, non-zero for failure.

**See also:**

[bind](#)

[accept](#)

**8.13.3.19 long ping (struct [sockaddr](#) \* *address*, unsigned int *address\_length*, unsigned int *time\_to\_live*, struct [pingdata](#) \* *response*)**

Pings the specified address.

Sends an ICMP echo request (ping) to a specified address. Note that this function is **NOT** safe to be called from multiple processes at the same time.

**Parameters:**

*address* IP address to send an ICMP echo request to

*address\_length* the length of the address structure (ignored)

*time\_to\_live* packets send by ping have this "time to live" setting

*response* data structure to fill in returned data (this argument must not be NULL)

**Returns:**

response time in milliseconds (0 means less than 1ms), or -1L for failure

The ping return data structure is defined as follows: reserved - Reserved field ip\_header - The IP header of the return packet icmp\_header - The ICMP header of the return packet icmp\_data - The ICMP data portion of the return packet (should be 0x20,0x21,0x22,...,0x3f)

#### 8.13.3.20 void setmacid (void)

Stores the MAC ID into the MAC ID storage area.

This is a redirected function. The DS80C400's default implementation of this function searches the 1-Wire for a DS2502U-E48 1-Wire chip which contains a MAC ID. This MAC ID is then stored into the MAC ID storage area, the location of which is stored in a pointer in the export table. Use the [getmacid](#) function to return a pointer to the MAC ID storage area.

See also:

[getmacid](#)

#### 8.13.3.21 int setnetworkparams (void \* *param\_buffer*)

Set the IPv4 configuration parameters.

Set the IPv4 configuration parameters, including IP address, subnet mask, and gateway. Input parameters should be formatted in the following form:

Parameter	Offset	Length	Description
(zero)	0	12	Must be 0
IP4ADDR	12	4	IP address
IP4SUBNET	16	4	Subnet mask
IP4PREFIX	20	1	Number of 1 bits in subnet mask
(zero)	21	12	Must be 0
IP4GATEWAY	33	4	Gateway IP address

Use this method to give the DS80C400 a static IP address. To dynamically configure an IP address, use methods from the DHCP library in [rom400\\_dhcp.h](#) (IP addresses leased by the DHCP client can still be retrieved by calling [getnetworkparams](#)).

IPv6 addresses are autoconfigured. To retrieve the IPv6 address, use the [getip6params](#) function.

**Parameters:**

*param\_buffer* pointer to buffer with IP configuration data

**Returns:**

0 for success, non-zero for failure

**See also:**

[getnetworkparams](#)  
[getip6params](#)

#### **8.13.3.22 int setsockowner (int *socket\_num*, unsigned int *process\_id*)**

Sets the socket's owner to a different task ID.

Sets the socket owner to a different task ID. This is useful where program code relies on [cleanup](#) to deallocate a process' resources, or in cases where ownership of a socket needs to be moved to a child process. Note that the new process ID is not checked for validity and it is possible to assign a socket to a non-existent task.

**Parameters:**

*socket\_num* socket handle  
*process\_id* the task PID of the new socket owner

**Returns:**

0 for success, non-zero for failure.

**See also:**

[cleanup](#)

#### **8.13.3.23 unsigned int sock\_version (void)**

Returns the version number of this socket library.

**Returns:**

Version number of this SOCK library.

#### **8.13.3.24 int syn\_accept (int *socket\_num*, struct [sockaddr](#) \* *address*)**

Accepts TCP connections on the specified socket.

Accepts a TCP connection on the specified socket. This function moves the first pending connection request from the listen queue into the established state, assigning a new local socket to the connection for communication. [accept](#) blocks if there are no pending incoming requests. The socket *socket\_num* must have been created with type [SOCKET\\_TYPE\\_STREAM](#), bound to an address using [bind](#), and given a listen queue by calling [listen](#).

**Parameters:**

*socket\_num* the handle of the socket that will wait for connections

*address* location to write remote address

**Returns:**

New socket handle for communicating with remote socket, or -1 for failure

**See also:**

[socket](#)

[bind](#)

[listen](#)

**8.13.3.25 int syn\_arp\_cacherequest (struct [sockaddr](#) \* *address*)**

Generate an ARP request for a given IPv4 address and add to the ARP cache.

If the given IP address is not in the ARP cache, generate an ARP request and add it to the cache.

**Parameters:**

*address* structure to store the address

**Returns:**

0 for success, non-zero for failure

**8.13.3.26 int syn\_arp\_generaterequest (struct [sockaddr](#) \* *address*)**

Generate an ARP request for a given IPv4 address.

Unconditionally generate an ARP request for a given IPv4 address. This functionality can be used to implement Zeroconf protocols.

**Parameters:**

*address* structure to store the address

**Returns:**

0 for success, non-zero for failure

**8.13.3.27 int syn\_bind (int *socket\_num*, struct [sockaddr](#) \* *address*)**

Binds a socket to a specified address.

Assigns a local address and port (stored in the *address* parameter) to a socket. Binding a socket is necessary for server sockets. For client sockets, use [bind](#) if a specific source port is desirable.

Fill *address* with 0's (for *sin\_addr* and *sin\_port*) to bind to any available local port. Use [getsockname](#) to discover which port the socket was bound to.

**NOTE:** When binding a UDP socket, matching inbound UDP packets will be queued up for the socket. Call [recvfrom](#) periodically to avoid the risk of running out of kernel memory.

**Parameters:**

*socket\_num* socket handle to bind to a local port number

*address* contains the local address (including port number)

**Returns:**

0 for success, non-zero for failure.

**See also:**

[listen](#)

[getsockname](#)

[recvfrom](#)

[unbind](#)

**8.13.3.28 int syn\_connect (int *socket\_num*, struct [sockaddr](#) \* *address*)**

Connects a TCP socket to a specified address.

Connects to a specified address with a streaming socket. This function can only be used once with each socket. The socket *socket\_num* must have been created with type [SOCKET\\_TYPE\\_STREAM](#).

**Parameters:**

*socket\_num* the socket handle to use to wait for and read a UDP packet

*address* IP address and port number to create a streaming connection to

**Returns:**

0 for success, non-zero for failure.

**See also:**

[socket](#)

**8.13.3.29 int syn\_getpeername (int *socket\_num*, struct [sockaddr](#) \* *address*)**

Gets the remote address of a connection-based (TCP socket).

Stores the IP address of the remote socket communicating with the socket specified by *socket\_num*. Use [getsockname](#) to get the local port's information.

**Parameters:**

*socket\_num* handle of the socket to get remote IP and port for



*address* structure where IP and port will be stored

**Returns:**

0 for success, non-zero for failure

**See also:**

[getsockname](#)

**8.13.3.30 int syn\_getsockname (int *socket\_num*, struct [sockaddr](#) \* *address*)**

Gets the local IP and port of a socket.

Stores the local IP and port number of the specified socket in the the *address* parameter. Use [getpeername](#) to get the remote port's information for a connection-based (TCP) socket.

**Parameters:**

*socket\_num* handle of the socket to get local IP and port for

*address* structure where IP and port will be stored

**Returns:**

0 for success, non-zero for failure

**See also:**

[getpeername](#)

**8.13.3.31 int syn\_getsockopt (int *socket\_num*, unsigned int *name*, void \* *buffer*)**

Get various socket options.

Reads a number of supported socket options. Data written into the buffer depends on the requested socket option.

Name	Description	Data in buffer
TCP_NODELAY	TCP Nagle Enable	1 byte
SO_LINGER	Ignored	N/A
SO_TIMEOUT	Inactivity timeout	4 bytes (milliseconds, MSB first)
SO_BINDADDR	Local socket IP	16 bytes

This function assumes there is enough room in *buffer* to store the requested data.

**Parameters:**

*socket\_num* socket to get option information for

*name* option to get

*buffer* location where option data will be written

**Returns:**

0 for success, non-zero for failure

**See also:**

[setsockopt](#)

**8.13.3.32 int syn\_gettftpserver (struct [sockaddr](#) \* *address*)**

Get the address of the TFTP server.

Returns the address of the server accessed by the TFTP functions. To communicate with a TFTP server, use the functions listed in [rom400\\_tftp.h](#), the TFTP library.

**Parameters:**

*address* structure to store the address of the TFTP server

**Returns:**

0 for success, non-zero for failure

**See also:**

[settftpserver](#)

**8.13.3.33 int syn\_join (int *socket\_num*, struct [sockaddr](#) \* *address*)**

Adds a socket to a specified multicast group.

Adds a UDP socket to a specified multicast group. In order to receive multicasts from a group, first [bind](#) the socket to the port number that the multicast group is using (it is not sufficient to include it here in order to receive).

Use the [leave](#) function to leave a multicast group.

**Warning:**

IPv6 multicasting is not supported

**Parameters:**

*socket\_num* handle for the datagram socket that will join a multicast group

*address* IP address of the multicast group to join

**Returns:**

0 for success, non-zero for failure.

**See also:**

[leave](#)

#### 8.13.3.34 `int syn_leave (int socket_num, struct sockaddr * address)`

Removes a socket from the specified multicast group.

Removes a UDP socket from the specified multicast group.

##### Parameters:

*socket\_num* handle for the datagram socket that will leave a multicast group

*address* IP address of the multicast group to leave

##### Returns:

0 for success, non-zero for failure.

##### See also:

[join](#)

#### 8.13.3.35 `int syn_recv (int socket_num, unsigned int length, void * buffer)`

Reads data from a TCP socket.

Reads data from a TCP socket. If there is no data available, [recv](#) blocks until there is data, subject to the value of [SO\\_TIMEOUT](#). **NOTE:** This function reads **up to** *length* bytes. Call this function repeatedly if you need to read a minimum number of bytes.

##### Parameters:

*socket\_num* handle of the streaming socket that will read data

*length* maximum amount of data to read

*buffer* location to write any data read

##### Returns:

The number of bytes read. If the operation times out according to the [SO\\_TIMEOUT](#), a value of -2 is returned. If another error occurs, -1 is returned. If the socket was closed by the other side, 0 is returned.

##### See also:

[connect](#)

[send](#)

#### 8.13.3.36 `int syn_recvfrom (int socket_num, unsigned int length, void * buffer)`

Receive a UDP datagram.

Receives a message on the specified socket, and stores the address that sent it in the address structure set by an earlier call to [syn\\_setDatagramAddress](#). If no data is available, [syn\\_recvfrom](#) blocks subject to the [SO\\_TIMEOUT](#) value. The socket *socket\_num*

must have been created with a type [SOCKET\\_TYPE\\_DATAGRAM](#). It is required to use [syn\\_bind](#) to assign a local port to the socket, before receiving data. **NOTE:** This function reads **up to** *length* bytes of a datagram. Any data not read in the datagram will be discarded.

**Parameters:**

*socket\_num* the socket handle to use to wait for and read a UDP packet

*length* the maximum number of bytes to read from a datagram socket

*buffer* the location to write any data read from the datagram socket

**Returns:**

The number of bytes read. If the operation times out according to the *SO\_TIMEOUT*, a value of -2 is returned. If another error occurs, -1 is returned.

**See also:**

[sendto](#)

[socket](#)

[bind](#)

**8.13.3.37 int syn\_send (int *socket\_num*, unsigned int *length*, void \* *buffer*)**

Sends data to a TCP socket.

Writes data to a TCP socket. The return value of this function is only a local success/failure code, and may not necessarily detect transmission errors.

**Parameters:**

*socket\_num* handle of the streaming socket that will write data

*length* number of bytes to write

*buffer* location of data to write

**Returns:**

0 for success, non-zero for failure.

**See also:**

[connect](#)

[recv](#)

**8.13.3.38 int syn\_sendto (int *socket\_num*, unsigned int *length*, void \* *buffer*)**

Sends a UDP datagram to an address earlier specified by a call to [syn\\_setDatagram-Address](#).

Sends a UDP datagram to an address earlier specified by a call to [syn\\_setDatagramAddress](#). The success/failure code this function returns says nothing of if the packet was recieved by the target, only that the socket layer was able to push the data out. The socket *socket\_num* must have been created with a type [SOCKET\\_TYPE\\_DATAGRAM](#).

**Parameters:**

*socket\_num* the socket handle to use to send a UDP packet

*length* the number of bytes to send in the datagram packet

*buffer* the data to send in the datagram packet

**Returns:**

0 for success, non-zero for failure.

**See also:**

[recvfrom](#)

[socket](#)

[syn\\_setDatagramAddress](#)

**8.13.3.39 int syn\_setDatagramAddress (int *socket\_num*, unsigned char *sending*, struct [sockaddr](#) \* *addr*)**

Set the IP address parameter for future datagram calls.

In order to keep the functions in this library multi-process-safe, datagram functions [syn\\_sendto](#) and [syn\\_recvfrom](#) cannot have as many parameters as their traditional counterparts. This function sets the pointer to the address structure that will be used as the address parameter for functions [syn\\_sendto](#) and [syn\\_recvfrom](#).

Note that the Berkeley style API is now supported and is multi-process safe, so user software should never have to call this function.

**Parameters:**

*socket\_num* Socket number to set address for

*sending* Set to 0 if this is an address for receiving, Set to 1 if this is an address for sending

*addr* Address structure that will be used in future calls to [syn\\_sendto](#) or [syn\\_recvfrom](#).

**Returns:**

socket\_num (for Macro purposes)

**See also:**

[syn\\_sendto](#)

[syn\\_recvfrom](#)

#### 8.13.3.40 `int syn_setsockopt (int socket_num, unsigned int name, void * buffer)`

Set various socket options.

Sets a number of supported socket options. Input data in the buffer depends on the desired socket option.

Name	Description	Data in buffer
TCP_NODELAY	TCP Nagle Enable	1 byte
SO_LINGER	Ignored	N/A
SO_TIMEOUT	Inactivity timeout	4 bytes (milliseconds, MSB first)
SO_BINDADDR	Read only	N/A

**Parameters:**

*socket\_num* socket to set option information for

*name* option to set

*buffer* location of option data that will be written

**Returns:**

0 for success, non-zero for failure

**See also:**

[getsockopt](#)

#### 8.13.3.41 `int syn_settftpserver (struct sockaddr * address)`

Set the address of the TFTP server.

Set the address of the server that the TFTP functions will use. The [settftpserver](#) function must be used if the address of the TFTP server is not acquired by DHCP or 1-Wire. Once the TFTP server's address is set, use the functions listed in [rom400\\_tftp.h](#) to begin receiving files.

**Parameters:**

*address* structure to store the address of the TFTP server

**Returns:**

0 for success, non-zero for failure

**See also:**

[gettftpserver](#)

#### 8.13.3.42 **int syn\_socket (unsigned int *type*)**

Create a network socket for TCP or UDP communication.

Creates a socket for network communication. This function returns a socket handle, but has not specific local address assigned to it. Note that this function calls [task\\_gettaskid](#) through the function redirect table.

**Parameters:**

*type* [SOCKET\\_TYPE\\_DATAGRAM](#) or [SOCK\\_DGRAM](#) for UDP, [SOCKET\\_TYPE\\_STREAM](#) or [SOCK\\_STREAM](#) for TCP

**Returns:**

-1 for failure, or the socket handle (socket number)

**See also:**

[bind](#)  
[connect](#)  
[closesocket](#)

#### 8.13.3.43 **int udpavailable (int *socket\_handle*, struct [sockaddr](#) \* *address*)**

Returns whether or not data is available to be read on a datagram socket.

Returns **1** if there is data available to be read on a UDP socket.

**Parameters:**

*socket\_handle* handle to socket to check for available datagrams

*address* location where the IP and port number will be written

**Returns:**

-1 if the socket is not a datagram socket 0 if no datagram packets are available 1 if a datagram is available

The IP and port of the socket are returned in *address*.

#### 8.13.3.44 **int unbind (int *socket\_num*)**

Unbind a bound socket.

Removes a local address and port from a socket that was assigned to it using [bind](#).

**Parameters:**

*socket\_num* socket handle

**Returns:**

0 for success, non-zero for failure.

**See also:**

[bind](#)

## 8.14 rom400\_task.h File Reference

### 8.14.1 Detailed Description

Process scheduler functions in the DS80C400 ROM.

This library contains functions for starting, suspending, killing, and managing tasks using the ROM's process scheduler.

For detailed information on the DS80C400 please see the [High-Speed Microcontroller User's Guide: DS80C400 Supplement](#).

#### Warning:

Some functions in this library are **NOT** multi-process safe—that is, if you call the same method from two different processes at the same time, the parameters to the function may be destroyed, yielding unpredictable results. Consult each individual function's documentation for details on which functions are multi-process safe.

#### Data Structures

- struct [TIME](#)
- struct [FARPTR](#)
- struct [TCB](#)

#### Defines

- #define [ROM400\\_TASK\\_VERSION](#) 9
- #define [ROM400\\_SCHED\\_VERSION](#) ROM400\_TASK\_VERSION  
*Included for legacy reasons. Please use [ROM400\\_TASK\\_VERSION](#) instead.*
- #define [RELOAD\\_14\\_746](#) 0xfb33
- #define [RELOAD\\_18\\_432](#) 0xfa00
- #define [RELOAD\\_29\\_491](#) 0xfd99
- #define [RELOAD\\_36\\_864](#) 0xfd00
- #define [RELOAD\\_58\\_982](#) 0xfecc
- #define [RELOAD\\_73\\_728](#) 0xfe80
- #define [MIN\\_PRIORITY](#) 1
- #define [NORM\\_PRIORITY](#) 128
- #define [MAX\\_PRIORITY](#) 255
- #define [FLAG\\_SLEEPING](#) 1
- #define [FLAG\\_IO\\_WAIT](#) 2
- #define [FLAG\\_DHCP\\_WAIT](#) 4
- #define [FLAG\\_USER0](#) 8
- #define [FLAG\\_USER1](#) 0x10



- #define `FLAG_USER2` 0x20
- #define `FLAG_USER3` 0x40
- #define `FLAG_USER4` 0x80
- #define `ROM_SAVESIZE` 384

## Functions

- void `task_genesis` (unsigned int savesize)  
*Initializes the process scheduler.*
- unsigned char `task_getcurrent` (void)  
*Gets the process ID for the current task.*
- unsigned char `task_getpriority` (unsigned char task\_id)  
*Gets the priority level for the given task.*
- unsigned char `task_setpriority` (unsigned char task\_id, unsigned char priority)  
*Sets the priority level for a given task.*
- unsigned int `task_fork` (unsigned char priority, unsigned int savesize)  
*Creates a new task.*
- unsigned char `task_kill` (unsigned char task\_id)  
*Kills the specified task.*
- unsigned char `task_suspend` (unsigned char task\_id, unsigned char event\_mask)  
*Suspends the specified task.*
- unsigned char `task_wait` (unsigned char task\_id, unsigned char event\_mask, long millis)  
*Puts the specified task to sleep.*
- unsigned char `task_signal` (unsigned char task\_id, unsigned char event\_mask)  
*Posts events to the specified task.*
- void `task_gettimemillis` (struct `TIME` \*t)  
*Returns the system tick count.*
- unsigned char `task_getthreadid` ()  
*Redirected function to return the current thread's ID number.*

- unsigned char [task\\_threadresume](#) (unsigned char thread, unsigned char task)  
*Redirected function to resume the specified thread.*
- unsigned char [task\\_threadsleep](#) (unsigned char infinite, unsigned long timeout)  
*Redirected function to put the current thread to sleep.*
- unsigned char [task\\_threadsleepnc](#) (unsigned char infinite, unsigned long timeout)  
*Redirected function to put the current thread (which is already in a critical section) to sleep.*
- void [task\\_threadsave](#) (void)  
*Redirected function to save the state of the current thread in anticipation of a task/thread swap.*
- void [task\\_threadrestore](#) (void)  
*Redirected function to restore the state of a thread.*
- unsigned char [task\\_sleep](#) (unsigned char task, long timeout)  
*Redirected function to put a specified task to sleep for a number of milliseconds.*
- unsigned char [task\\_gettaskid](#) ()  
*Redirected function to get the ID of the current task.*
- void [task\\_entercritsection](#) (void)  
*Enters a critical section.*
- void [task\\_leavecritsection](#) (void)  
*Leaves a critical section.*
- unsigned int [task\\_gettickreload](#) (void)  
*Gets the current reload value for the system's millisecond ticker.*
- void [task\\_settickreload](#) (unsigned int reload)  
*Sets the current reload value for the system's millisecond ticker.*
- unsigned int [task\\_version](#) (void)  
*Returns the version number of this process scheduling library.*
- void xdata \* [task\\_reentrant\\_stack](#) (unsigned int size)  
*Reserves space on the reentrant stack.*

## 8.14.2 Define Documentation

### 8.14.2.1 #define FLAG\_DHCP\_WAIT 4

Event flag for putting a task to sleep. Reserved by the system.

See also:

[task\\_wait](#)

### 8.14.2.2 #define FLAG\_IO\_WAIT 2

Event flag for putting a task to sleep. Reserved by the system.

See also:

[task\\_wait](#)

### 8.14.2.3 #define FLAG\_SLEEPING 1

Event flag for putting a task to sleep.

See also:

[task\\_wait](#)

### 8.14.2.4 #define FLAG\_USER0 8

Event flag for putting a task to sleep.

See also:

[task\\_wait](#)

### 8.14.2.5 #define FLAG\_USER1 0x10

Event flag for putting a task to sleep.

See also:

[task\\_wait](#)

### 8.14.2.6 #define FLAG\_USER2 0x20

Event flag for putting a task to sleep.

See also:

[task\\_wait](#)

#### **8.14.2.7 #define FLAG\_USER3 0x40**

Event flag for putting a task to sleep.

**See also:**

[task\\_wait](#)

#### **8.14.2.8 #define FLAG\_USER4 0x80**

Event flag for putting a task to sleep.

**See also:**

[task\\_wait](#)

#### **8.14.2.9 #define MAX\_PRIORITY 255**

Maximum priority level assignable to a task.

**See also:**

[task\\_setpriority](#)

[task\\_getpriority](#)

#### **8.14.2.10 #define MIN\_PRIORITY 1**

Minimum priority level assignable to a task.

**See also:**

[task\\_setpriority](#)

[task\\_getpriority](#)

#### **8.14.2.11 #define NORM\_PRIORITY 128**

Normal priority for a task. This is the default priority for the default task.

**See also:**

[task\\_setpriority](#)

[task\\_getpriority](#)

#### **8.14.2.12 #define RELOAD\_14\_746 0xfb33**

Timer reload value for 14.746 MHz crystal.

**See also:**

[task\\_settickreload](#)

[task\\_gettickreload](#)

#### **8.14.2.13    #define RELOAD\_18\_432 0xfa00**

Timer reload value for 18.432 MHz crystal.

**See also:**

[task\\_settickreload](#)  
[task\\_gettickreload](#)

#### **8.14.2.14    #define RELOAD\_29\_491 0xfd99**

Timer reload value for 29.491 MHz crystal.

**See also:**

[task\\_settickreload](#)  
[task\\_gettickreload](#)

#### **8.14.2.15    #define RELOAD\_36\_864 0xfd00**

Timer reload value for 36.864 MHz crystal.

**See also:**

[task\\_settickreload](#)  
[task\\_gettickreload](#)

#### **8.14.2.16    #define RELOAD\_58\_982 0xfecc**

Timer reload value for 58.982 MHz crystal.

**See also:**

[task\\_settickreload](#)  
[task\\_gettickreload](#)

#### **8.14.2.17    #define RELOAD\_73\_728 0xfe80**

Timer reload value for 73.728 MHz crystal.

**See also:**

[task\\_settickreload](#)  
[task\\_gettickreload](#)

#### 8.14.2.18 **#define ROM400\_TASK\_VERSION 9**

Version number associated with this header file. Should be the same as the version number returned by the [task\\_version](#) function.

**See also:**

[task\\_version](#)

#### 8.14.2.19 **#define ROM\_SAVE\_SIZE 384**

Default size for task switching buffer.

**See also:**

[task\\_genesis](#)

### 8.14.3 **Function Documentation**

#### 8.14.3.1 **void task\_entercritsection (void)**

Enters a critical section.

Enters a critical section, which disallows process swapping until the critical section is left. Calls to [task\\_entercritsection](#) should be balanced with calls to [task\\_leavecritsection](#) (or [task\\_threadiosleepnc](#)).

This function is safe to be called from multiple processes at the same time.

**NOTE:** An application should not stay in a critical section for extended periods of time. 100-200us should be considered the maximum time.

**See also:**

[task\\_leavecritsection](#)

[task\\_threadiosleepnc](#)

#### 8.14.3.2 **unsigned int task\_fork (unsigned char *priority*, unsigned int *savesize*)**

Creates a new task.

Spawns a new task, returning the process ID of the new task to the parent task. Note that because of the way the Keil compiler assigns variables, calls to task\_fork should be wrapped inside a critical section. Make sure the child's process ID is stored in a secure location before exiting the critical section. Note that only the parent need leave the critical section, the child will not run until the parent has left it.

This function is safe to be called from multiple processes at the same time.

**Parameters:**

*priority* priority level for the new task.

*savesize* size of the task state buffer for the new task

**Returns:**

0x0FFFF for failure, else 0 if this is the child task, or the child's PID if this is the parent.

**See also:**

[MIN\\_PRIORITY](#)  
[NORM\\_PRIORITY](#)  
[MAX\\_PRIORITY](#)  
[ROM\\_SAVESIZE](#)  
[task\\_kill](#)  
[task\\_reentrant\\_stack](#)

### 8.14.3.3 void task\_genesis (unsigned int *savesize*)

Initializes the process scheduler.

Note that calling the function [init\\_rom](#) from the initialization library is the preferred way of initializing the ROM.

This function is safe to be called from multiple processes at the same time.

**Parameters:**

*savesize* Size of the task buffer for saving information on task switches.

### 8.14.3.4 unsigned char task\_getcurrent (void)

Gets the process ID for the current task.

Returns the process ID for the current task, which can be used to manage that task.

This function is safe to be called from multiple processes at the same time.

**Returns:**

PID for the current task.

**See also:**

[task\\_kill](#)  
[task\\_setpriority](#)  
[task\\_getpriority](#)

#### 8.14.3.5 unsigned char task\_getpriority (unsigned char *task\_id*)

Gets the priority level for the given task.

Given the process ID of a task, return the priority level for that task. Use a *task\_id* of 0 for the current task.

This function is safe to be called from multiple processes at the same time.

##### Parameters:

*task\_id* Task PID to get the priority for. A task PID of zero means the current task.

##### Returns:

Priority level of the task.

##### See also:

[MIN\\_PRIORITY](#)  
[NORM\\_PRIORITY](#)  
[MAX\\_PRIORITY](#)

#### 8.14.3.6 unsigned char task\_gettaskid ()

Redirected function to get the ID of the current task.

This is a redirected function that should be used to get the process ID of the current task. The default implementation of this function calls the function [task\\_getcurrent](#).

For more information on redirected functions, see the section *ROM Redirect Function Table* in the DS80C400 User's Guide at <http://pdfserv.maxim-ic.com/arpdf/Design/DS80C400UG.pdf>

This function is safe to be called from multiple processes at the same time.

##### Returns:

Task Id of the current task.

##### See also:

[task\\_getcurrent](#)

#### 8.14.3.7 unsigned char task\_getthreadid ()

Redirected function to return the current thread's ID number.

This is a redirected function that should be used to retrieve the current thread's ID number. However, the DS80C400 ROM does not support threads, so the default implementation of this function always returns 0x01.



For more information on redirected functions, see the section *ROM Redirect Function Table* in the DS80C400 User's Guide at <http://pdfserv.maxim-ic.com/arpdf/Design/DS80C400UG.pdf>.

This function is safe to be called from multiple processes at the same time.

**Returns:**

default implementation returns 0x01

#### 8.14.3.8 unsigned int task\_gettickreload (void)

Gets the current reload value for the system's millisecond ticker.

Gets the current reload value for the system's millisecond ticker. When initialized, this reload value may not be correct for the system, and calls to *task\_gettimemillis* may show the resulting inaccuracy (for example, wall time may record 10 seconds while the DS80C400 thinks 12 seconds have passes). Use this function to verify the system's current system millisecond ticker reload value.

This function is safe to be called from multiple processes at the same time.

**See also:**

[task\\_settickreload](#)  
[task\\_gettimemillis](#)

#### 8.14.3.9 void task\_gettimemillis (struct **TIME** \* *t*)

Returns the system tick count.

The default implementation of this function returns the approximate number of milliseconds since the system started. Note that the largest raw data structure supported by Keil is 4 bytes, yet the DS80C400's tick counter is 5 bytes, therefore the special **TIME** structure is used.

This is a redirected function. The ROM includes a default process scheduler implementation. See the *DS80C400 User's Guide* for information on replacing the default process scheduler with your own.

This function is safe to be called from multiple processes at the same time.

**Parameters:**

*t* pointer to a structure of type **TIME** (a 5-byte structure). The result is written to this pointer, MSB first.

**See also:**

[TIME](#)

#### 8.14.3.10 unsigned char task\_kill (unsigned char *task\_id*)

Kills the specified task.

Kill the specified task. Use a *task\_id* of 0 to indicate the current task. This function does not close or clean up any sockets. Use the socket library function [cleanup](#) to clean any sockets owned by the task before any more processes are created.

This function is safe to be called from multiple processes at the same time.

##### Parameters:

*task\_id* Task PID to kill.

##### Returns:

0 for Success, non-zero for failure

##### See also:

[task\\_fork](#)

#### 8.14.3.11 void task\_leavecritsection (void)

Leaves a critical section.

Leaves a critical section, which allows process swapping to continue. Calls to [task\\_leavecritsection](#) should have a matching call to [task\\_entercritsection](#).

This function is safe to be called from multiple processes at the same time.

**NOTE:** An application should not stay in a critical section for extended periods of time. 100-200us should be considered the maximum time.

##### See also:

[task\\_entercritsection](#)

[task\\_threadiosleepnc](#)

#### 8.14.3.12 void xdata\* task\_reentrant\_stack (unsigned int *size*)

Rerserves space on the reentrant stack.

This function reserves the specified amount of space on the reentrant stack. Any task that uses functions declared "reentrant" MUST call task\_reentrant\_stack. **Note that space on the reentrant stack is NOT freed when a task is killed.** An function called from an interrupt uses the reentrant stack of the foreground process. Note that the reentrant stack is part of XDATA. You can adjust the top of the reentrant stack in startup400.a51.

##### Parameters:

*size* Amount (in bytes) to reserve on the reentrant stack.

**Returns:**

-1 for failure (reentrant stack disabled in startup.a51), else the new lower bounds of the reentrant stack.

**See also:**

[task\\_fork](#)

**8.14.3.13 unsigned char task\_setpriority (unsigned char *task\_id*, unsigned char *priority*)**

Sets the priority level for a given task.

Given the process ID of a task, set the priority level for that task. Use a *task\_id* of 0 for the current task.

This function is safe to be called from multiple processes at the same time.

**Parameters:**

*task\_id* Task PID to set the priority for. A task PID of zero means the current task.

*priority* Priority setting for PID *task\_id*. Can be any value between [MIN\\_PRIORITY](#) and [MAX\\_PRIORITY](#)

**Returns:**

0 for Success, non-zero for failure

**See also:**

[MIN\\_PRIORITY](#)  
[NORM\\_PRIORITY](#)  
[MAX\\_PRIORITY](#)

**8.14.3.14 void task\_settickreload (unsigned int *reload*)**

Sets the current reload value for the system's millisecond ticker.

Sets the current reload value for the system's millisecond ticker. When initialized, this reload value may not be correct for the system, and calls to [task\\_gettimemillis](#) may show the resulting inaccuracy (for example, wall time may record 10 seconds while the DS80C400 thinks 12 seconds have passes). Use this function to set the system's current system millisecond ticker reload value.

This function is safe to be called from multiple processes at the same time. This function should only be called after [init\\_rom](#) has been called. If you do not have a 1-Wire device attached for MAC address storage, you should call [init\\_setclock](#) or [init\\_setfrequency](#) before calling [init\\_rom](#) to initialize the system with a good clock reload value.

**Parameters:**

*reload* New value for the system's millisecond reload timer. Some reloads for common crystal frequencies include [RELOAD\\_14\\_746](#), [RELOAD\\_18\\_432](#), [RELOAD\\_29\\_491](#), [RELOAD\\_36\\_864](#), [RELOAD\\_58\\_982](#), and [RELOAD\\_73\\_728](#). Values for other crystals (and crystal settings) can also be used. See the [High Speed Microcontroller's User Guide](#) for more information on timers and timer settings.

**See also:**

[init\\_setclock](#)  
[init\\_setfrequency](#)  
[task\\_gettickreload](#)  
[task\\_gettimemillis](#)

**8.14.3.15 unsigned char task\_signal (unsigned char task\_id, unsigned char event\_mask)**

Posts events to the specified task.

Sends the event(s) in *event\_mask* to process *task\_id*. If the task is waiting for no other events, it will wake up and be electable to run by the task scheduler.

Use the event flags *FLAG\_USER0* through *FLAG\_USER4* or a bitwise OR of these flags. For tasks suspended in *task\_wait*, *FLAG\_SLEEPING* should also be specified, otherwise the task will sleep until the sleep time has elapsed. *FLAG\_SLEEPING* can also be posted to prematurely wake a task suspended in *task\_sleep*.

This function is safe to be called from multiple processes at the same time.

**Parameters:**

*task\_id* Task PID to signal.  
*event\_mask* Bitmap of events to signal.

**Returns:**

0 for Success, non-zero for failure

**See also:**

[task\\_sleep](#)  
[task\\_suspend](#)  
[task\\_wait](#)

**8.14.3.16 unsigned char task\_sleep (unsigned char task, long timeout)**

Redirected function to put a specified task to sleep for a number of milliseconds.

This is a redirected function that should be used to put a task to sleep for some known period of time. The default implementation of this function calls the function [task\\_wait](#). The task can be woken up prematurely using [task\\_signal](#).

For more information on redirected functions, see the section *ROM Redirect Function Table* in the DS80C400 User's Guide at <http://pdfserv.maxim-ic.com/arpdf/Design/DS80C400UG.pdf>

This function is safe to be called from multiple processes at the same time.

**Parameters:**

*task* task ID to put to sleep. A value of zero means put the current task to sleep.

*timeout* amount of time to put 'task' to sleep for

**See also:**

[task\\_wait](#)

[task\\_signal](#)

**8.14.3.17 unsigned char task\_suspend (unsigned char *task\_id*, unsigned char *event\_mask*)**

Suspends the specified task.

Suspends the execution of the specified task until all specified events have occurred. Use the function [task\\_signal](#) to wake the task up. Use the event flags *FLAG\_USER0* through *FLAG\_USER4* or a bitwise OR of these flags only, all other bits are system reserved.

This function is safe to be called from multiple processes at the same time.

**Parameters:**

*task\_id* Task PID to suspend. A task PID of zero means suspend the current task.

*event\_mask* Bitmap of events to wait for before wakeup.

**Returns:**

0 for Success, non-zero for failure

**See also:**

[task\\_signal](#)

[task\\_sleep](#)

**8.14.3.18 unsigned char task\_threadsleep (unsigned char *infinite*, unsigned long *timeout*)**

Redirected function to put the current thread to sleep.

This is a redirected function that should be used to put a thread to sleep. However, the DS80C400 does not support threads, so the default implementation of this function puts the current task to sleep.

For more information on redirected functions, see the section *ROM Redirect Function Table* in the DS80C400 User's Guide at <http://pdfserv.maxim-ic.com/arpdf/Design/DS80C400UG.pdf>

**Warning:**

This function is not multi-process safe. If two processes try to call this function at the same time, its parameters may be destroyed, yielding unpredictable results.

**Parameters:**

*infinite* 0 for non-infinite timeout, non-zero for infinite timeout (until woken)

*timeout* amount of time to sleep (if infinite==0)

**Returns:**

0 for Success, non-zero for failure

**See also:**

[task\\_threadiosleepnc](#)

[task\\_threadresume](#)

**8.14.3.19 unsigned char task\_threadiosleepnc (unsigned char *infinite*, unsigned long *timeout*)**

Redirected function to put the current thread (which is already in a critical section) to sleep.

This is a redirected function that should be used to put a thread to sleep, when the thread has already entered a critical section. However, the DS80C400 does not support threads, so the default implementation of this function puts the current task to sleep (which is assumed to be operating within a critical section).

For more information on redirected functions, see the section *ROM Redirect Function Table* in the DS80C400 User's Guide at <http://pdfserv.maxim-ic.com/arpdf/Design/DS80C400UG.pdf>

**Warning:**

This function is not multi-process safe. If two processes try to call this function at the same time, its parameters may be destroyed, yielding unpredictable results.

**Parameters:**

*infinite* 0 for non-infinite timeout, non-zero for infinite timeout (until woken)

*timeout* amount of time to sleep (if infinite==0)

**Returns:**

0 for Success, non-zero for failure

**See also:**

[task\\_threadiosleep](#)  
[task\\_threadresume](#)  
[task\\_entercritsection](#)

**8.14.3.20 void task\_threadrestore (void)**

Redirected function to restore the state of a thread.

This is a redirected function that should be used to restore the state of a thread that was earlier saved with a call to [task\\_threadsave](#). However, the DS80C400 does not support threads, so the default implementation of this function does nothing.

For more information on redirected functions, see the section *ROM Redirect Function Table* in the DS80C400 User's Guide at <http://pdfserv.maxim-ic.com/arpdf/Design/DS80C400UG.pdf>

This function is safe to be called from multiple processes at the same time.

**See also:**

[task\\_threadsave](#)

**8.14.3.21 unsigned char task\_threadresume (unsigned char *thread*, unsigned char *task*)**

Redirected function to resume the specified thread.

This is a redirected function that should be used to resume a suspended or sleeping thread. However, the DS80C400 ROM does not support threads, so the default implementation of this function resumes the task with a process ID matching *task*.

For more information on redirected functions, see the section *ROM Redirect Function Table* in the DS80C400 User's Guide at <http://pdfserv.maxim-ic.com/arpdf/Design/DS80C400UG.pdf>

This function is safe to be called from multiple processes at the same time.

**Parameters:**

*thread* thread ID to resume  
*task* ID of the process that *thread* belongs to

**Returns:**

0 for Success, non-zero for failure

**See also:**

[task\\_threadiosleep](#)  
[task\\_threadiosleepnc](#)

#### **8.14.3.22 void task\_threadsave (void)**

Redirected function to save the state of the current thread in anticipation of a task/thread swap.

This is a redirected function that should be used to save the state of the current thread so it may be executed again later, after a call to [task\\_threadrestore](#). However, the DS80C400 does not support threads, so the default implementation of this function does nothing.

For more information on redirected functions, see the section *ROM Redirect Function Table* in the DS80C400 User's Guide at <http://pdfserv.maxim-ic.com/arpdf/Design/DS80C400UG.pdf>

This function is safe to be called from multiple processes at the same time.

**See also:**

[task\\_threadrestore](#)

#### **8.14.3.23 unsigned int task\_version (void)**

Returns the version number of this process scheduling library.

This function is safe to be called from multiple processes at the same time.

**Returns:**

Version number of this TASK library.

#### **8.14.3.24 unsigned char task\_wait (unsigned char task\_id, unsigned char event\_mask, long millis)**

Puts the specified task to sleep.

Suspends the execution of the specified task until all specified events have occurred, and/or until a set amount of time has elapsed.

When calling *task\_wait*, use the event flags *FLAG\_USER0* through *FLAG\_USER4* or a bitwise OR of these flags only, all other bits are system reserved.

Use the function [task\\_signal](#) to wake the task up. To properly wake a task, specify *FLAG\_SLEEPING* in the call to *task\_signal* as well as the user event flag(s).

This function is safe to be called from multiple processes at the same time.



**Parameters:**

- task\_id* Task PID to put to sleep. A task PID of zero means put the current task to sleep.
- event\_mask* Bitmap of events to wait for before wakeup.
- millis* Maximum number of milliseconds to sleep for.

**Returns:**

0 for Success, non-zero for failure

**See also:**

[task\\_signal](#)  
[task\\_sleep](#)  
[task\\_suspend](#)

## 8.15 rom400\_tftp.h File Reference

### 8.15.1 Detailed Description

TFTP Client functions in the DS80C400 ROM.

This library contains functions for downloading files from a TFTP server. Note that the function [setftpserver](#) from the socket library must be used to initialize the IP address of the TFTP server before communication can begin.

For detailed information on the DS80C400 please see the [High-Speed Microcontroller User's Guide: DS80C400 Supplement](#).

**Warning:**

The functions in this library are multi-process safe—that is, if you call the same method from two different processes at the same time, the parameters to the function will not be destroyed. However, only one TFTP client is available, and it uses system-wide resources. Therefore, it is recommended that one process manage the TFTP client.

**Defines**

- #define [ROM400\\_TFTP\\_VERSION](#) 5
- #define [TFTP\\_MORE\\_DATA](#) 0
- #define [TFTP\\_LAST\\_SEGMENT](#) 1

**Functions**

- unsigned int [tftp\\_init](#) (void)  
*Initialize the TFTP client.*

- unsigned int [tftp\\_first](#) (unsigned char \*filename)  
*Requests a file from the TFTP server.*
- unsigned int [tftp\\_next](#) (unsigned int ack\_only)  
*Read subsequent blocks of a file from a TFTP server.*
- void \* [tftp\\_getdata](#) (void)  
*Get the pointer to the TFTP client's read buffer.*
- void [tftp\\_close](#) (void)  
*Closes the socket used by the TFTP library.*
- unsigned int [tftp\\_version](#) (void)  
*Returns the version number of this TFTP library.*

## 8.15.2 Define Documentation

### 8.15.2.1 `#define ROM400_TFTP_VERSION 5`

Version number associated with this header file. Should be the same as the version number returned by the [tftp\\_version](#) function.

See also:

[tftp\\_version](#)

### 8.15.2.2 `#define TFTP_LAST_SEGMENT 1`

Argument to function [tftp\\_next](#) requesting the connection be closed.

See also:

[tftp\\_next](#)

### 8.15.2.3 `#define TFTP_MORE_DATA 0`

Argument to function [tftp\\_next](#) requesting more data.

See also:

[tftp\\_next](#)

### 8.15.3 Function Documentation

#### 8.15.3.1 void tftp\_close (void)

Closes the socket used by the TFTP library.

Closes the socket used by the TFTP library. Every call to [tftp\\_first](#) creates a new socket, and must be balanced by a call to [tftp\\_close](#) or the system will have lingering, inaccessible sockets.

**See also:**

[tftp\\_first](#)  
[tftp\\_next](#)

#### 8.15.3.2 unsigned int tftp\_first (unsigned char \* *filename*)

Requests a file from the TFTP server.

Requests the specified file from the TFTP server. As long as the file exists and this function returns successfully, use the buffer pointer returned from [tftp\\_getdata](#) to read the first block of the requested file. Use [tftp\\_next](#) to read subsequent blocks of data. After the TFTP transaction is complete (or an error has occurred and the TFTP transaction will be abandoned), use [tftp\\_close](#) to clean up the transmission socket.

**Parameters:**

*filename* pointer to a null-terminated string that is the file to be requested from the TFTP server

**Returns:**

0x0FFFF on failure, else the number of bytes read this time

**See also:**

[tftp\\_next](#)  
[tftp\\_close](#)  
[tftp\\_getdata](#)

#### 8.15.3.3 void\* tftp\_getdata (void)

Get the pointer to the TFTP client's read buffer.

Applications should read the TFTP data after every call to [tftp\\_first](#) or [tftp\\_next](#). This function only needs to be called once after [tftp\\_init](#) has been called (the buffer pointer does not change).

**Returns:**

Pointer to the area that the TFTP client is writing to

**See also:**

[tftp\\_first](#)  
[tftp\\_next](#)

#### 8.15.3.4 unsigned int tftp\_init (void)

Initialize the TFTP client.

Initializes the TFTP client. Note that the IP address of the TFTP server must be set using the [setftpserver](#) function from the socket library. After the TFTP Client is initialized, call the [tftp\\_getdata](#) function to request a pointer to the TFTP client's buffer.

**Returns:**

0 for success, non-zero for failure

**See also:**

[tftp\\_getdata](#)

#### 8.15.3.5 unsigned int tftp\_next (unsigned int *ack\_only*)

Read subsequent blocks of a file from a TFTP server.

Requests the next block of a file be read from the TFTP server. Use the buffer pointer returned from [tftp\\_getdata](#) to read the block read from the TFTP server. If this function returns less than 512 bytes read, it means this is the last block of data. Call [tftp\\_next](#) one more time with the argument [TFTP\\_LAST\\_SEGMENT](#) to clean up. After the TFTP transaction is complete (or an error has occurred and the TFTP transaction will be abandoned), use [tftp\\_close](#) to clean up the transmission socket.

**Parameters:**

*ack\_only* Use [TFTP\\_MORE\\_DATA](#) to request more data until the amount returned is less than 512 bytes. Use [TFTP\\_LAST\\_SEGMENT](#) to acknowledge the last segment was recieved.

**Returns:**

0x0FFFF on failure, or the number of bytes read.

**See also:**

[tftp\\_first](#)  
[tftp\\_close](#)  
[tftp\\_getdata](#)  
[TFTP\\_MORE\\_DATA](#)  
[TFTP\\_LAST\\_SEGMENT](#)

### 8.15.3.6 unsigned int tftp\_version (void)

Returns the version number of this TFTP library.

#### Returns:

Version number of this TFTP library.

## 8.16 rom400\_useriopoll.h File Reference

### 8.16.1 Detailed Description

User IO Poll registration routines for the DS80C400 ROM.

This library contains functions to register User IO Poll routines. User IO Poll routines are called at least every 4 milliseconds by the system task scheduler. These allow programs to put their applications to sleep while waiting for input, and register a polling routine that will be called to check for that input. The sleeping process can then be signalled to wake up from the polling routine.

For detailed information on the DS80C400 please see the [High-Speed Microcontroller User's Guide: DS80C400 Supplement](#).

The functions in this library are multi-process safe—that is, if you call the same method from two different processes at the same time, the parameters to the function are preserved, and the function should execute correctly.

#### Defines

- #define [ROM400\\_USERIOPOLL\\_VERSION](#) 1

#### Functions

- unsigned char [useriopoll\\_isinstalled](#) (void)  
*Checks to see if the User IO Poll library has already been initialized.*
- void [useriopoll\\_init](#) (unsigned char num\_routines)  
*Initializes the User IO Poll library.*
- unsigned char [useriopoll\\_registerpollroutine](#) (void \*funct, unsigned char number)  
*Registers an IO Poll routine.*
- unsigned char [useriopoll\\_removepollroutine](#) (unsigned char number)  
*Removes a registered IO Poll routine.*

- void \* [useriopoll\\_getpollroutine](#) (unsigned char number)  
*Gets the address of a registered IO Poll routine.*
- unsigned char [useriopoll\\_getlistsize](#) (void)  
*Returns the number of polling routines allowed.*
- unsigned int [useriopoll\\_version](#) (void)  
*Returns the version number of this User IO Poll library.*

## 8.16.2 Define Documentation

### 8.16.2.1 #define ROM400\_USERIOPOLL\_VERSION 1

Version number associated with this header file. Should be the same as the version number returned by the [useriopoll\\_version](#) function.

See also:

[useriopoll\\_version](#)

## 8.16.3 Function Documentation

### 8.16.3.1 unsigned char useriopoll\_getlistsize (void)

Returns the number of polling routines allowed.

Returns the size of the internal array that holds the registered polling routines. This is the same as the number of entries that this library was initialized for. This number can be considered the bounds of valid indexes for the [useriopoll\\_getpollroutine](#), [useriopoll\\_removepollroutine](#), and [useriopoll\\_registerpollroutine](#) functions.

**Returns:**

Size of the list of polling routines.

See also:

[useriopoll\\_init](#)  
[useriopoll\\_registerpollroutine](#)  
[useriopoll\\_getpollroutine](#)  
[useriopoll\\_removepollroutine](#)

### 8.16.3.2 void\* useriopoll\_getpollroutine (unsigned char number)

Gets the address of a registered IO Poll routine.

Gets the address of an entry in the list of registered IO Poll routines. If no entry exists in the list at this location, this function returns NULL.

**Parameters:**

*number* location in the list of polling routines to clear

**Returns:**

address of the registered IO Poll routine, or NULL if no routine exists at that position in the list

**See also:**

[useriopoll\\_init](#)  
[useriopoll\\_registerpollroutine](#)  
[useriopoll\\_removepollroutine](#)

**8.16.3.3 void useriopoll\_init (unsigned char *num\_routines*)**

Initializes the User IO Poll library.

Initializes memory space required by the User IO Poll library. The argument should be the maximum number of IO Poll routines that will be needed by the library. Internally, this is represented by an array of function pointers. Every 4 milliseconds (or more often), all the function pointers in the array are invoked (if they have been set). Therefore, it is in an application's best interest to make this number the lowest possible to reduce overhead.

The functions registered as IO Poll routines should not destroy any registers aside from the following: psw, acc, dptr0. All other registers should be preserved.

**Parameters:**

*num\_routines* number of IO Poll routines that can be registered

**See also:**

[useriopoll\\_isinstalled](#)  
[useriopoll\\_registerpollroutine](#)

**8.16.3.4 unsigned char useriopoll\_isinstalled (void)**

Checks to see if the User IO Poll library has already been initialized.

Checks to see if the [useriopoll\\_init](#) function has already been called. This function allows libraries to determine if they need to initialize this library or not.

**Returns:**

0 if the library has not been initialized, 1 if it has

**See also:**

[useriopoll\\_init](#)

#### **8.16.3.5 unsigned char useriopoll\_registerpollroutine (void \* *funct*, unsigned char *number*)**

Registers an IO Poll routine.

Registers the given IO Poll routine to be called by the task scheduler. The function will be installed in the list of functions at the position defined by *number*, even if a function already exists at that location.

##### **Parameters:**

*funct* function pointer of the IO Poll routine

*number* location in the list of polling routines to place this function

##### **Returns:**

0 if the operation was successful, 1 if *number* was out of bounds

##### **See also:**

[useriopoll\\_init](#)

[useriopoll\\_removepollroutine](#)

[useriopoll\\_getpollroutine](#)

#### **8.16.3.6 unsigned char useriopoll\_removepollroutine (unsigned char *number*)**

Removes a registered IO Poll routine.

Removes an entry in the list of registered IO Poll routines. If no entry exists in the list at this location, this function has no effect.

##### **Parameters:**

*number* location in the list of polling routines to clear

##### **Returns:**

0 if the operation was successful, 1 if *number* was out of bounds

##### **See also:**

[useriopoll\\_init](#)

[useriopoll\\_registerpollroutine](#)

[useriopoll\\_getpollroutine](#)

#### **8.16.3.7 unsigned int useriopoll\_version (void)**

Returns the version number of this User IO Poll library.

##### **Returns:**

Version number of this User IO Poll library.



## 8.17 rom400\_util.h File Reference

### 8.17.1 Detailed Description

Utility functions in the DS80C400 ROM.

This library contains CRC, pseudo-RNG and utility memory functions.

For detailed information on the DS80C400 please see the [High-Speed Microcontroller User's Guide: DS80C400 Supplement](#).

#### Warning:

Some functions in this library are **NOT** multi-process safe—that is, if you call the same method from two different processes at the same time, the parameters to the function may be destroyed, yielding unpredictable results. Consult each individual function's documentation for details on which functions are multi-process safe.

#### Defines

- #define [ROM400\\_UTIL\\_VERSION](#) 5
- #define [REDIRECT\\_KERNELMALLOC](#) 1
- #define [REDIRECT\\_KERNELFREE](#) 2
- #define [REDIRECT\\_MALLOC](#) 3
- #define [REDIRECT\\_FREE](#) 4
- #define [REDIRECT\\_MALLOCDIRTY](#) 5
- #define [REDIRECT\\_TINIEXPORT\\_MM\\_DEREF](#) 6
- #define [REDIRECT\\_GETFREERAM](#) 7
- #define [REDIRECT\\_GETTIMEMILLIS](#) 8
- #define [REDIRECT\\_GETTHREADID](#) 9
- #define [REDIRECT\\_THREADRESUME](#) 10
- #define [REDIRECT\\_THREADIOSLEEP](#) 11
- #define [REDIRECT\\_THREADIOSLEEPNC](#) 12
- #define [REDIRECT\\_THREADSAVE](#) 13
- #define [REDIRECT\\_THREADRESTORE](#) 14
- #define [REDIRECT\\_SLEEP](#) 15
- #define [REDIRECT\\_GETTASKID](#) 16
- #define [REDIRECT\\_INFOSENDCHAR](#) 17
- #define [REDIRECT\\_IP\\_COMPUTECHECKSUM\\_SOFTWARE](#) 18
- #define [REDIRECT\\_0](#) 19
- #define [REDIRECT\\_DHCPNOTIFY](#) 20
- #define [REDIRECT\\_ROM\\_TASK\\_CREATE](#) 21
- #define [REDIRECT\\_ROM\\_TASK\\_DUPLICATE](#) 22
- #define [REDIRECT\\_ROM\\_TASK\\_DESTROY](#) 23
- #define [REDIRECT\\_ROM\\_TASK\\_SWITCH\\_IN](#) 24

- #define REDIRECT\_ROM\_TASK\_SWITCH\_OUT 25
- #define REDIRECT\_OWIP\_READCONFIG 26
- #define REDIRECT\_SETMACID 27
- #define REDIRECT\_MM\_UNDEREF 28
- #define REDIRECT\_USER\_IOPOLL 29
- #define REDIRECT\_ERROR\_NOTIFICATION 30

## Functions

- unsigned int [util\\_crc16](#) (unsigned char value, unsigned int seed)  
*Generates a 16-bit CRC given a seed.*
- unsigned char [util\\_getpseudorandom](#) (void)  
*Gets a pseudo-random byte.*
- void [util\\_setrandomseed](#) (unsigned int seed)  
*Sets the seed of the random number generator.*
- void [util\\_memclear](#) (void \*target, unsigned int length)  
*Clears a block of memory.*
- void [util\\_memcpy](#) (void \*source, void \*dest, unsigned int length)  
*Copies a block of memory.*
- unsigned char [util\\_memcompare](#) (void \*block0, void \*block1, unsigned int length)  
*Compares the values in 2 blocks of memory.*
- void [util\\_infosendchar](#) (unsigned char ch)  
*Sends a character to serial port 0.*
- void [util\\_installhook](#) (void \*fncptr, unsigned int fncindex)  
*Installs a new function pointer into the ROM redirect table.*
- unsigned int [util\\_version](#) (void)  
*Returns the version number of this utility library.*

## 8.17.2 Define Documentation

### 8.17.2.1 #define REDIRECT\_0 19

Reserved for future use with the [util\\_installhook](#) method.

See also:

[util\\_installhook](#)

#### 8.17.2.2 **#define REDIRECT\_DHCPNOTIFY 20**

Value to be used in conjunction with the [util\\_installhook](#) method to override the *DHCP-Notify* method.

See also:

[util\\_installhook](#)

#### 8.17.2.3 **#define REDIRECT\_ERROR\_NOTIFICATION 30**

Value to be used in conjunction with the [util\\_installhook](#) method to override the *Error-Notification* method.

See also:

[util\\_installhook](#)

#### 8.17.2.4 **#define REDIRECT\_FREE 4**

Value to be used in conjunction with the [util\\_installhook](#) method to override the *mem\_free* method.

See also:

[util\\_installhook](#)

#### 8.17.2.5 **#define REDIRECT\_GETFREERAM 7**

Value to be used in conjunction with the [util\\_installhook](#) method to override the *mem\_getfreeram* method.

See also:

[util\\_installhook](#)

#### 8.17.2.6 **#define REDIRECT\_GETTASKID 16**

Value to be used in conjunction with the [util\\_installhook](#) method to override the *task\_gettaskid* method.

See also:

[util\\_installhook](#)

#### 8.17.2.7 **#define REDIRECT\_GETTHREADID 9**

Value to be used in conjunction with the [util\\_installhook](#) method to override the [task\\_getthreadid](#) method.

See also:

[util\\_installhook](#)

#### 8.17.2.8 **#define REDIRECT\_GETTIMEMILLIS 8**

Value to be used in conjunction with the [util\\_installhook](#) method to override the [task\\_gettimemillis](#) method.

See also:

[util\\_installhook](#)

#### 8.17.2.9 **#define REDIRECT\_INFOSENDCHAR 17**

Value to be used in conjunction with the [util\\_installhook](#) method to override the [util\\_infosendchar](#) method.

See also:

[util\\_installhook](#)

#### 8.17.2.10 **#define REDIRECT\_IP\_COMPUTECHECKSUM\_SOFTWARE 18**

Value to be used in conjunction with the [util\\_installhook](#) method to override the [IP\\_ComputeChecksum](#) method.

See also:

[util\\_installhook](#)

#### 8.17.2.11 **#define REDIRECT\_KERNELFREE 2**

Value to be used in conjunction with the [util\\_installhook](#) method to override the [Kernel-Free](#) method.

See also:

[util\\_installhook](#)

#### 8.17.2.12 **#define REDIRECT\_KERNELMALLOC 1**

Value to be used in conjunction with the [util\\_installhook](#) method to override the *Kernel-Malloc* method.

**See also:**

[util\\_installhook](#)

#### 8.17.2.13 **#define REDIRECT\_MALLOC 3**

Value to be used in conjunction with the [util\\_installhook](#) method to override the *mem\_malloc* method.

**See also:**

[util\\_installhook](#)

#### 8.17.2.14 **#define REDIRECT\_MALLOCDIRTY 5**

Value to be used in conjunction with the [util\\_installhook](#) method to override the *mem\_mallocdirty* method.

**See also:**

[util\\_installhook](#)

#### 8.17.2.15 **#define REDIRECT\_MM\_UNDEREF 28**

Value to be used in conjunction with the [util\\_installhook](#) method to override the *M\_UnDeref* method.

**See also:**

[util\\_installhook](#)

#### 8.17.2.16 **#define REDIRECT\_OWIP\_READCONFIG 26**

Value to be used in conjunction with the [util\\_installhook](#) method to override the *OWIP\_ReadConfig* method.

**See also:**

[util\\_installhook](#)

#### **8.17.2.17 #define REDIRECT\_ROM\_TASK\_CREATE 21**

Value to be used in conjunction with the [util\\_installhook](#) method to override the *Task-Create* method.

**See also:**

[util\\_installhook](#)

#### **8.17.2.18 #define REDIRECT\_ROM\_TASK\_DESTROY 23**

Value to be used in conjunction with the [util\\_installhook](#) method to override the *task-kill* method.

**See also:**

[util\\_installhook](#)

#### **8.17.2.19 #define REDIRECT\_ROM\_TASK\_DUPLICATE 22**

Value to be used in conjunction with the [util\\_installhook](#) method to override the *Task-Duplicate* method.

**See also:**

[util\\_installhook](#)

#### **8.17.2.20 #define REDIRECT\_ROM\_TASK\_SWITCH\_IN 24**

Value to be used in conjunction with the [util\\_installhook](#) method to override the *Task-SwitchIn* method.

**See also:**

[util\\_installhook](#)

#### **8.17.2.21 #define REDIRECT\_ROM\_TASK\_SWITCH\_OUT 25**

Value to be used in conjunction with the [util\\_installhook](#) method to override the *Task-SwitchOut* method.

**See also:**

[util\\_installhook](#)

#### 8.17.2.22 **#define REDIRECT\_SETMACID 27**

Value to be used in conjunction with the [util\\_installhook](#) method to override the *Set-MACID* method.

See also:

[util\\_installhook](#)

#### 8.17.2.23 **#define REDIRECT\_SLEEP 15**

Value to be used in conjunction with the [util\\_installhook](#) method to override the *task\_sleep* method.

See also:

[util\\_installhook](#)

#### 8.17.2.24 **#define REDIRECT\_THREADIOSLEEP 11**

Value to be used in conjunction with the [util\\_installhook](#) method to override the *task\_threadiosleep* method.

See also:

[util\\_installhook](#)

#### 8.17.2.25 **#define REDIRECT\_THREADIOSLEEPNC 12**

Value to be used in conjunction with the [util\\_installhook](#) method to override the *task\_threadiosleepnc* method.

See also:

[util\\_installhook](#)

#### 8.17.2.26 **#define REDIRECT\_THREADRESTORE 14**

Value to be used in conjunction with the [util\\_installhook](#) method to override the *task\_threadrestore* method.

See also:

[util\\_installhook](#)

#### 8.17.2.27 **#define REDIRECT\_THREADRESUME 10**

Value to be used in conjunction with the [util\\_installhook](#) method to override the [task\\_threadresume](#) method.

See also:

[util\\_installhook](#)

#### 8.17.2.28 **#define REDIRECT\_THREADSAVE 13**

Value to be used in conjunction with the [util\\_installhook](#) method to override the [task\\_threadsave](#) method.

See also:

[util\\_installhook](#)

#### 8.17.2.29 **#define REDIRECT\_TINIEXPORT\_MM\_DEREF 6**

Value to be used in conjunction with the [util\\_installhook](#) method to override the [MM\\_Deref](#) method.

See also:

[util\\_installhook](#)

#### 8.17.2.30 **#define REDIRECT\_USER\_IOPOLL 29**

Value to be used in conjunction with the [util\\_installhook](#) method to override the [User\\_IOPoll](#) method.

See also:

[util\\_installhook](#)

#### 8.17.2.31 **#define ROM400\_UTIL\_VERSION 5**

Version number associated with this header file. Should be the same as the version number returned by the [util\\_version](#) function.

See also:

[util\\_version](#)

### 8.17.3 Function Documentation

#### 8.17.3.1 unsigned int util\_crc16 (unsigned char *value*, unsigned int *seed*)



Generates a 16-bit CRC given a seed.

Implements the Cyclic-Redundancy Check CRC16. This CRC is based on the polynomial  $X^{16} + X^{15} + X^2 + 1$ . It is used extensively in operations with Dallas Semiconductor 1-Wire devices.

This function is safe to be called from multiple processes at the same time.

**Parameters:**

*value* single byte input value to the crc function

*seed* 16 bit 'previous result' seed

**Returns:**

16 bit CRC result

### 8.17.3.2 unsigned char util\_getpseudorandom (void)

Gets a pseudo-random byte.

Returns a pseudo-random byte generated with the help of the CRC function. This is not a true random byte, as there is no real source of entropy.

This function is safe to be called from multiple processes at the same time.

**Returns:**

One pseudorandom byte.

### 8.17.3.3 void util\_infosendchar (unsigned char *ch*)

Sends a character to serial port 0.

This is a redirected function. The DS80C400 silicon software version of this function accesses the serial loader pin (P1.7) and does nothing if this pin is in the logic low state. The DS80C400 silicon software does not use interrupt driver I/O to the serial port.

This function is safe to be called from multiple processes at the same time.

**Parameters:**

*ch* character to send to the debug port

### 8.17.3.4 void util\_installhook (void \**fncptr*, unsigned int *fncindex*)

Installs a new function pointer into the ROM redirect table.

This function alters the redirect table, which allows functions in the ROM to be overridden by intredpid users. The function that is redirected will now call the code at address

*fnctr*. It is not advised that *fnctr* point to a C function unless no arguments are expected (there is no way without writing an assembler wrapper to get the arguments to the C function in the Keil compiler).

See the DS80C400 User's Guide Supplement for more on the meaning of *redirected functions*.

This function is safe to be called from multiple processes at the same time.

**Parameters:**

*fnctr* address of the function that will be inserted into the redirect table

*fnindex* number of the redirected function that will be altered (i.e. [REDIRECT\\_KERNELMALLOC](#))

#### 8.17.3.5 void util\_memclear (void \* *target*, unsigned int *length*)

Clears a block of memory.

Sets *length* bytes to zero starting at address *target*.

This function is safe to be called from multiple processes at the same time.

**Parameters:**

*target* beginning address of memory to clear

*length* number of bytes to clear

#### 8.17.3.6 unsigned char util\_memcompare (void \* *block0*, void \* *block1*, unsigned int *length*)

Compares the values in 2 blocks of memory.

Compares *length* bytes from *block0* to *length* bytes from *block1* for equality. If the two memory blocks are identical, the function returns 0.

**Warning:**

This function is not multi-process safe. If two processes try to call this function at the same time, its parameters may be destroyed, yielding unpredictable results.

**Parameters:**

*block0* first input block to compare

*block1* second input block to compare

*length* maximum number of bytes to compare

**Returns:**

0 if the blocks are identical, non-zero otherwise

#### 8.17.3.7 void util\_memcpy (void \* *source*, void \* *dest*, unsigned int *length*)

Copies a block of memory.

Copies *length* bytes of data from the *source* pointer to the *dest* pointer. The copy operation starts from the beginning of the *source* pointer, placing bytes from the beginning of the *dest* buffer. Therefore, if the buffers referenced by *source* and *dest* overlap, some bytes from *source* bytes will be overwritten prior to being copied to the target.

##### Warning:

This function is not multi-process safe. If two processes try to call this function at the same time, its parameters may be destroyed, yielding unpredictable results.

##### Parameters:

*source* pointer to bytes that will be the source of the copy

*dest* pointer to the bytes that will be copied to

*length* number of bytes to copy from *source* to *dest*

#### 8.17.3.8 void util\_setrandomseed (unsigned int *seed*)

Sets the seed of the random number generator.

Changes the current value of the random seed to the random number generator, allowing for additional randomness to be inserted into the generation. Note that additional randomness is also generated by the timer bytes and the millisecond counter, so this seed is not the only source.

This function is safe to be called from multiple processes at the same time.

##### Parameters:

*seed* new random seed

#### 8.17.3.9 unsigned int util\_version (void)

Returns the version number of this utility library.

This function is safe to be called from multiple processes at the same time.

##### Returns:

Version number of this UTIL library.

## 8.18 rom400\_xnetstack.h File Reference

### 8.18.1 Detailed Description

Enhanced network stack for the DS80C400 ROM.

This library contains a replacement network stack with better performance and more standards compliant functionality. Since this library will replace the default ROM network stack, be careful of the physical location this library. If this library is targeted to reside in flash memory, your system will be limited by the speed of your flash.

To use this functionality, add `xnetstack_install()` to your program before calling `init_rom` and add the library to your build process.

For detailed information on the DS80C400 please see the [High-Speed Microcontroller User's Guide: DS80C400 Supplement](#).

## Defines

- `#define ROM400_XNETSTACK_VERSION` 16
- `#define SOCKET_TYPE_RAW` 2
- `#define SOCK_RAW` 2
- `#define MDIO_ENABLE` 0
- `#define MDIO_DISABLE_HDX` 1
- `#define MDIO_DISABLE_FDX` 2

## Functions

- `void xnetstack_install (void)`  
*Installs the enhanced network stack.*
- `unsigned int xnetstack_version (void)`  
*Returns the version number of this library.*
- `void xnetstack_set_tcptimeoutfactor (int factor)`  
*Sets a factor to scale all TCP timeouts.*
- `int xnetstack_get_tcptimeoutfactor (void)`  
*Gets the factor to scale all TCP timeouts.*
- `void xnetstack_set_ipv6 (int enable)`  
*Enables/disables IPv6.*
- `void xnetstack_set_icmpechoresponses (int enable)`  
*Enables/disables ICMP echo replies.*
- `void xnetstack_set_icmpdestinationunreachable (int enable)`  
*Enables/disables ICMP destination unreachable messages.*
- `void xnetstack_set_igmpreporttype (int type)`

*Sets the IGMP membership report type.*

- void [xnetstack\\_set\\_arptimeout](#) (int timeout)  
*Sets the ARP timeout.*
- int [xnetstack\\_get\\_arptimeout](#) (void)  
*Gets the ARP timeout value.*
- int [xnetstack\\_set\\_arptablesz](#) (int entries)  
*Sets the number of ARP table entries.*
- int [xnetstack\\_get\\_arptablesz](#) (void)  
*Gets the maximum number of ARP table entries.*
- void [xnetstack\\_set\\_rawfilter](#) (unsigned int proto)  
*Sets a protocol filter for the RAW socket.*
- void [xnetstack\\_disable\\_rawfilter](#) (void)  
*Disables the protocol filter for the RAW socket.*
- void [xnetstack\\_set\\_mdio](#) (int value)  
*Sets whether the MII interface should be used to talk to the physical network interface chip (PHY).*
- void [xnetstack\\_set\\_igmp](#) (int enable)  
*Enables/disables inbound IGMP processing.*

## 8.18.2 Define Documentation

### 8.18.2.1 #define MDIO\_DISABLE\_FDX 2

Argument to function [xnetstack\\_set\\_mdio](#) to disable MDIO link detection and to force the link to full duplex.

**See also:**

[xnetstack\\_set\\_mdio](#)

### 8.18.2.2 #define MDIO\_DISABLE\_HDX 1

Argument to function [xnetstack\\_set\\_mdio](#) to disable MDIO link detection and to force the link to half duplex.

**See also:**

[xnetstack\\_set\\_mdio](#)

#### 8.18.2.3 **#define MDIO\_ENABLE 0**

Argument to function [xnetstack\\_set\\_mdio](#) to enable MDIO link detection.

See also:

[xnetstack\\_set\\_mdio](#)

#### 8.18.2.4 **#define ROM400\_XNETSTACK\_VERSION 16**

Version number associated with this header file. Should be the same as the version number returned by the [xnetstack\\_version](#) function.

See also:

[xnetstack\\_version](#)

#### 8.18.2.5 **#define SOCK\_RAW 2**

Argument to function [socket](#) to create a RAW socket (same as [SOCKET\\_TYPE\\_RAW](#))

See also:

[socket](#)

#### 8.18.2.6 **#define SOCKET\_TYPE\_RAW 2**

Argument to function [socket](#) to create a RAW socket (same as [SOCK\\_RAW](#))

See also:

[socket](#)

### 8.18.3 **Function Documentation**

#### 8.18.3.1 **void xnetstack\_disable\_rawfilter (void)**

Disables the protocol filter for the RAW socket.

This function disables the filter set by [xnetstack\\_set\\_rawfilter](#).

See also:

[xnetstack\\_set\\_rawfilter](#)

#### **8.18.3.2 int xnetstack\_get\_arptablesize (void)**

Gets the maximum number of ARP table entries.

**Returns:**

ARP table size

**See also:**

[xnetstack\\_set\\_arptimeout](#)

#### **8.18.3.3 int xnetstack\_get\_arptimeout (void)**

Gets the ARP timeout value.

**Returns:**

ARP timeout

**See also:**

[xnetstack\\_set\\_arptimeout](#)

#### **8.18.3.4 int xnetstack\_get\_tcptimeoutfactor (void)**

Gets the factor to scale all TCP timeouts.

**Returns:**

TCP scale factor

**See also:**

[xnetstack\\_set\\_tcptimeoutfactor](#)

#### **8.18.3.5 void xnetstack\_install (void)**

Installs the enhanced network stack.

This function installs the enhanced network stack functionality. The function has to be called before [init\\_rom\(\)](#).

#### **8.18.3.6 int xnetstack\_set\_arptablesize (int *entries*)**

Sets the number of ARP table entries.

**Parameters:**

*entries* ARP table entries (16 to 127)

**Returns:**

0 on success

This function allocates the new ARP table from the kernel memory subsystem. The memory allocated will be the table entry count \* 12. **NOTE:** Resizing the ARP table will destroy all entries, including manually set static entries.

**See also:**

[xnetstack\\_set\\_arptablesize](#)

**8.18.3.7 void xnetstack\_set\_arptimeout (int *timeout*)**

Sets the ARP timeout.

**Parameters:**

*timeout* ARP timeout (1 to 255, default: 16)

This function manipulates the amount of timer ticks an ARP entry can be pending (be unresolved) before the network stack considers a host to be unreachable.

Note that [init\\_rom\(\)](#) resets the ARP timeout value.

**See also:**

[xnetstack\\_get\\_arptimeout](#)

**8.18.3.8 void xnetstack\_set\_icmpdestinationunreachable (int *enable*)**

Enables/disables ICMP destination unreachable messages.

**Parameters:**

*enable* 1 to enable, 0 to disable

Setting this to 0 prevents the network stack from generating ICMP destination unreachables (i.e. the device will not respond when an unused port is accessed).

**8.18.3.9 void xnetstack\_set\_icmpechoreplies (int *enable*)**

Enables/disables ICMP echo replies.

**Parameters:**

*enable* 1 to enable, 0 to disable

Setting this to 0 prevents the network stack from generating ICMP echo replies (i.e. the device will no longer respond to "ping").



#### 8.18.3.10 void xnetstack\_set\_igmp (int *enable*)

Enables/disables inbound IGMP processing.

**Parameters:**

*enable* 1 to enable, 0 to disable

**NOTE:** This function disables the IGMP receiver and feeds all inbound packets to the raw packet queue. If there is no raw socket, or if the raw filter doesn't match, the packets will be discarded.

#### 8.18.3.11 void xnetstack\_set\_igmpreporttype (int *type*)

Sets the IGMP membership report type.

**Parameters:**

*type* (0x12 for version 1, 0x16 for version 2)

**NOTE:** This does not enable IGMPv2 compatibility, it merely changes the type of membership reports to work around a problem with certain switches. The default is IGMPv1.

#### 8.18.3.12 void xnetstack\_set\_ipv6 (int *enable*)

Enables/disables IPv6.

**Parameters:**

*enable* 1 to enable, 0 to disable

**NOTE:** This function disables the IPv6 receiver and transmitter. An application can still send packet to IPv6 addresses without receiving an error message; these packets will be discarded at the driver level.

#### 8.18.3.13 void xnetstack\_set\_mdio (int *value*)

Sets whether the MII interface should be used to talk to the physical network interface chip (PHY).

**Parameters:**

*value* [MDIO\\_ENABLE](#), [MDIO\\_DISABLE\\_HDX](#), [MDIO\\_DISABLE\\_FDX](#)

This function enables / disables MII communication over the MII interface. MII communication with the physical interface is used to determine link availability and status. When disabling MDIO, you must specify whether the link is half duplex or full duplex. This function should be called after [xnetstack\\_install](#), but before [init\\_rom](#).

#### 8.18.3.14 void xnetstack\_set\_rawfilter (unsigned int *proto*)

Sets a protocol filter for the RAW socket.

**Parameters:**

*proto* Ethernet protocol (e.g. 0x0800 for IPv4)

This function reduces the system load when a raw socket is used by filtering for a given Ethernet protocol type at the Ethernet driver layer. Note: The network stack only supports Ethernet II frames.

**See also:**

[xnetstack\\_disable\\_rawfilter](#)

#### 8.18.3.15 void xnetstack\_set\_tcptimeoutfactor (int *factor*)

Sets a factor to scale all TCP timeouts.

**Parameters:**

*factor* TCP scale factor (1 to 255, default: 32)

**See also:**

[xnetstack\\_get\\_tcptimeoutfactor](#)

#### 8.18.3.16 unsigned int xnetstack\_version (void)

Returns the version number of this library.

**Returns:**

Version number of this library.

## 8.19 stdio.h File Reference

### 8.19.1 Detailed Description

File and other IO functions.

This library contains functions for file system operations and formatting input and output data. The file system has been adapted from TINI's Java Runtime Environment to be able to be called from a C program.

The file system must reside in contiguous memory. Pages in the file system are 256-byte blocks (on 256-byte boundaries).

**Maximum File System Size** The maximum size(15MB), of the file system is likely to be far beyond the needs of most/any applications. The file system's memory manager has several overhead blocks used to maintain information on block allocation. The number of overhead blocks cannot exceed 255 blocks (65280 bytes).

### Overhead Bytes

- 11 blocks for filesystem overhead
- 5 bytes magic signature
- 'num\_blocks' bytes for the free list
- 'numfd' \* 26 bytes for open file descriptors

Assuming we use the usual 'numfd' value of 8 open file descriptors,  $65280 - (11 * 256) - 5 - (8 * 26) = 62251$  max bytes are available for a free list. This yields a maximum file system size of just over 15MB, although not all the space can be utilized by file data due to file system overhead.

**Example of File System Memory Usage: 64KB Available** We have 64KB of memory available for file system space and we use 'numfd' of 8 open file descriptors, that is 256 blocks of size 256 bytes. Initialized via `finit(8, 256, start_address)`. To determine the amount of data space this allows we subtract the overhead blocks from 64KB  $65536 - (11 * 256) - 5 - (8 * 26)$  leaving 62507 bytes for the free list and the file data. We need 1 byte per free page, so  $62507 / 256 = 244.16$  means one page/block is needed to hold the free list, leaving us with 243 blocks for file system data. 243 blocks is 62208 bytes of available file space, but not all can be used due to internal fragmentation of the file system. File system sectors are allocated in 768 bytes chunks with 512 bytes of data and 256 bytes of filesystem overhead. So the maximum useable file space is  $(62208 / 768) * 512 = 41472$  bytes.

**Example of File System Memory Usage: 128KB of Storage Required** Our application requires that we have 128KB of file system space. Again we will use 8 open file descriptors. 128KB normally means 131072 bytes. Due to internal fragmentation we require  $(131072 / 512) * 768 = 196608$  bytes. This means we require 768 free blocks of file space. 768 blocks require 3 blocks for the free list. As previously the 5 magic bytes and the open file descriptor space come to 213 bytes, this equates to 1 block.  $768 + 11 + 3 + 1 = 783$ . To initialize the file system for our requirements: `finit(8, 783, start_address)`.

For detailed information on the DS80C400 please see the [High-Speed Microcontroller User's Guide: DS80C400 Supplement](#).

**Warning:**

Some functions in this library are **NOT** multi-process safe—that is, if you call the same method from two different processes at the same time, the parameters to the function may be destroyed, yielding unpredictable results. Consult each individual function's documentation for details on which functions are multi-process safe.

```
#include <stddef.h>
```

**Data Structures**

- struct [file\\_structure](#)

**Defines**

- #define [FS\\_VERSION](#) 10
- #define [NULL](#) ((void \*) 0)
- #define [FILE\\_FLAGS\\_EOF](#) 1
- #define [FILE\\_FLAGS\\_TEMP](#) 2
- #define [FILE\\_TYPE\\_TINIFS](#) 1
- #define [FILENAME\\_MAX](#) 255
- #define [FOPEN\\_MAX](#) 8
- #define [L\\_tmpnam](#) 20
- #define [SEEK\\_CUR](#) 0x5555
- #define [SEEK\\_END](#) 0x5556
- #define [SEEK\\_SET](#) 0x5557
- #define [TMP\\_MAX](#) 10
- #define [EOF](#) -1
- #define [P\\_tmpdir](#) "temp"

**Typedefs**

- typedef unsigned int [size\\_t](#)
- typedef unsigned int [off\\_t](#)
- typedef long [fpos\\_t](#)
- typedef [file\\_structure](#) [FILE](#)

**Functions**

- void [clearerr](#) ([FILE](#) \*f\_handle)  
*Clear the error indicators for a file stream.*
- int [fclose](#) ([FILE](#) \*f\_handle)

*Closes the file stream.*

- `int feof (FILE *f_handle)`  
*Checks to see if this stream has reached the end of the file.*
- `int ferror (FILE *f_handle)`  
*Gets the error indicator for the file stream.*
- `int fgetc (FILE *f_handle)`  
*Gets the next unsigned character from the file stream.*
- `int fgetpos (FILE *f_handle, fpos\_t *position)`  
*Gets the current value of the file position indicator.*
- `char * fgets (char *string, int num, FILE *f_handle)`  
*Reads a string from the file stream.*
- `FILE * fopen (const char *filename, const char *mode)`  
*Opens the specified file.*
- `int fputc (int ch, FILE *f_handle)`  
*Writes a character to a file stream.*
- `int fputs (const char *str, FILE *f_handle)`  
*Writes a string to a file stream.*
- `size\_t fread (void *ptr, size\_t size, size\_t num, FILE *f_handle)`  
*Read a number of bytes from a file stream.*
- `FILE * freopen (const char *newfilename, const char *mode, FILE *old_handle)`  
*Associates an open stream with a different file.*
- `int fseek (FILE *f_handle, long int offset, int tag)`  
*Sets the file position indicator.*
- `int fseeko (FILE *f_handle, off\_t offset, int tag)`  
*Sets the file position indicator.*
- `int fsetpos (FILE *f_handle, const fpos\_t *position)`  
*Sets the file position indicator.*
- `long ftell (FILE *f_handle)`

*Gets the file position indicator.*

- `off_t ftello (FILE *f_handle)`  
*Gets the file position indicator.*
- `void flockfile (FILE *f_handle)`  
*Gets exclusive access to a file.*
- `int ftrylockfile (FILE *f_handle)`  
*Tries to get exclusive access to a file.*
- `void funlockfile (FILE *f_handle)`  
*Release exclusive access on a file.*
- `size_t fwrite (const void *ptr, size_t size, size_t num, FILE *f_handle)`  
*Write a number of bytes to a file stream.*
- `int getc (FILE *f_handle)`  
*Gets the next unsigned character from the file stream.*
- `int putc (int value, FILE *f_handle)`  
*Writes a character to a file stream.*
- `int remove (const char *filename)`  
*Removes a file from the file system.*
- `int rename (const char *oldname, const char *newname)`  
*Renames a file.*
- `void rewind (FILE *f_handle)`  
*Resets the file position indicator for a stream.*
- `char * tempnam (const char *dirname, const char *pfx)`  
*Generates a path/filename that can be used for a temporary file.*
- `FILE * tmpfile (void)`  
*Generates a stream to a temporary file.*
- `char * tmpnam (char *nametarget)`  
*Generates a unique temporary filename.*
- `int fflush (FILE *f_handle)`

*Flushes the buffers for a file stream.*

- int **fcleaninit** (char numfd, int numblocks, void \*start\_address)  
*Initializes the file system to a blank state.*
- int **finit** (char numfd, int numblocks, void \*start\_address)  
*Initializes the file system.*
- int **fexists** (char \*filename)  
*Tests for the existence of a file.*
- void \* **fopen\_fd** (const char \*filename, const char \*mode)  
*Helper function that opens a file descriptor.*
- unsigned int **freadbytes** (void \*buffer, int length, **FILE** \*stream)  
*Reads bytes into a buffer from a file stream.*
- unsigned int **fwritebytes** (void \*buffer, int length, **FILE** \*stream)  
*Writes bytes to a file stream.*
- unsigned long **getfreefsram** ()  
*Gets the amount of free space in the file system.*
- int **mkdir** (char \*dirname)  
*Creates a directory.*
- char **\_getkey** (void)  
*Keil-provided function.*
- char **getchar** (void)  
*Keil-provided function.*
- char **ungetchar** (char)  
*Keil-provided function.*
- char **putchar** (char)  
*Keil-provided function.*
- int **printf** (const char \*,...)  
*Keil-provided function.*
- int **sprintf** (char \*, const char \*,...)

*Keil-provided function.*

- int [vprintf](#) (const char \*, char \*)  
*Keil-provided function.*
- int [vsprintf](#) (char \*, const char \*, char \*)  
*Keil-provided function.*
- char \* [gets](#) (char \*, int n)  
*Keil-provided function.*
- int [scanf](#) (const char \*,...)  
*Keil-provided function.*
- int [sscanf](#) (char \*, const char \*,...)  
*Keil-provided function.*
- int [puts](#) (const char \*)  
*Keil-provided function.*
- unsigned int [filesystem\\_version](#) (void)  
*Returns the version number of this file system library.*

## **8.19.2 Define Documentation**

### **8.19.2.1 #define EOF -1**

Define for end-of-file.

### **8.19.2.2 #define FILE\_FLAGS\_EOF 1**

Definition for file flag. Denotes that the end of the file has been reached for this file.

**See also:**

[FILE](#)

### **8.19.2.3 #define FILE\_FLAGS\_TEMP 2**

Definition for file flag. Denotes that this is a temporary file.

**See also:**

[FILE](#)



#### 8.19.2.4 **#define FILE\_TYPE\_TINIFS 1**

Type for the file. Currently, this file system only supports the TINI File System type.

See also:

[FILE](#)

#### 8.19.2.5 **#define FILENAME\_MAX 255**

Maximum size in bytes of the longest filename string that the implementation guarantees can be opened.

See also:

[fopen](#)

#### 8.19.2.6 **#define FOPEN\_MAX 8**

Number of streams which the implementation guarantees can be open simultaneously.

See also:

[fopen](#)

#### 8.19.2.7 **#define FS\_VERSION 10**

Version number associated with this header file. Should be the same as the version number returned by the [filesystem\\_version](#) function.

See also:

[filesystem\\_version](#)

#### 8.19.2.8 **#define L\_tmpnam 20**

Maximum size of character array to hold [tmpnam](#) output.

See also:

[tmpnam](#)

#### 8.19.2.9 **#define NULL ((void \*) 0)**

Definition for a null pointer.

#### **8.19.2.10 #define P\_tmpdir "temp"**

Default directory that temporary file names will be built into.

**See also:**

[tmpnam](#)

#### **8.19.2.11 #define SEEK\_CUR 0x5555**

Seek offset is from the current location in the file.

**Warning:**

Option currently not supported.

**See also:**

[fseek](#)

[fseeko](#)

#### **8.19.2.12 #define SEEK\_END 0x5556**

Seek offset is from the end of the file.

**Warning:**

Option currently not supported.

**See also:**

[fseek](#)

[fseeko](#)

#### **8.19.2.13 #define SEEK\_SET 0x5557**

Seek offset is from the beginning of the file.

**See also:**

[fseek](#)

[fseeko](#)

#### **8.19.2.14 #define TMP\_MAX 10**

Maximum number of guaranteed unique file names that can be created by the [tmpnam](#) function.

**See also:**

[tmpnam](#)

### 8.19.3 Typedef Documentation

#### 8.19.3.1 typedef struct [file\\_structure](#) [FILE](#)

Type definition for a C file object.

#### 8.19.3.2 typedef long [fpos\\_t](#)

Type definition for the position in a file.

#### 8.19.3.3 typedef unsigned int [off\\_t](#)

Type definition for the offset in a file.

#### 8.19.3.4 typedef unsigned int [size\\_t](#)

Type definition for the amount of data to be written or read.

### 8.19.4 Function Documentation

#### 8.19.4.1 [char](#) [\\_getkey](#) ([void](#))

Keil-provided function.

This is a Keil-provided function. Please see Keil's documentation for information on this function. It is exported by this header file so that we can override [stdio.h](#).

#### 8.19.4.2 [void](#) [clearerr](#) ([FILE](#) \*[f\\_handle](#))

Clear the error indicators for a file stream.

Clears the error and end-of-file indicators for a file stream.

This function is safe to be called from multiple processes at the same time.

#### Parameters:

[f\\_handle](#) file handle to file to clear error flag for

#### 8.19.4.3 [int](#) [fcleaninit](#) ([char](#) [numfd](#), [int](#) [numblocks](#), [void](#) \*[start\\_address](#))

Initializes the file system to a blank state.

Initializes the file system. This method (or [finit](#)) must be called every time the DS80C400 boots up and wants to use the file system. Starts with a blank file system automatically.

Note that the [init\\_rom](#) function must be called before the file system is initialized.

This function is safe to be called from multiple processes at the same time.

**Parameters:**

*numfd* Maximum number of file descriptors that can be open at one time in the system.

*numblocks* Number of 256-byte blocks available to the file system.

*start\_address* Starting address of the memory allocated for the file system. The bounds of the memory allocated for the file system are then from *start\_address* to (*start\_address* + 256 \* *numblocks* + File System and Memory Manager overhead). Refer to top of this file for examples of file system memory usage.

**Returns:**

Non-zero, since the file system memory had to be erased.

**See also:**

[init\\_rom](#) [in the initialization library]  
[finit](#)

#### 8.19.4.4 **int fclose (FILE \* *f\_handle*)**

Closes the file stream.

Closes the stream associated with *f\_handle*. In the TINI File System, there are no buffers, so this function has nothing to flush before closing.

This function is safe to be called from multiple processes at the same time.

**Parameters:**

*f\_handle* handle of file to close

**Returns:**

Always 0

**See also:**

[fopen](#)

#### 8.19.4.5 **int feof (FILE \* *f\_handle*)**

Checks to see if this stream has reached the end of the file.

Tests the end-of-stream indicator for this file stream.

This function is safe to be called from multiple processes at the same time.

**Parameters:**

*f\_handle* handle to file to check end-of-file condition for

**Returns:**

Non-zero if the end of the file has been reached, otherwise 0

**8.19.4.6 int ferror (FILE \**f\_handle*)**

Gets the error indicator for the file stream.

Gets the current error indicator for the file stream.

This function is safe to be called from multiple processes at the same time.

**Parameters:**

*f\_handle* handle to file to get current error code for

**Returns:**

Current error code for file denoted by *f\_handle*. 0 means no error.

**8.19.4.7 int fexists (char \**filename*)**

Tests for the existence of a file.

Checks to see if the file *filename* exists in this file system.

This function is safe to be called from multiple processes at the same time.

**Parameters:**

*filename* File to check for the existence of.

**Returns:**

0 if the file exists, non-zero if it does not exist.

**8.19.4.8 int fflush (FILE \**f\_handle*)**

Flushes the buffers for a file stream.

The TINI File System has no buffers (data is read and written directly on the file system, since it resides in XDATA). Therefore, this function only clears the error flag.

This function is safe to be called from multiple processes at the same time.

**Parameters:**

*f\_handle* File handle to flush output buffers for

**Returns:**

0 on success.

#### 8.19.4.9 int fgetc ([FILE](#) \* *f\_handle*)

Gets the next unsigned character from the file stream.

Returns the next unsigned character (if available) from the file stream (converted to an int), advancing the file position pointer.

This function is safe to be called from multiple processes at the same time.

##### Parameters:

*f\_handle* handle of the file we will read from

##### Returns:

The next character from the file, or [EOF](#) if the end of file has been reached

##### See also:

[getc](#)  
[feof](#)  
[fputc](#)

#### 8.19.4.10 int fgetpos ([FILE](#) \* *f\_handle*, [fpos\\_t](#) \* *position*)

Gets the current value of the file position indicator.

Puts the current value of the file position indicator into the location *position*. The value in *position* after the function call is to be used for resetting the stream to this position using a later call to [fsetpos](#).

##### Warning:

This function is not multi-process safe. If two processes try to call this function at the same time, its parameters may be destroyed, yielding unpredictable results.

##### Parameters:

*f\_handle* handle to file to get current position for

*position* pointer to location for position information

##### Returns:

Always 0

##### See also:

[fsetpos](#)  
[ftell](#)

#### 8.19.4.11 **char\* fgets (char \* *string*, int *num*, FILE \* *f\_handle*)**

Reads a string from the file stream.

Reads at most *num*-1 characters from the file stream. Will not return any data read after a newline character (which is included) or the end of the file. A null character is appended to the data read.

Note that the implementation of this method is not efficient. For more efficient reading of data, use the *fread* function.

This function is safe to be called from multiple processes at the same time.

##### **Parameters:**

*string* buffer to write string data to

*num* read a maximum of (*num*-1) bytes, leaving 1 for a terminating 0

*f\_handle* handle to file to read from

##### **Returns:**

Input pointer *string*, or **NULL** if **EOF** or errors were encountered. Data will be written as 0-terminated string to *string*

##### **See also:**

*fread*

*fputs*

*feof*

#### 8.19.4.12 **unsigned int filesystem\_version (void)**

Returns the version number of this file system library.

##### **Returns:**

Version number of this FILESYSTEM library.

#### 8.19.4.13 **int finit (char *numfd*, int *numblocks*, void \* *start\_address*)**

Initializes the file system.

Initializes the file system. This method (or *fcleaninit*) must be called every time the DS80C400 boots up and wants to use the file system. If the file system does not exist or is corrupted, it will erase and start with a blank file system. Also, if any of the parameters given to *finit* do not match how the file system was previously initialized, the file system will erase and start blank.

Note that the *init\_rom* function must be called before the file system is initialized.

This function is safe to be called from multiple processes at the same time.

**Parameters:**

***numfd*** Maximum number of file descriptors that can be open at one time in the system.

***numblocks*** Number of 256-byte blocks available to the file system.

***start\_address*** Starting address of the memory allocated for the file system. The bounds of the memory allocated for the file system are then from *start\_address* to (*start\_address* + 256 \* *numblocks* + File System and Memory Manager overhead). Refer to top of this file for examples of file system memory usage.

**Returns:**

0 if the file system previously existed and was restored. Non-zero if the file system memory had to be erased.

**See also:**

[init\\_rom](#) [in the initialization library]  
[fcleaninit](#)

**8.19.4.14 void flockfile ([FILE](#) \**f\_handle*)**

Gets exclusive access to a file.

Sleeps until exclusive access to a file is available. Note that locks cannot be nested. A nested lock will be released on the very first call to [funlockfile](#), and **not** the matching call.

This function is safe to be called from multiple processes at the same time.

**Parameters:**

***f\_handle*** handle of file to acquire exclusive access for

**See also:**

[ftrylockfile](#)  
[funlockfile](#)

**8.19.4.15 [FILE](#)\* fopen (const char \**filename*, const char \**mode*)**

Opens the specified file.

Opens the file specified and associates a stream with it. Files can be opened in read, write, or append mode.

**Warning:**

This function is not multi-process safe. If two processes try to call this function at the same time, its parameters may be destroyed, yielding unpredictable results.



**Parameters:**

*filename* name of the file to get a handle for

*mode* - If mode[0] == 'r', open a reading file stream. If mode[0] == 'a', open a writing stream for appending. If mode[0] == 'w', open a writing stream for a blank file.

**Returns:**

handle to the file, or [NULL](#) on failure

**See also:**

[freopen](#)

[fclose](#)

**8.19.4.16 void\* fopen\_fd (const char \**filename*, const char \**mode*)**

Helper function that opens a file descriptor.

Helper function that opens a file descriptor. File descriptors are not immediately useful to any C library function. Applications should use the [fopen](#) function to open a file.

**Warning:**

This function is not multi-process safe. If two processes try to call this function at the same time, its parameters may be destroyed, yielding unpredictable results.

**Parameters:**

*filename* Name of the file to get a descriptor for. The data pointed to by *filename* must stay consistent for the duration of the use of the file descriptor. The [fopen](#) method avoids this limitation by creating a copy of the name data.

*mode* Read/Write/Append mode string

**Returns:**

pointer to a file descriptor

**See also:**

[fopen](#)

**8.19.4.17 int fputc (int *ch*, [FILE](#) \**f\_handle*)**

Writes a character to a file stream.

Writes the specified character (converted from an int) to a file stream, advancing the file position indicator.

This function is safe to be called from multiple processes at the same time.

**Parameters:**

*ch* character that will be written to the file *f\_handle*

*f\_handle* handle of the file we will write character to

**Returns:**

Character written if successful, else EOF

**See also:**

[fgetc](#)

[putc](#)

**8.19.4.18 int fputs (const char \* *str*, FILE \* *f\_handle*)**

Writes a string to a file stream.

Writes a null-terminated string to a file stream. The terminating character is not written.

**Warning:**

This function is not multi-process safe. If two processes try to call this function at the same time, its parameters may be destroyed, yielding unpredictable results.

**Parameters:**

*str* null-terminated string to write to a file

*f\_handle* handle to file to write string to

**Returns:**

number of bytes written, or EOF on failure

**See also:**

[fgets](#)

[fwrite](#)

**8.19.4.19 size\_t fread (void \* *ptr*, size\_t *size*, size\_t *num*, FILE \* *f\_handle*)**

Read a number of bytes from a file stream.

Reads a block of data from a file stream. This function allows you to read *num* elements of size *size*. However, note that this function always behaves as if it had been called by:

```
fread(ptr, 1, size*num, f_handle);
```

This function is safe to be called from multiple processes at the same time.

**Parameters:**

*ptr* pointer to buffer to read data into  
*size* size of each element to be read  
*num* number of elements to read  
*f\_handle* handle to file to read from

**Returns:**

number of elements read

**See also:**

[fgetc](#)  
[fwrite](#)

**8.19.4.20 unsigned int freadbytes (void \* *buffer*, int *length*, [FILE](#) \* *stream*)**

Reads bytes into a buffer from a file stream.

Reads a specified number of bytes into a buffer from a file stream. This function is used by [fread](#) as a helper function. It may safely be used from user applications, although it is not a standard file reading function (is not part of an ANSI-C standard library).

**Warning:**

This function is not multi-process safe. If two processes try to call this function at the same time, its parameters may be destroyed, yielding unpredictable results.

**Parameters:**

*buffer* Location to read data into  
*length* Number of bytes to read  
*stream* File to read data from

**Returns:**

Number of bytes read, or [EOF](#) if the end of file is reached.

**See also:**

[fread](#)  
[fwritebytes](#)

**8.19.4.21 [FILE](#)\* freopen (const char \* *newfilename*, const char \* *mode*, [FILE](#) \* *old\_handle*)**

Associates an open stream with a different file.

Closes the file associated with *old\_handle* and opens a stream to the file *newfilename*.

This function is safe to be called from multiple processes at the same time.

**Parameters:**

*newfilename* name of file to open  
*mode* mode to open *newfilename* in (see *fopen* for details)  
*old\_handle* file handle to flush and close

**Returns:**

Handle to file *newfilename*, or **NULL** if the file could not be opened.

**See also:**

[fopen](#)  
[fclose](#)

**8.19.4.22 int fseek (FILE \*f\_handle, long int offset, int tag)**

Sets the file position indicator.

Sets the file position indicator for a file stream. Note that the only currently supported value for *tag* is **SEEK\_SET**, meaning that the value *offset* will always be interpreted as the offset from the beginning of the file.

After a call to *fseek*, the end-of-file indicator for the file stream is reset.

This function behaves the same as *fseeko*. The only difference is that *fseeko* accepts an *offset* parameter of type **off\_t**.

**Warning:**

This function is not multi-process safe. If two processes try to call this function at the same time, its parameters may be destroyed, yielding unpredictable results.

**Parameters:**

*f\_handle* handle of file to set position for  
*offset* offset to set for file position  
*tag* only **SEEK\_SET** is supported

**Returns:**

Always 0.

**See also:**

[ftell](#)  
[fseeko](#)  
[fsetpos](#)

#### 8.19.4.23 `int fseeko (FILE * f_handle, off_t offset, int tag)`

Sets the file position indicator.

Sets the file position indicator for a file stream. Note that the only currently supported value for *tag* is `SEEK_SET`, meaning that the value *offset* will always be interpreted as the offset from the beginning of the file.

After a call to `fseeko`, the end-of-file indicator for the file stream is reset.

This function behaves the same as `fseek`. The only difference is that `fseeko` accepts an *offset* parameter of type **long int**.

This function is safe to be called from multiple processes at the same time.

##### Parameters:

*f\_handle* handle of file to set position for

*offset* offset to set for file position

*tag* only `SEEK_SET` is supported

##### Returns:

Always 0.

##### See also:

`ftello`

`fseek`

`fsetpos`

#### 8.19.4.24 `int fsetpos (FILE * f_handle, const fpos_t * position)`

Sets the file position indicator.

Sets a stream's file position indicator from the position information pointed to by *position*. The value in *position* should have been obtained by a call to `fgetpos`. If successful, this function will also clear the end-of-file indicator for the stream.

##### Warning:

This function is not multi-process safe. If two processes try to call this function at the same time, its parameters may be destroyed, yielding unpredictable results.

##### Parameters:

*f\_handle* handle of file we will set the position for

*position* position in the file to set

##### Returns:

Always 0

**See also:**

[fgetpos](#)

[fseek](#)

#### **8.19.4.25 long ftell (FILE \**f\_handle*)**

Gets the file position indicator.

Gets the file position indicator for the specified file. This is the number of characters from the beginning of the file.

This function behaves the same as [ftello](#). The only difference is that [ftello](#) returns a value of type [off\\_t](#).

This function is safe to be called from multiple processes at the same time.

**Parameters:**

*f\_handle* handle of file to get current position of

**Returns:**

Current position in file, or -1L on failure.

**See also:**

[fseek](#)

[ftello](#)

[fgetpos](#)

#### **8.19.4.26 off\_t ftello (FILE \**f\_handle*)**

Gets the file position indicator.

Gets the file position indicator for the specified file. This is the number of characters from the beginning of the file.

This function behaves the same as [ftell](#). The only difference is that [ftell](#) returns a value of type **long**.

This function is safe to be called from multiple processes at the same time.

**Parameters:**

*f\_handle* handle of file to get current position of

**Returns:**

Current position in file, or -1L on failure.

**See also:**

[fseek](#)

[ftello](#)

[fgetpos](#)

#### 8.19.4.27 **int ftrylockfile** (**FILE** \**f\_handle*)

Tries to get exclusive access to a file.

Obtains exclusive access to a file if it is available. Otherwise, returns without waiting for exclusive access. Note that locks cannot be nested. A nested lock will be released on the very first call to [funlockfile](#), and **not** the matching call.

This function is safe to be called from multiple processes at the same time.

##### **Parameters:**

*f\_handle* handle to file we will try to get exclusive access to

##### **Returns:**

0 if the file was locked, non-zero if someone else has the lock

##### **See also:**

[flockfile](#)  
[funlockfile](#)

#### 8.19.4.28 **void funlockfile** (**FILE** \**f\_handle*)

Release exclusive access on a file.

Releases exclusive access that was earlier acquired on this file using [flockfile](#) or [ftrylockfile](#). Note that locks cannot be nested. This function will release all locks that the current thread/process have on the file.

This function is safe to be called from multiple processes at the same time.

##### **Parameters:**

*f\_handle* handle to file to release exclusive access for

##### **See also:**

[flockfile](#)  
[ftrylockfile](#)

#### 8.19.4.29 **size\_t fwrite** (const void \**ptr*, **size\_t** *size*, **size\_t** *num*, **FILE** \**f\_handle*)

Write a number of bytes to a file stream.

Writes a block of data to a file stream. This function allows you to write *num* elements of size *size*. However, note that this function always behaves as if it had been called by:

```
fwrite(ptr, 1, size*num, f_handle);
```

This function is safe to be called from multiple processes at the same time.

**Parameters:**

*ptr* pointer to buffer of data to be written

*size* size of each element to be written

*num* number of elements to write

*f\_handle* handle to file to write to

**Returns:**

number of elements written

**See also:**

[fputc](#)

[fread](#)

**8.19.4.30 unsigned int fwritebytes (void \* *buffer*, int *length*, [FILE](#) \* *stream*)**

Writes bytes to a file stream.

Writes the specified number of bytes to a file stream. This function is used by [fwrite](#) as a helper function. It may safely be used from user applications, although it is not a standard file writing function (is not part of an ANSI-C standard library).

**Warning:**

This function is not multi-process safe. If two processes try to call this function at the same time, its parameters may be destroyed, yielding unpredictable results.

**Parameters:**

*buffer* Location to write data from

*length* Number of bytes to write

*stream* File to write data to

**Returns:**

Number of bytes written, or [EOF](#) if an error occurred

**See also:**

[fwrite](#)

[freadbytes](#)



#### 8.19.4.31 int getc (**FILE** \**f\_handle*)

Gets the next unsigned character from the file stream.

Returns the next unsigned character (if available) from the file stream (converted to an int), advancing the file position pointer. Note: This function is equivalent to *fgetc*.

This function is safe to be called from multiple processes at the same time.

##### Parameters:

*f\_handle* handle of the file we will read from

##### Returns:

The next character from the file, or **EOF** if the end of file has been reached

##### See also:

[fgetc](#)  
[feof](#)  
[putc](#)

#### 8.19.4.32 char getchar (void)

Keil-provided function.

This is a Keil-provided function. Please see Keil's documentation for information on this function. It is exported by this header file so that we can override *stdio.h*.

#### 8.19.4.33 unsigned long getfreefsram ()

Gets the amount of free space in the file system.

Returns the number of bytes available to the file system. Note that this number is completely independent of the amount of free RAM available from the ROM's memory manager. The TINI File System uses its own independent memory manager.

This function is safe to be called from multiple processes at the same time.

##### Returns:

Amount of free RAM available to the file system.

#### 8.19.4.34 char\* gets (char \*, int *n*)

Keil-provided function.

This is a Keil-provided function. Please see Keil's documentation for information on this function. It is exported by this header file so that we can override *stdio.h*.

#### 8.19.4.35 int mkdir (char \* *dirname*)

Creates a directory.

Creates a directory with the specified directory name.

This function is safe to be called from multiple processes at the same time.

#### Returns:

non-zero on success, 0 on failure

#### 8.19.4.36 int printf (const char \*, ...)

Keil-provided function.

This is a Keil-provided function. Please see Keil's documentation for information on this function. It is exported by this header file so that we can override [stdio.h](#).

#### 8.19.4.37 int putc (int *value*, FILE \* *f\_handle*)

Writes a character to a file stream.

Writes the specified character (converted from an int) to a file stream, advancing the file position indicator. Note: This function is equivalent to [fputc](#).

This function is safe to be called from multiple processes at the same time.

#### Parameters:

*value* character that will be written to the file *f\_handle*

*f\_handle* handle of the file we will write character to

#### Returns:

Character written if successful, else [EOF](#)

#### See also:

[getc](#)

[fputc](#)

#### 8.19.4.38 char putchar (char)

Keil-provided function.

This is a Keil-provided function. Please see Keil's documentation for information on this function. It is exported by this header file so that we can override [stdio.h](#).

#### 8.19.4.39 `int puts (const char *)`

Keil-provided function.

This is a Keil-provided function. Please see Keil's documentation for information on this function. It is exported by this header file so that we can override [stdio.h](#).

#### 8.19.4.40 `int remove (const char * filename)`

Removes a file from the file system.

Deletes the file specified by *filename*.

This function is safe to be called from multiple processes at the same time.

##### Parameters:

*filename* file name that will be deleted

##### Returns:

0 on success, non-zero on failure

##### See also:

[rename](#)

#### 8.19.4.41 `int rename (const char * oldname, const char * newname)`

Renames a file.

Renames the file identified by *oldname* to now be identified by *newname*.

##### Warning:

This function is not multi-process safe. If two processes try to call this function at the same time, its parameters may be destroyed, yielding unpredictable results.

##### Parameters:

*oldname* filename of the file that will change names

*newname* new name for the file called *oldname*

##### Returns:

0 on success, non-zero on failure

##### See also:

[remove](#)

#### 8.19.4.42 void rewind (**FILE** \**f\_handle*)

Resets the file position indicator for a stream.

Sets the file position indicator for the stream to the beginning of the file. It also resets the end of file condition. This is functionally equivalent to:

```
fseek(f_handle, 0, SEEK_SET);  
clearerr(f_handle);
```

This function is safe to be called from multiple processes at the same time.

#### Parameters:

*f\_handle* handle to file that the streams will be reset to the beginning for

#### See also:

[fseek](#)  
[fsetpos](#)

#### 8.19.4.43 int scanf (const char \*, ...)

Keil-provided function.

This is a Keil-provided function. Please see Keil's documentation for information on this function. It is exported by this header file so that we can override [stdio.h](#).

#### 8.19.4.44 int sprintf (char \*, const char \*, ...)

Keil-provided function.

This is a Keil-provided function. Please see Keil's documentation for information on this function. It is exported by this header file so that we can override [stdio.h](#).

#### 8.19.4.45 int sscanf (char \*, const char \*, ...)

Keil-provided function.

This is a Keil-provided function. Please see Keil's documentation for information on this function. It is exported by this header file so that we can override [stdio.h](#).

#### 8.19.4.46 char\* tempnam (const char \**dirname*, const char \**pfx*)

Generates a path/filename that can be used for a temporary file.

Generates a path/filename that can be used to create a temporary file with. The pointer that is returned is suitable to be freed using [mem\\_free](#). Make sure to use the Dallas Semiconductor memory management library ([rom400\\_mem.h](#)) rather than the Keil memory manager to free the memory.

This function is safe to be called from multiple processes at the same time.

**Parameters:**

*dirname* Directory for temporary file name to be created for. A default directory will be used if *dirname* is null.

*pfx* Prefix to prepend to temporary file name

**Returns:**

Pointer to temporary file name. Use [mem\\_free](#) to delete the memory.

**See also:**

[tmpnam](#)

[tmpfile](#)

[mem\\_free](#) [in the memory manager library]

#### 8.19.4.47 [FILE\\*](#) **tmpfile** (void)

Generates a stream to a temporary file.

Generates a stream to a temporary file, opened for writing/update.

This function is safe to be called from multiple processes at the same time.

**Returns:**

File handle to a temporary file, or [NULL](#) on failure.

**See also:**

[tempnam](#)

[tmpnam](#)

#### 8.19.4.48 [char\\*](#) **tmpnam** (char \* *nametarget*)

Generates a unique temporary filename.

Capable of generating [TMP\\_MAX](#) unique temporary filenames. This filename is suitable for using in a call to [fopen](#). If the name is written to a static location, then this call destroys the previous filename stored in that location.

This function is safe to be called from multiple processes at the same time.

**Parameters:**

*nametarget* Storage location for new temporary name. If [NULL](#), the temporary name will be copied to a static location.

**Returns:**

Location where temporary name is stored. This may be the same as *nametarget*.

See also:

[tempnam](#)

[tmpfile](#)

#### 8.19.4.49 char ungetchar (char)

Keil-provided function.

This is a Keil-provided function. Please see Keil's documentation for information on this function. It is exported by this header file so that we can override [stdio.h](#).

#### 8.19.4.50 int vprintf (const char \*, char \*)

Keil-provided function.

This is a Keil-provided function. Please see Keil's documentation for information on this function. It is exported by this header file so that we can override [stdio.h](#).

#### 8.19.4.51 int vsprintf (char \*, const char \*, char \*)

Keil-provided function.

This is a Keil-provided function. Please see Keil's documentation for information on this function. It is exported by this header file so that we can override [stdio.h](#).

## 8.20 tini400\_canbus.h File Reference

### 8.20.1 Detailed Description

CAN Bus Interrupt Driver for DS80C390 / 400.

This library provides an interrupt driven interface for the CAN peripherals on the DS80C390 / 400 Microcontroller. This driver allows applications to asynchronously transmit & receive CAN data while the applications perform other processing.

For detailed information on the DS80C400 please see the [High-Speed Microcontroller User's Guide: DS80C400 Supplement](#).

The CAN library consists of four sub modules:

**Initialization module** initializes the global data structures, message centers, transmit & receive buffers and interrupt handlers.

**Configuration Module** enables the application to configure the CAN peripheral and the individual message centers.

**Data Access Module** provides interface to application to transmit & receive data.

**Interrupt handlers** contains interrupt handlers for all CAN interrupts.

## Data Structures

- struct [CanFrame](#)  
*CAN Frame structure. Denotes the structure of a Transmitted or received CAN frame.*
- struct [MCConfig](#)  
*CAN Message center configuration structure. Used for configuration of receive parameters of Message Centers.*

## Defines

- #define [TRUE](#) 1  
*TRUE.*
- #define [FALSE](#) 0  
*FALSE.*
- #define [TINI400\\_CANBUS\\_VERSION](#) 3
- #define [CAN\\_ERROR\\_NOERROR](#) 0
- #define [CAN\\_ERROR\\_GENERIC](#) -1
- #define [CAN\\_ERROR\\_BUSOFF](#) -2
- #define [CAN\\_ERROR\\_TIMEOUT](#) -3
- #define [CAN\\_ERROR\\_NOT\\_INITIALIZED](#) -4
- #define [CAN\\_ERROR\\_ARGUMENT](#) -5
- #define [CAN\\_ERROR\\_PORT\\_ENABLED](#) -6
- #define [CAN\\_ERROR\\_PORT\\_DISABLED](#) -7
- #define [CAN\\_ERROR\\_MC\\_ACTIVE](#) -8
- #define [CAN\\_ERROR\\_BIT\\_STUFF](#) -9
- #define [CAN\\_ERROR\\_FORMAT](#) -10
- #define [CAN\\_ERROR\\_TRANSMIT\\_NO\\_ACK](#) -11
- #define [CAN\\_ERROR\\_BIT\\_ONE](#) -12
- #define [CAN\\_ERROR\\_BIT\\_ZERO](#) -13
- #define [CAN\\_ERROR\\_CRC](#) -14
- #define [CAN\\_ERROR\\_COUNT\\_EXCEEDED](#) -15
- #define [CAN\\_ERROR\\_NOFREEMC](#) -16
- #define [CAN\\_ERROR\\_BUFFULL](#) -17
- #define [CAN\\_ERROR\\_BUFEMPTY](#) -18
- #define [CAN\\_ERROR\\_INVALID\\_TSEG](#) -19
- #define [CAN\\_ERROR\\_FRAMESDROPPED](#) -20

## Typedefs

- typedef signed char [int8\\_t](#)  
*8 bit signed integer*
- typedef unsigned char [uint8\\_t](#)  
*8 bit unsigned integer*
- typedef signed int [int16\\_t](#)  
*16 bit signed integer*
- typedef unsigned int [uint16\\_t](#)  
*16 bit unsigned integer*
- typedef signed long [int32\\_t](#)  
*32 bit signed integer*
- typedef unsigned long [uint32\\_t](#)  
*32 bit unsigned integer*
- typedef unsigned char [boolean](#)  
*boolean*

## Functions

- [uint16\\_t can\\_version](#) (void)  
*Returns the version number of this CAN library. this function is safe to be called from multiple processes at the same time.*
- void [can\\_init](#) (void)  
*Initializes CAN library.*
- [int8\\_t can\\_resetcontroller](#) (uint8\_t CAN\_No)  
*Resets CAN controller.*
- [int8\\_t can\\_setsiestamode](#) (uint8\_t CAN\_No)  
*Puts the CAN Controller in SIESTA (low power) mode.*
- [int8\\_t can\\_disablecontroller](#) (uint8\_t CAN\_No)  
*Disables the CAN controller.*
- [int8\\_t can\\_enablecontroller](#) (uint8\_t CAN\_No)



*Enables the CAN controller.*

- `int8_t can_enablecontrollerpassive (uint8_t CAN_No)`  
*Enables the CAN controller, but doesn't connect CAN transmit to the bus.*
- `int8_t can_setrxwriteoverenable (uint8_t CAN_No, boolean writeover)`  
*Sets the state of write over in the receiver buffer.*
- `int8_t can_set11bitglobalidmask (uint8_t CAN_No, uint32_t *mask)`  
*Sets the 11 bit Standard Global Id Mask.*
- `int8_t can_set29bitglobalidmask (uint8_t CAN_No, uint32_t *mask)`  
*Sets the 29 bit Standard Global Id Mask.*
- `int8_t can_set11bitmessagecenter15idmask (uint8_t CAN_No, uint32_t *mask)`  
*Sets the global 11 Bit Message Center 15 ID Mask.*
- `int8_t can_set29bitmessagecenter15idmask (uint8_t CAN_No, uint32_t *mask)`  
*Sets the global 29 Bit Message Center 15 ID Mask.*
- `int8_t can_setmediaidmask (uint8_t CAN_No, uint16_t mask)`  
*Sets the global media ID mask.*
- `int8_t can_setmediaidarbitration (uint8_t CAN_No, uint16_t value)`  
*Sets the global media ID arbitration.*
- `int8_t can_setbaudrateprescaler (uint8_t CAN_No, uint16_t prescaler)`  
*Sets the basic time quantum (t<sub>qu</sub>) necessary for CAN communication.*
- `int8_t can_setsynchronizationjumpwidth (uint8_t CAN_No, uint8_t jump-Width)`  
*Sets the Synchronization Jump Width necessary for adjusting TSEG1 and TSEG2.*
- `int8_t can_setsamplerate (uint8_t CAN_No, uint8_t sampleRate)`  
*Sets the sample rate which is whether to use one or three samples per bit time during CAN communication.*
- `int8_t can_settseg1 (uint8_t CAN_No, uint8_t tseg1)`  
*Sets Timing Segment 1 to a specified number of time quanta.*
- `int8_t can_settseg2 (uint8_t CAN_No, uint8_t tseg2)`

*Sets Timing Segment 1 to a specified number of time quanta.*

- `int8_t can_enablemessagecenter (uint8_t CAN_No, uint8_t messageCenter)`  
*Puts the message center into Active mode if disabled.*
- `int8_t can_disablemessagecenter (uint8_t CAN_No, uint8_t messageCenter)`  
*Puts the message center into Disabled mode if active.*
- `int8_t can_freemessagecenter (uint8_t CAN_No, uint8_t messageCenter)`  
*Returns the message center to the free pool.*
- `int8_t can_setmessagecentertx (uint8_t CAN_No, uint8_t messageCenter)`  
*Sets Tx/Rx bit of a specific message center to 1 (transmit).*
- `int8_t can_setmessagecenterrx (uint8_t CAN_No, uint8_t messageCenter)`  
*Sets Tx/Rx bit of a specific message center to 1 (receive).*
- `int8_t can_set11bitmessagecenterarbitrationid (uint8_t CAN_No, uint8_t messageCenter, uint32_t *ID)`  
*Sets the 11 bit Arbitration ID.*
- `int8_t can_set29bitmessagecenterarbitrationid (uint8_t CAN_No, uint8_t messageCenter, uint32_t *ID)`  
*Sets the 29 bit Arbitration ID.*
- `int8_t can_setmessagecentermessageidmaskenable (uint8_t CAN_No, uint8_t messageCenter, boolean maskEnable)`  
*Enables or disables Message ID Masking for a specific message center.*
- `int8_t can_setmessagecentermediaidmaskenable (uint8_t CAN_No, uint8_t messageCenter, boolean maskEnable)`  
*Enables or disables Media ID Masking for a specific message center.*
- `int8_t can_sendframe (uint8_t CAN_No, CanFrame *frame)`  
*Transmits a data or RFR frame.*
- `int8_t can_getrxmessagecenter (uint8_t CAN_No, MConfig *config)`  
*Gets the first available message centre and configure it for reception.*
- `int8_t can_receiveframesavailable (uint8_t CAN_No)`  
*Gets the number of frames pending in the receive buffer.*
- `int8_t can_receiveframe (uint8_t CAN_No, CanFrame *frame)`

*Gets a frame from the receive buffer.*

- `int8_t can_getautoanswerrfrmessagecenter (uint8_t CAN_No, CanFrame *frame)`

*Gets the first available message centre and configure it for Auto-answering Remote RFRs.*

- `int16_t can_gettxerrorcount (uint8_t CAN_No)`

*Gets the transmitter error count.*

- `int16_t can_getrxerrorcount (uint8_t CAN_No)`

*Gets the receiver error count.*

## **8.20.2 Define Documentation**

### **8.20.2.1 #define CAN\_ERROR\_ARGUMENT -5**

Improper argument.

### **8.20.2.2 #define CAN\_ERROR\_BIT\_ONE -12**

Bit 1 error.

### **8.20.2.3 #define CAN\_ERROR\_BIT\_STUFF -9**

Bit stuff error. CAN controller detected more than five consecutive bits of an identical state in an incoming message.

### **8.20.2.4 #define CAN\_ERROR\_BIT\_ZERO -13**

Bit 0 error.

### **8.20.2.5 #define CAN\_ERROR\_BUFEMPTY -18**

Receiver buffer is empty.

### **8.20.2.6 #define CAN\_ERROR\_BUFFULL -17**

Transmitter buffer is full.

### **8.20.2.7 #define CAN\_ERROR\_BUSOFF -2**

CAN Bus off error. Transmit error count has reached or exceeded 256.

#### **8.20.2.8 #define CAN\_ERROR\_COUNT\_EXCEEDED -15**

Transmit or Receiver Error counter has exceeded the error count of 96 (in case of ERCS bit of C0C register is 0) or reached 128 (in case of ERCS bit of C0C register is 1).

#### **8.20.2.9 #define CAN\_ERROR\_CRC -14**

CRC error. The calculated CRC of the received message message does not match the CRC embedded in the message.

#### **8.20.2.10 #define CAN\_ERROR\_FORMAT -10**

Format error. Received message has the wrong format.

#### **8.20.2.11 #define CAN\_ERROR\_FRAMESDROPPED -20**

One or more frames have been dropped. In this case also, valid frames are returned from the receiver buffer. User can chose to ignore this error and treat it as CAN\_ERROR\_NOERROR.

#### **8.20.2.12 #define CAN\_ERROR\_GENERIC -1**

Message Center is not in disabled mode.

#### **8.20.2.13 #define CAN\_ERROR\_INVALID\_TSEG -19**

Time Segment value not set.

#### **8.20.2.14 #define CAN\_ERROR\_MC\_ACTIVE -8**

Message Center Active.

#### **8.20.2.15 #define CAN\_ERROR\_NOERROR 0**

No Error

#### **8.20.2.16 #define CAN\_ERROR\_NOFREEMC -16**

No Free Message Center available.

#### **8.20.2.17 #define CAN\_ERROR\_NOT\_INITIALIZED -4**

Can controller not initialized.

#### 8.20.2.18 `#define CAN_ERROR_PORT_DISABLED -7`

CAN Port disabled.

#### 8.20.2.19 `#define CAN_ERROR_PORT_ENABLED -6`

CAN Port enabled.

#### 8.20.2.20 `#define CAN_ERROR_TIMEOUT -3`

Time out error.

#### 8.20.2.21 `#define CAN_ERROR_TRANSMIT_NO_ACK -11`

Transmit not acknowledged error. Requested node did not acknowledge the sent message.

#### 8.20.2.22 `#define TINI400_CANBUS_VERSION 3`

Version number associated with this header file. Should be the same as the version number returned by the [can\\_version](#) function.

See also:

[can\\_version](#)

## 8.21 `tini400_crypt.h` File Reference

### 8.21.1 Detailed Description

SHA-1 and MD4 functions for the DS80C400.

This library contains functions that compute the SHA-1 hash and MD4 hash of a byte array.

For detailed information on the DS80C400 please see the [High-Speed Microcontroller User's Guide: DS80C400 Supplement](#).

#### Warning:

The functions in this library are **NOT** multi-process safe—that is, if you call the same method from two different processes at the same time, the parameters to the function may be destroyed, yielding unpredictable results. Consult each individual function's documentation for details on which functions are multi-process safe.

#### Defines

- `#define TINI400_CRYPT_VERSION 3`

## Functions

- unsigned int `crypt_version` (void)  
*Returns the version number of this CRYPT library.*
- void `crypt_sha1` (short inLength, void \*inBuff, void \*outBuff)  
*Computes a SHA-1 hash on the given message.*
- void `crypt_md4` (unsigned char \*out, unsigned char \*in, int n)  
*Computes a MD4 hash on the given message.*

### 8.21.2 Define Documentation

#### 8.21.2.1 #define TINI400\_CRYPT\_VERSION 3

Version number associated with this header file. Should be the same as the version number returned by the `crypt_version` function.

See also:

`crypt_version`

### 8.21.3 Function Documentation

#### 8.21.3.1 void crypt\_md4(unsigned char \*out, unsigned char \*in, int n)

Computes a MD4 hash on the given message.

See RFC 1320 for more information. WARNING! MD4 has known cryptographic weaknesses. Where possible, SHA-1 should be used instead.

##### Parameters:

- out* holds the hash value on return (16 bytes)
- in* the message to hash
- n* length of the message to hash

#### 8.21.3.2 void crypt\_sha1(short inLength, void \*inBuff, void \*outBuff)

Computes a SHA-1 hash on the given message.

See FIPS 180-1 for more information on SHA-1.

##### Parameters:

- inLength* length of the message to hash
- inBuff* the message to hash
- outBuff* holds the hash value on return (20 bytes minimum)

### 8.21.3.3 unsigned int crypt\_version (void)

Returns the version number of this CRYPT library.

#### Returns:

Version number of this CRYPT library.

## 8.22 tini400\_debugport.h File Reference

### 8.22.1 Detailed Description

Functions supporting the debug port on the TINIs400 module.

This library contains functions that write to the debug port on the TINIs400. More information on the debug port can be found in the application note 614, *Diagnostic Port for the TINIs400*, found at <http://pdfserv.maxim-ic.com/en/an/app614.pdf>.

For detailed information on the TINIs400 debug port please see [Application Note 614: Diagnostic Port for the TINIs400](#).

#### Warning:

The functions in this library are **NOT** multi-process safe—that is, if you call the same method from two different processes at the same time, the parameters to the function may be destroyed, yielding unpredictable results. Consult each individual function’s documentation for details on which functions are multi-process safe.

#### Defines

- #define [TINI400\\_DEBUGPORT\\_VERSION](#) 2

#### Functions

- unsigned int [debugport\\_version](#) (void)  
*Returns the version number of this DEBUGPORT library.*
- void [debugport\\_init](#) (void)  
*Initializes the timing for the debug port.*
- void [debugport\\_sendbyte](#) (unsigned char ch)  
*Sends a character to the debug port.*
- void [debugport\\_sendhex](#) (unsigned char b)  
*Prints a hexadecimal value to the debug port.*

- void [debugport\\_sendstring](#) (unsigned char \*s)

*Sends a string to the debug port.*

## 8.22.2 Define Documentation

### 8.22.2.1 #define TINI400\_DEBUGPORT\_VERSION 2

Version number associated with this header file. Should be the same as the version number returned by the [debugport\\_version](#) function.

See also:

[debugport\\_version](#)

## 8.22.3 Function Documentation

### 8.22.3.1 void debugport\_init (void)

Initializes the timing for the debug port.

This function must be called after init\_rom before the debug port can be used. For correct serial port timing, set the clock frequency using [init\\_setfrequency\(\)](#).

### 8.22.3.2 void debugport\_sendbyte (unsigned char ch)

Sends a character to the debug port.

This function sends a character to the debug port at 115200 bps. Note: This function disables interrupts while sending the character.

### 8.22.3.3 void debugport\_sendhex (unsigned char b)

Prints a hexadecimal value to the debug port.

This function converts a byte into hexadecimal and sends the result to the debug port at 115200 bps. Note: This function disables interrupts while sending each character.

### 8.22.3.4 void debugport\_sendstring (unsigned char \* s)

Sends a string to the debug port.

This function sends a zero-terminated string to the debug port at 115200 bps. Note: This function disables interrupts while sending each character.



### 8.22.3.5 unsigned int debugport\_version (void)

Returns the version number of this DEBUGPORT library.

#### Returns:

Version number of this DEBUGPORT library.

## 8.23 tini400\_dns.h File Reference

### 8.23.1 Detailed Description

DNS Client functions for the DS80C400 ROM.

This library contains functions for resolving a host name to an IP address that is usable by the silicon software for making socket function calls. Note that the functions in this library are not safe to be called from multiple processes at the same time. The functions in this library store their results in static memory locations, and must be retrieved and stored in alternate locations before further DNS operations are performed.

Note that as of version 3, this library has been changed to use the system-wide DNS server entries, which might be set by the DHCP client (from data recieved in a DHCP response). Applications can make sure they have a valid server entry by making sure the DNS server IP addresses are not all 0's, since the ROM initialization functions clear the DNS server entries.

For detailed information on the DS80C400 please see the [High-Speed Microcontroller User's Guide: DS80C400 Supplement](#).

#### Warning:

The functions in this library are **NOT** multi-process safe—that is, if you call the same method from two different processes at the same time, the parameters to the function may be destroyed, yielding unpredictable results.

The functions in this library use String functions such as **sprintf** for data formatting, which are not multiprocess safe. Care must be taken that DNS functions do not operate at the same time as other string formatting operations.

```
#include <stdlib.h>
```

#### Data Structures

- struct [hostent](#)
- struct [mailhostent](#)

#### Defines

- #define [TINI400\\_DNS\\_VERSION](#) 7

## Functions

- `hostent * gethostbyaddr` (void \*addr, `size_t` len, int type)  
*Looks up information on a host given an IP address.*
- `hostent * gethostbyname` (char \*name)  
*Looks up information on a host given a host name.*
- void `dns_init` (void)  
*Initializes the DNS client code.*
- void `dns_settimeout` (unsigned long t)  
*Sets the socket timeout value used for DNS server communications.*
- void `dns_setmaxtimeout` (unsigned long t)  
*Sets the maximum socket timeout value used for DNS server communications.*
- unsigned long `dns_gettimeout` (void)  
*Gets the socket timeout value used for DNS server communications.*
- void `dns_getprimary` (struct `sockaddr` \*sa)  
*Gets the address of the primary DNS server.*
- void `dns_setprimary` (struct `sockaddr` \*sa)  
*Sets the address of the primary DNS server.*
- void `dns_getsecondary` (struct `sockaddr` \*sa)  
*Gets the address of the secondary DNS server.*
- void `dns_setsecondary` (struct `sockaddr` \*sa)  
*Sets the address of the secondary DNS server.*
- `mailhostent * dns_getmx` (char \*name)  
*Performs a DNS MX record lookup.*
- void `dns_enableipv6queries` (unsigned char enable)  
*Enables/disables attempts to make IPv6 DNS queries.*
- unsigned int `dns_version` ()  
*Returns the version number of this DNS client library.*

## 8.23.2 Define Documentation

### 8.23.2.1 #define TINI400\_DNS\_VERSION 7

Version number associated with this header file. Should be the same as the version number returned by the [dns\\_version](#) function.

See also:

[dns\\_version](#)

## 8.23.3 Function Documentation

### 8.23.3.1 void dns\_enableipv6queries (unsigned char *enable*)

Enables/disables attempts to make IPv6 DNS queries.

Use an *enable* value of 0 to disable attempts to perform IPv6 queries. Disabling IPv6 queries can dramatically increase the speed of the library routines. Use an *enable* value of non-zero to enable IPv6 DNS queries.

**Parameters:**

*enable* 0 to disable IPv6 DNS queries, non-zero to enable

### 8.23.3.2 struct [mailhostent](#)\* dns\_getmx (char \* *name*)

Performs a DNS MX record lookup.

MX records are mail exchanger records. In order to send an email without using a mail relay (mail host), you need to look up the MX record of the remote domain and then open the SMTP connection to the address returned by [dns\\_getmx\(\)](#).

**Parameters:**

*name* domain to look up.

**Returns:**

DNS response(s) or NULL for failed lookup. If any valid data is returned, the first invalid [mailhostent](#) entry will have NULL for a host name.

See also:

[mailhostent](#)

### 8.23.3.3 void dns\_getprimary (struct [sockaddr](#) \* *sa*)

Gets the address of the primary DNS server.

Fills in an address structure with the IP of the secondary DNS server used by this DNS client code. DNS operations first try to use a server designated as primary, and the use a server designated as secondary if the primary fails to return results.

Note that this gets the system's primary DNS server setting. This may have been set by the DHCP client or by previous calls to [dns\\_setprimary](#). This function is equivalent to [dhcp\\_getprimarydns](#).

**Parameters:**

*sa* will be filled in with the address of the primary DNS server

**See also:**

[dns\\_setprimary](#)  
[dns\\_setsecondary](#)  
[dns\\_getsecondary](#)  
[dhcp\\_getprimarydns](#)

**8.23.3.4 void dns\_getsecondary (struct [sockaddr](#) \* *sa*)**

Gets the address of the secondary DNS server.

Fills in an address structure with the IP of the secondary DNS server used by this DNS client code. DNS operations first try to use a server designated as primary, and the use a server designated as secondary if the primary fails to return results.

Note that this gets the system's secondary DNS server setting. This may have been set by the DHCP client or by previous calls to [dns\\_setsecondary](#). This function is equivalent to [dhcp\\_getsecondarydns](#).

**Parameters:**

*sa* will be filled in with the address of the secondary DNS server

**See also:**

[dns\\_setprimary](#)  
[dns\\_getprimary](#)  
[dns\\_setsecondary](#)  
[dhcp\\_getsecondarydns](#)

**8.23.3.5 unsigned long dns\_gettimeout (void)**

Gets the socket timeout value used for DNS server communications.

Gets the timeout value applied to all sockets that communicate with the DNS server. Call this function to verify the timeout used by DNS socket operations.

**Returns:**

Global timeout value for sockets use in DNS server communications

**See also:**

[dns\\_settimeout](#)

#### **8.23.3.6 void dns\_init (void)**

Initializes the DNS client code.

Performs initialization for the DNS client. This function need only be called once at the start of the application.

#### **8.23.3.7 void dns\_setmaxtimeout (unsigned long *t*)**

Sets the maximum socket timeout value used for DNS server communications.

Sets the maximum timeout value that can be applied sockets that communicate with the DNS server. DNS operations are retried up to 4 times, and each time the timeout is doubled. This function sets the maximum timeout allowed for a single operation.

**Parameters:**

*t* Global maximum timeout value for sockets use in DNS server communications

**See also:**

[dns\\_gettimeout](#)

[dns\\_settimeout](#)

#### **8.23.3.8 void dns\_setprimary (struct [sockaddr](#) \* *sa*)**

Sets the address of the primary DNS server.

Sets the address of the primary DNS server used by this DNS client code. DNS operations first try to use a server designated as primary, and the use a server designated as secondary if the primary fails to return results.

Note that this sets the system's primary DNS server setting. If the system's primary DNS server entry had been previously set by the DHCP client, that information will be destroyed by this function.

**Parameters:**

*sa* address of primary DNS server

**See also:**

[dns\\_getprimary](#)

[dns\\_setsecondary](#)

[dns\\_getsecondary](#)

#### 8.23.3.9 void dns\_setsecondary (struct [sockaddr](#) \* *sa*)

Sets the address of the secondary DNS server.

Sets the address of the secondary DNS server used by this DNS client code. DNS operations first try to use a server designated as primary, and then use a server designated as secondary if the primary fails to return results.

Note that this sets the system's secondary DNS server setting. If the system's secondary DNS server entry had been previously set by the DHCP client, that information will be destroyed by this function.

##### Parameters:

*sa* address of secondary DNS server

##### See also:

[dns\\_getprimary](#)  
[dns\\_setprimary](#)  
[dns\\_getsecondary](#)

#### 8.23.3.10 void dns\_settimeout (unsigned long *t*)

Sets the socket timeout value used for DNS server communications.

Sets the timeout value applied to all sockets that communicate with the DNS server. Call this function to make sure DNS operations fail after a reasonable waiting time. All DNS operations are retried up to 4 times. In each retry, the local timeout will be doubled, up to the maximum timeout allowed.

##### Parameters:

*t* Global timeout value for sockets use in DNS server communications

##### See also:

[dns\\_gettimeout](#)  
[dns\\_setmaxtimeout](#)

#### 8.23.3.11 unsigned int dns\_version ()

Returns the version number of this DNS client library.

##### Returns:

Version number of this DNS client library.

### 8.23.3.12 struct [hostent](#)\* gethostbyaddr (void \* *addr*, [size\\_t](#) *len*, int *type*)

Looks up information on a host given an IP address.

Contacts a DNS server and attempts to find known host names for the given IP address.

#### Parameters:

*addr* IP address structure, either [in\\_addr](#) or [in6\\_addr](#)

*len* The length of the input structure passed to *addr* (4 or 16)

*type* [AF\\_INET](#) or [AF\\_INET6](#)

#### Returns:

Host structure with any names found, or *NULL* if the operation failed.

#### See also:

[AF\\_INET](#)

[AF\\_INET6](#)

[in\\_addr](#)

[in6\\_addr](#)

[gethostbyname](#)

[inet\\_addr](#)

[hostent](#)

### 8.23.3.13 struct [hostent](#)\* gethostbyname (char \* *name*)

Looks up information on a host given a host name.

Contacts a DNS server and attempts to find known IP addresses given a host name.

#### Parameters:

*name* String representing the host name

#### Returns:

Host structure with any names found, or *NULL* if the operation failed.

#### See also:

[gethostbyaddr](#)

[hostent](#)

## 8.24 tini400\_ftpclient.h File Reference

### 8.24.1 Detailed Description

FTP Client functions for DS80C400.

This library contains functions for FTP Client.

**Warning:**

The functions in this library are **NOT** multi-process safe—that is, if you call the same method from two different processes at the same time, the parameters to the function may be destroyed, yielding unpredictable results.

For detailed information on the DS80C400 please see the [High-Speed Microcontroller User's Guide: DS80C400 Supplement](#).

```
#include "rom400_sock.h"
#include "stdio.h"
#include "ftpcodes.h"
#include <string.h>
#include <ctype.h>
```

**Defines**

- #define [FTPCLIENT\\_VERSION\\_NUMBER](#) 1
- #define [FTPCLIENT\\_ASCII](#) 0
- #define [FTPCLIENT\\_BINARY](#) 1
- #define [FTPCLIENT\\_PORTNUMBER](#) 21
- #define [FTPCLIENT\\_ACTIVE\\_MODE](#) 1
- #define [FTPCLIENT\\_PASSIVE\\_MODE](#) 0
- #define [FTPCLIENT\\_DETAILED\\_DIRLISTING](#) 1
- #define [FTPCLIENT\\_SHORT\\_DIRLISTING](#) 0
- #define [FTPCLIENT\\_STATUS\\_SUCCESS](#) 0
- #define [FTPCLIENT\\_SOCKET\\_ERROR](#) -1
- #define [FTPCLIENT\\_FILE\\_NOT\\_FOUND](#) -2
- #define [FTPCLIENT\\_FILE\\_IO\\_ERROR](#) -3
- #define [FTPCLIENT\\_ALREADY\\_LOGGEDIN](#) -4
- #define [FTPCLIENT\\_NOT\\_CONNECTED](#) -5

**Functions**

- unsigned int [ftpclient\\_version](#) (void)  
*Returns version number of ftpclient library.*
- void [ftpclient\\_init](#) (long milli\_seconds)  
*Initializes the ftpclient library.*
- int [ftpclient\\_connect](#) (struct [sockaddr\\_in](#) \*sa, char \*user, char \*passwd)  
*Connects with FTP server.*



- int [ftpclient\\_settransmissionmode](#) (char flag)  
*Sets data transfer mode in FTP server.*
- void [ftpclient\\_setdataconnectionmode](#) (char flag)  
*Set data connection mode in ftpclient library.*
- int [ftpclient\\_getfile](#) (char \*filename, char \*storeas\_filename)  
*Downloads file from FTP server.*
- int [ftpclient\\_putfile](#) (char \*filename, char \*storeas\_filename)  
*Uploads file to FTP server.*
- int [ftpclient\\_dir](#) (char \*name, char \*dir\_str, int dir\_str\_len, char format)  
*Returns FTP server directory list.*
- int [ftpclient\\_pwd](#) (char \*path\_str, int path\_str\_len)  
*Returns current FTP server directory path.*
- int [ftpclient\\_cd](#) (char \*path\_str)  
*Changes server working directory.*
- int [ftpclient\\_rawcmd](#) (char \*input\_cmd)  
*Sends command to FTP server.*
- int [ftpclient\\_dataconnection](#) ()  
*Configures for new data connection. exchange port number and ip address information with FTP server for data connection.*
- int [ftpclient\\_get\\_dataconnection\\_handler](#) ()  
*Establishes new data connection and returns socket handler.*
- int [ftpclient\\_disconnect](#) (void)  
*Terminates connection with FTP server.*
- char \* [ftpclient\\_getlaststatus](#) (void)  
*Returns last FTP server response string.*

## 8.24.2 Define Documentation

### 8.24.2.1 #define FTPCLIENT\_ACTIVE\_MODE 1

Definition for active data connection mode

See also:

[ftpclient\\_setdataconnectionmode](#)

#### **8.24.2.2 #define FTPCLIENT\_ALREADY\_LOGGEDIN -4**

Error value indicates that client application is already logged-in

#### **8.24.2.3 #define FTPCLIENT\_ASCII 0**

Definition for ASCII data transfer mode

See also:

[ftpclient\\_settransmissionmode](#)

#### **8.24.2.4 #define FTPCLIENT\_BINARY 1**

Definition for BINARY data transfer mode

See also:

[ftpclient\\_settransmissionmode](#)

#### **8.24.2.5 #define FTPCLIENT\_DETAILED\_DIRLISTING 1**

Definition for detailed directory listing

See also:

[ftpclient\\_dir](#)

#### **8.24.2.6 #define FTPCLIENT\_FILE\_IO\_ERROR -3**

File operation error value

#### **8.24.2.7 #define FTPCLIENT\_FILE\_NOT\_FOUND -2**

File not found error value

#### **8.24.2.8 #define FTPCLIENT\_NOT\_CONNECTED -5**

Error value indicates that server is not connected

#### **8.24.2.9 #define FTPCLIENT\_PASSIVE\_MODE 0**

Definition for passive data connection mode

**See also:**

[ftpclient\\_setdataconnectionmode](#)

#### **8.24.2.10 #define FTPCLIENT\_PORTNUMBER 21**

Definition for default FTP server port number

**See also:**

[ftpclient\\_connect](#)

#### **8.24.2.11 #define FTPCLIENT\_SHORT\_DIRLISTING 0**

Definition for short directory listing

**See also:**

[ftpclient\\_dir](#)

#### **8.24.2.12 #define FTPCLIENT\_SOCKET\_ERROR -1**

Socket error value

#### **8.24.2.13 #define FTPCLIENT\_STATUS\_SUCCESS 0**

FTP Client Status Success value, this value is returned when operation is completed successfully.

#### **8.24.2.14 #define FTPCLIENT\_VERSION\_NUMBER 1**

Version number associated with this header file. Should be the same as the version number returned by the [ftpclient\\_version](#) function.

**See also:**

[ftpclient\\_version](#)

### **8.24.3 Function Documentation**

#### **8.24.3.1 int ftpclient\_cd (char \* *path\_str*)**

Changes server working directory.

This function changes server working directory

**Parameters:**

*path\_str* Address of memory buffer that contains new working directory path name

**Returns:**

- *FTPCLIENT\_NOT\_CONNECTED* - if connection is not established
- *FTPCLIENT\_SOCKET\_ERROR* - if socket communication error happens

Otherwise, returns FTP server status code

**8.24.3.2 int ftpclient\_connect (struct sockaddr\_in \* *sa*, char \* *user*, char \* *passwd*)**

Connects with FTP server.

This function establishes connection with FTP server. Connection with FTP server must be established before calling any other functions that interact with FTP server.

**Parameters:**

*sa* socket address contains server ip address and FTP server portnumber **NOTE:** Passing zero value for portnumber enables ftpclient library to use default ftp port number

*user* User name

*passwd* Password

**Returns:**

One of the following values:

- *FTPCLIENT\_ALREADY\_LOGGEDIN* - if ftpclient is already connected with server
- *FTPCLIENT\_SOCKET\_ERROR* - if there is any error in socket communication

Otherwise, FTP server status code will be returned for successful or failed authentication

**NOTE:** In case of error, the server socket will be closed before returning from function

**8.24.3.3 int ftpclient\_dataconnection ()**

Configures for new data connection. exchange port number and ip address information with FTP server for data connection.

This function configures for new data connection. For Active mode connection, sends IP address and port number of ftp client to which the data connection have to be established. For passive mode connection, it gets server IP address and port number for data connection

**Returns:**

[\*FTPCLIENT\\_SOCKET\\_ERROR\*](#) if socket communication error happens. Otherwise, returns FTP server status code

**8.24.3.4 int ftpclient\_dir (char \* name, char \* dir\_str, int dir\_str\_len, char format)**

Returns FTP server directory list.

This function returns FTP server directory list in short format or detailed format. This function can also be used to retrieve information about specific file.

**Parameters:**

**name** Name of the file to get file attributes information. If NULL, then information about all entries of current directory will be returned.

**dir\_str** Address of memory buffer where directory information will be stored

**dir\_str\_len** Maximum amount of data to be stored in *dir\_str* memory buffer

**format** Specifies the format of directory listing. The value for this parameter should be either [\*FTPCLIENT\\_DETAILED\\_DIRLISTING\*](#) or [\*FTPCLIENT\\_SHORT\\_DIRLISTING\*](#)

**Returns:**

- [\*FTPCLIENT\\_NOT\\_CONNECTED\*](#) - if connection is not established
- [\*FTPCLIENT\\_SOCKET\\_ERROR\*](#) - if socket communication error happens

Otherwise, returns FTP server status code

**8.24.3.5 int ftpclient\_disconnect (void)**

Terminates connection with FTP server.

This function terminates connection with FTP server. the server socket will be closed even if there is any socket error

**Returns:**

- [\*FTPCLIENT\\_SOCKET\\_ERROR\*](#) - if socket communication error happens
- [\*FTPCLIENT\\_NOT\\_CONNECTED\*](#) - if connection is not established

Otherwise, returns FTP server status code

**8.24.3.6 int ftpclient\_get\_dataconnection\_handler ()**

Establishes new data connection and returns socket handler.

This function establishes new data connection and returns socket handler.

**IMPORTANT NOTE:** For Active mode connection, This function has to be called after sending control command to server to initiate the data transfer as server will establish data connection after receiving control command. For passive mode connection, this function has to be called before sending control command to server to initiate the data transfer as server expects data connection to be made before responding for control connection.

**Returns:**

*FTPCLIENT\_SOCKET\_ERROR* if socket communication error happens. Otherwise, returns FTP server status code

#### **8.24.3.7 int ftpclient\_getfile (char \* *filename*, char \* *storeas\_filename*)**

Downloads file from FTP server.

This function downloads file from FTP server and store it in tini file system.

**Parameters:**

*filename* Name of file to get from the FTP server

*storeas\_filename* Name of file to store on TINI. If value for this parameter is NULL, then the file will be stored under same name as it is on the FTP server.

**Returns:**

- *FTPCLIENT\_NOT\_CONNECTED* - if connection is not established
- *FTPCLIENT\_FILE\_IO\_ERROR* - if error happens while storing file
- *FTPCLIENT\_SOCKET\_ERROR* - if socket communication error happens

Otherwise, returns FTP server status code

#### **8.24.3.8 char\* ftpclient\_getlaststatus (void)**

Returns last FTP server response string.

This function returns the FTP server's response status string for the last control command sent to the server.

**Returns:**

Pointer to response status string

#### **8.24.3.9 void ftpclient\_init (long *milli\_seconds*)**

Initializes the ftpclient library.

This function initializes ftpclient library internal datastructure and configures the library with following default configuration

- ASCII file transfer mode
- Active data connection mode

**Parameters:**

*milli\_seconds* socket timeout value

### 8.24.3.10 int ftpclient\_putfile (char \* *filename*, char \* *storeas\_filename*)

Uploads tini file to FTP server.

This function uploads tini file to FTP server.

**Parameters:**

*filename* Name of file on the TINI to send to the server

*storeas\_filename* Name to give the file put on the FTP server. If NULL, then the name for the file on TINI will be used.

**Returns:**

- *FTPCLIENT\_NOT\_CONNECTED* - if connection is not established
- *FTPCLIENT\_FILE\_NOT\_FOUND* - if the input tini file name is not there in tini file system
- *FTPCLIENT\_SOCKET\_ERROR* - if socket communication error happens

Otherwise, returns FTP server status code

### 8.24.3.11 int ftpclient\_pwd (char \* *path\_str*, int *path\_str\_len*)

Returns current FTP server directory path.

This function returns the current FTP server directory path name

**Parameters:**

*path\_str* Address of memory buffer where the current FTP server path name will be stored

*path\_str\_len* Maximum amount of data can be stored in path\_str memory buffer

**Returns:**

- *FTPCLIENT\_NOT\_CONNECTED* - if connection is not established
- *FTPCLIENT\_SOCKET\_ERROR* - if socket communication error happens

Otherwise, returns FTP server status code

#### 8.24.3.12 `int ftpclient_rawcmd (char * input_cmd)`

Sends command to FTP server.

This function sends command to FTP server through control connection and returns FTP server status code. This function does **NOT** check whether server is connected.

**NOTE:** To retrieve the response string of server for control command, call the [\*ftpclient\\_getlaststatus\*](#) function

**Parameters:**

*input\_cmd* command to send to the FTP server

**Returns:**

[\*FTPCLIENT\\_SOCKET\\_ERROR\*](#) if socket communication error happens. Otherwise, returns FTP server status code

#### 8.24.3.13 `void ftpclient_setdataconnectionmode (char flag)`

Set data connection mode in ftpclient library.

This function sets data connection mode in ftpclient library. All future data connections will be made in the mode set by this function

**Parameters:**

*flag* should be either [\*FTPCLIENT\\_ACTIVE\\_MODE\*](#) or [\*FTPCLIENT\\_PASSIVE\\_MODE\*](#)

**Warning:**

Invalid value for "flag" yields unexpected behavior of ftpclient data transfer functions

#### 8.24.3.14 `int ftpclient_settransmissionmode (char flag)`

Sets data transfer mode in FTP server.

This function sets data transfer mode in FTP server and ftpclient library

**Parameters:**

*flag* should be either [\*FTPCLIENT\\_ASCII\*](#) or [\*FTPCLIENT\\_BINARY\*](#)

**NOTE:** Invalid input for *flag* will be interpreted as [\*FTPCLIENT\\_BINARY\*](#).

**Returns:**

returns FTP server status code



### 8.24.3.15 unsigned int ftpclient\_version (void)

Returns version number of ftpclient library.

#### Returns:

Version number of ftpclient library

## 8.25 tini400\_isr.h File Reference

### 8.25.1 Detailed Description

Interrupt Service Routine installation functions.

This library contains functions that allow processes to install their own ISR's from C programs. Normally, the Keil compiler would automatically install interrupts in their proper locations. However, the act of initializing the ROM sets the entire interrupt vector table, so any interrupt vector that the Keil compiler generates are destroyed. These functions allow programs to restore or update their interrupt vector tables.

To use interrupts written in C with the Keil compiler, functions should be defined with the **interrupt** keyword. Also, under the Project Target options dialog, under the C51 panel, uncheck the box labeled *Interrupt Vectors at Address:*. Then make sure to call [isr\\_setinterruptvector](#) sometime after [init\\_rom](#) has been called.

For detailed information on the DS80C400 please see the [High-Speed Microcontroller User's Guide: DS80C400 Supplement](#).

The functions in this library are multi-process safe—that is, if you call the same method from two different processes at the same time, the parameters to the function will not be destroyed.

#### Defines

- #define [TINI400\\_ISR\\_VERSION](#) 4
- #define [ISR\\_EXTERNALINT0](#) 0
- #define [ISR\\_TIMER0](#) 1
- #define [ISR\\_EXTERNALINT1](#) 2
- #define [ISR\\_TIMER1](#) 3
- #define [ISR\\_SERIAL0](#) 4
- #define [ISR\\_TIMER2](#) 5
- #define [ISR\\_POWERFAIL](#) 6
- #define [ISR\\_SERIAL1](#) 7
- #define [ISR\\_EXTERNALINT2345](#) 8
- #define [ISR\\_EXTERNALINT2](#) 8
- #define [ISR\\_TIMER3](#) 9
- #define [ISR\\_EXTERNALINT3](#) 9

- #define [ISR\\_SERIAL2](#) 10
- #define [ISR\\_EXTERNALINT4](#) 10
- #define [ISR\\_WRITEPROTECT](#) 11
- #define [ISR\\_EXTERNALINT5](#) 11
- #define [ISR\\_WATCHDOG](#) 12
- #define [ISR\\_CAN0](#) 13
- #define [ISR\\_ETHERNET](#) 14
- #define [ISR\\_CAN1](#) 14
- #define [ISR\\_ETHERNETPOWER](#) 15
- #define [ISR\\_CANBUSACTIVITY](#) 15

## Functions

- void [isr\\_setinterruptvector](#) (int vector\_number, void \*function\_ptr)  
*Installs an interrupt vector.*
- void \* [isr\\_getinterruptvector](#) (int vector\_number)  
*Gets the current value of an interrupt vector.*
- unsigned int [isr\\_version](#) (void)  
*Returns the version number of this ISR library.*

## 8.25.2 Define Documentation

### 8.25.2.1 #define [ISR\\_CAN0](#) 13

Interrupt vector number for the **CAN 0** interrupt.

See also:

[isr\\_setinterruptvector](#)  
[isr\\_getinterruptvector](#)

### 8.25.2.2 #define [ISR\\_CAN1](#) 14

Interrupt vector number for the **CAN 1** interrupt. **Applicable to DS80C390 only.**

See also:

[isr\\_setinterruptvector](#)  
[isr\\_getinterruptvector](#)

#### 8.25.2.3 **#define ISR\_CANBUSACTIVITY 15**

Interrupt vector number for the **CAN 0 & 1 bus activity** interrupt. **Applicable to DS80C390 only.**

See also:

[isr\\_setinterruptvector](#)  
[isr\\_getinterruptvector](#)

#### 8.25.2.4 **#define ISR\_ETHERNET 14**

Interrupt vector number for the **Ethernet Activity** interrupt. **Applicable to DS80C400 only.**

See also:

[isr\\_setinterruptvector](#)  
[isr\\_getinterruptvector](#)

#### 8.25.2.5 **#define ISR\_ETHERNETPOWER 15**

Interrupt vector number for the **External Power Mode** interrupt. **Applicable to DS80C400 only.**

See also:

[isr\\_setinterruptvector](#)  
[isr\\_getinterruptvector](#)

#### 8.25.2.6 **#define ISR\_EXTERNALINT0 0**

Interrupt vector number for the **External Interrupt 0** interrupt.

See also:

[isr\\_setinterruptvector](#)  
[isr\\_getinterruptvector](#)

#### 8.25.2.7 **#define ISR\_EXTERNALINT1 2**

Interrupt vector number for the **External Interrupt 1** interrupt.

See also:

[isr\\_setinterruptvector](#)  
[isr\\_getinterruptvector](#)

#### 8.25.2.8 **#define ISR\_EXTERNALINT2 8**

Interrupt vector number for the **External Interrupt 2** interrupt. **Applicable to DS80C390 only.**

See also:

[isr\\_setinterruptvector](#)

[isr\\_getinterruptvector](#)

#### 8.25.2.9 **#define ISR\_EXTERNALINT2345 8**

Interrupt vector number for the **External Interrupt 2/3/4/5** interrupt. **Applicable to DS80C400 only.**

See also:

[isr\\_setinterruptvector](#)

[isr\\_getinterruptvector](#)

#### 8.25.2.10 **#define ISR\_EXTERNALINT3 9**

Interrupt vector number for the **External Interrupt 3** interrupt. **Applicable to DS80C390 only.**

See also:

[isr\\_setinterruptvector](#)

[isr\\_getinterruptvector](#)

#### 8.25.2.11 **#define ISR\_EXTERNALINT4 10**

Interrupt vector number for the **External Interrupt 4** interrupt. **Applicable to DS80C390 only.**

See also:

[isr\\_setinterruptvector](#)

[isr\\_getinterruptvector](#)

#### 8.25.2.12 **#define ISR\_EXTERNALINT5 11**

Interrupt vector number for the **External Interrupt 5** interrupt. **Applicable to DS80C390 only.**

See also:

[isr\\_setinterruptvector](#)

[isr\\_getinterruptvector](#)

#### 8.25.2.13 **#define ISR\_POWERFAIL 6**

Interrupt vector number for the **Power Fail** interrupt.

See also:

[isr\\_setinterruptvector](#)  
[isr\\_getinterruptvector](#)

#### 8.25.2.14 **#define ISR\_SERIAL0 4**

Interrupt vector number for the **Serial Port 0** interrupt.

See also:

[isr\\_setinterruptvector](#)  
[isr\\_getinterruptvector](#)

#### 8.25.2.15 **#define ISR\_SERIAL1 7**

Interrupt vector number for the **Serial Port 1** interrupt.

See also:

[isr\\_setinterruptvector](#)  
[isr\\_getinterruptvector](#)

#### 8.25.2.16 **#define ISR\_SERIAL2 10**

Interrupt vector number for the **Serial Port 2** interrupt. **Applicable to DS80C400 only.**

See also:

[isr\\_setinterruptvector](#)  
[isr\\_getinterruptvector](#)

#### 8.25.2.17 **#define ISR\_TIMER0 1**

Interrupt vector number for the **Timer 0** interrupt.

See also:

[isr\\_setinterruptvector](#)  
[isr\\_getinterruptvector](#)

#### 8.25.2.18 **#define ISR\_TIMER1 3**

Interrupt vector number for the **Timer 1** interrupt.

See also:

[isr\\_setinterruptvector](#)  
[isr\\_getinterruptvector](#)

#### 8.25.2.19 **#define ISR\_TIMER2 5**

Interrupt vector number for the **Timer 2** interrupt.

See also:

[isr\\_setinterruptvector](#)  
[isr\\_getinterruptvector](#)

#### 8.25.2.20 **#define ISR\_TIMER3 9**

Interrupt vector number for the **Timer 3** interrupt. **Applicable to DS80C400 only.**

See also:

[isr\\_setinterruptvector](#)  
[isr\\_getinterruptvector](#)

#### 8.25.2.21 **#define ISR\_WATCHDOG 12**

Interrupt vector number for the **Watchdog Timer** interrupt.

See also:

[isr\\_setinterruptvector](#)  
[isr\\_getinterruptvector](#)

#### 8.25.2.22 **#define ISR\_WRITEPROTECT 11**

Interrupt vector number for the **Write Protect** interrupt. **Applicable to DS80C400 only.**

See also:

[isr\\_setinterruptvector](#)  
[isr\\_getinterruptvector](#)

#### 8.25.2.23 #define TINI400\_ISR\_VERSION 4

Version number associated with this header file. Should be the same as the version number returned by the [isr\\_version](#) function.

See also:

[isr\\_version](#)

### 8.25.3 Function Documentation

#### 8.25.3.1 void\* isr\_getinterruptvector (int *vector\_number*)

Gets the current value of an interrupt vector.

Returns a function pointer to the interrupt service routine for the interrupt defined by *vector\_number*. Note that *vector\_number* is **NOT** the address of the interrupt, but the number corresponding to that interrupt as described in the Keil documentation. For example, a *vector\_number* of **1** corresponds to the interrupt at address 0Bh, which is the timer 0 overflow interrupt. A *vector\_number* of **4** corresponds to the interrupt at address 23h, which is the serial port 0 interrupt.

This file contains several defines for common interrupts that can be used for the *vector\_number* parameter.

##### Parameters:

***vector\_number*** ID of the interrupt to be installed. It is up to the user to make sure this parameter is in range

##### Returns:

function pointer for the interrupt service routine. Returns *NULL* if the instruction at the interrupt's address is not an LJMP.

See also:

[isr\\_setinterruptvector](#)

#### 8.25.3.2 void isr\_setinterruptvector (int *vector\_number*, void \**function\_ptr*)

Installs an interrupt vector.

Installs the function *function\_ptr* as the interrupt service routine for the interrupt defined by *vector\_number*. Note that *vector\_number* is **NOT** the address of the interrupt, but the number corresponding to that interrupt as described in the Keil documentation. For example, a *vector\_number* of **1** corresponds to the interrupt at address 0Bh, which is the timer 0 overflow interrupt. A *vector\_number* of **4** corresponds to the interrupt at address 23h, which is the serial port 0 interrupt.

This file contains several defines for common interrupts that can be used for the *vector\_number* parameter.

The function *function\_ptr* should terminate with a *reti* statement (functions declared with the *interrupt* keyword in Keil automatically have this).

**Parameters:**

*vector\_number* ID of the interrupt to be installed. It is up to the user to make sure this parameter is in range

*function\_ptr* function that will be the interrupt service routine

**See also:**

[isr\\_getinterruptvector](#)

### 8.25.3.3 unsigned int isr\_version (void)

Returns the version number of this ISR library.

**Returns:**

Version number of this ISR library.

## 8.26 tini400\_mime.h File Reference

### 8.26.1 Detailed Description

MIME Library functions for DS80C400 processor.

This library contains functions for encoding and decoding mime messages

For detailed information on the DS80C400 please see the [High-Speed Microcontroller User's Guide: DS80C400 Supplement](#).

**Warning:**

The functions in this library are **NOT** multi-process safe—that is, if you call the same method from two different processes at the same time, the parameters to the function may be destroyed, yielding unpredictable results.

**Defines**

- #define [BASE64](#) 1
- #define [QUOTED\\_PRINTABLE](#) 2
- #define [MIME\\_VERSION](#) 1



## Functions

- void [mime\\_init](#) (void)  
*Initializes mime library.*
- char \* [mime\\_encode](#) (unsigned char far \*inbuf, unsigned int size, char encode\_flag)  
*Encodes the given message to mime format.*
- char \* [mime\\_decode](#) (char far \*inbuf, char decode\_flag)  
*Decodes the given mime message.*

### 8.26.2 Define Documentation

#### 8.26.2.1 #define BASE64 1

Definition for mime base64 encoding and decoding method

See also:

[mime\\_encode](#), [mime\\_decode](#)

#### 8.26.2.2 #define MIME\_VERSION 1

Version number associated with this header file. Should be the same as the version number returned by the [mime\\_version](#) function.

See also:

[mime\\_version](#)

#### 8.26.2.3 #define QUOTED\_PRINTABLE 2

Definition for mime quoted printable encoding and decoding method

See also:

[mime\\_encode](#), [mime\\_decode](#)

### 8.26.3 Function Documentation

#### 8.26.3.1 char\* mime\_decode (char far \* inbuf, char decode\_flag)

Decodes the given mime message.

See RFC1521 for more information on MIME

**Parameters:**

*inbuf* - mime message to decode

*decode\_flag* - decoding flag indicates what decoding method to be used, should be either [BASE64](#) or [QUOTED\\_PRINTABLE](#)

**Returns:**

address of decoded message buffer or NULL if function failed

### 8.26.3.2 char\* mime\_encode (unsigned char far \* *inbuf*, unsigned int *size*, char *encode\_flag*)

Encodes the given message to mime format.

See RFC1521 for more information on MIME

**Parameters:**

*inbuf* input buffer to encode

*size* length of the input buffer

*encode\_flag* not used, reserved for future use

**Returns:**

address of encoded mime message buffer or NULL if function failed

## 8.27 tini400\_ntlm.h File Reference

### 8.27.1 Detailed Description

NTLM Library functions for DS80C400 processor.

This library contains functions for managing NeTwork Lan Manager(NTLM) authentication protocol

For detailed information on the DS80C400 please see the [High-Speed Microcontroller User's Guide: DS80C400 Supplement](#).

**Warning:**

The functions in this library are **NOT** multi-process safe—that is, if you call the same method from two different processes at the same time, the parameters to the function may be destroyed, yielding unpredictable results.

**Data Structures**

- [struct \\_sbufhdr](#)
- [struct \\_type1msghdr](#)

- struct [\\_type1msg](#)
- struct [\\_type2msgghdr](#)
- struct [\\_type2msg](#)
- struct [\\_type3msgghdr](#)
- struct [\\_type3msg](#)

## Defines

- #define [MAX\\_NTLM\\_BUF](#) 1024
- #define [NTLM\\_SIGN](#) "NTLMSSP\0"
- #define [NTLM\\_TYPE1\\_MSG](#) 1
- #define [NTLM\\_TYPE3\\_MSG](#) 3
- #define [NTLM\\_FLAGS](#) 0x0000b207L

## Typedefs

- typedef [\\_sbufhdr](#) sbufhdr
- typedef [\\_type1msgghdr](#) type1msgghdr
- typedef [\\_type1msg](#) type1msg
- typedef [\\_type2msgghdr](#) type2msgghdr
- typedef [\\_type2msg](#) type2msg
- typedef [\\_type3msgghdr](#) type3msgghdr
- typedef [\\_type3msg](#) type3msg

## Functions

- void [generate\\_type1\\_msg](#) (type1msg \*t1\_msg, char \*user)  
*Generates type1 NTLM message.*
- void [generate\\_type3\\_msg](#) (type2msg \*t2\_msg, type3msg \*t3\_msg, char \*user, char \*pass)  
*Generates type3 NTLM message.*

### 8.27.2 Define Documentation

#### 8.27.2.1 #define MAX\_NTLM\_BUF 1024

definition for maximum ntlm security buffer length.

See also:

[generate\\_type1\\_msg](#), [generate\\_type3\\_msg](#)

#### **8.27.2.2 #define NTLM\_FLAGS 0x0000b207L**

definition for NTLM flags

See also:

[generate\\_type1\\_msg](#), [generate\\_type3\\_msg](#)

#### **8.27.2.3 #define NTLM\_SIGN "NTLMSSP\0"**

definition for NTLM signature

See also:

[generate\\_type1\\_msg](#), [generate\\_type3\\_msg](#)

#### **8.27.2.4 #define NTLM\_TYPE1\_MSG 1**

definition for type 1 NTLM Message

See also:

[generate\\_type1\\_msg](#)

#### **8.27.2.5 #define NTLM\_TYPE3\_MSG 3**

definition for type 3 NTLM Message

See also:

[generate\\_type3\\_msg](#)

### **8.27.3 Typedef Documentation**

#### **8.27.3.1 typedef struct [\\_sbufhdr](#) sbufhdr**

Structure for security buffer header

#### **8.27.3.2 typedef struct [\\_type1msg](#) type1msg**

Structure for type1 message

#### **8.27.3.3 typedef struct [\\_type1msghdr](#) type1msghdr**

Structure for type1 message header

#### **8.27.3.4 typedef struct [\\_type2msg](#) type2msg**

Structure for type2 message

#### 8.27.3.5 typedef struct [\\_type2msghdr](#) type2msghdr

Structure for type2 message header

#### 8.27.3.6 typedef struct [\\_type3msg](#) type3msg

Structure for type3 message

#### 8.27.3.7 typedef struct [\\_type3msghdr](#) type3msghdr

Structure for type3 message header

### 8.27.4 Function Documentation

#### 8.27.4.1 void generate\_type1\_msg ([type1msg](#) \* *t1\_msg*, char \* *user*)

Generates type1 NTLM message.

This function generates Type1 NTLM message that is sent to server to get type2 message. For more information, See NTLM authentication protocol specification.

##### Parameters:

*t1\_msg* the NTLM type 1 message

*user* the user name

##### See also:

[generate\\_type3\\_msg](#)

#### 8.27.4.2 void generate\_type3\_msg ([type2msg](#) \* *t2\_msg*, [type3msg](#) \* *t3\_msg*, char \* *user*, char \* *pass*)

Generates type3 NTLM message.

This function generates Type3 NTLM message that contains both LAN Manager and NT LAN manager responses for server challenge. For more information, See NTLM authentication protocol specification.

##### Parameters:

*t2\_msg* the type 2 NTLM message

*t3\_msg* the type 3 NTLM message

*user* user name

*pass* password

##### See also:

[generate\\_type1\\_msg](#)

## 8.28 tini400\_pop3.h File Reference

### 8.28.1 Detailed Description

Pop3 Library functions for DS80C400 processor.

This library contains functions for receiving mails from pop3 mail server

For detailed information on the DS80C400 please see the [High-Speed Microcontroller User's Guide: DS80C400 Supplement](#).

#### Warning:

The functions in this library are **NOT** multi-process safe—that is, if you call the same method from two different processes at the same time, the parameters to the function may be destroyed, yielding unpredictable results.

#### Data Structures

- struct [\\_mailheader](#)
- struct [\\_userheader](#)
- struct [\\_mail](#)
- struct [\\_maillist](#)
- struct [\\_pop3\\_session](#)

#### Defines

- #define [MAX\\_LINE\\_SIZE](#) 1024
- #define [POP3\\_VERSION](#) 2
- #define [POP3\\_MAXATTACHMENTSIZ](#) 5
- #define [POP3\\_MAXUSERHEADERSIZ](#) 20
- #define [POP3\\_INSUFFICIENT\\_MEMORY](#) -1
- #define [POP3\\_RECEIVEMAIL\\_ERROR](#) -5
- #define [POP3\\_INVALID\\_MAILNUMBER](#) -6
- #define [POP3\\_SOCKET\\_ERROR](#) -7
- #define [POP3\\_INVALID\\_USER\\_PASSWORD](#) -10
- #define [POP3\\_LIBRARY\\_IS\\_NOT\\_CONFIGURED](#) -11
- #define [POP3\\_NOT\\_CONNECTED](#) -12
- #define [POP3\\_FILE\\_ERROR](#) -13
- #define [POP3\\_STATUS\\_SUCCESS](#) 0

## Typedefs

- typedef [\\_mailheader](#) mailheader
- typedef [\\_userheader](#) userheader
- typedef [\\_mail](#) mail
- typedef [\\_maillist](#) maillist
- typedef [\\_pop3\\_session](#) pop3\_session

## Functions

- void [pop3\\_init](#) (void)  
*Initializes pop3 library.*
- void [pop3\\_setuserheaderlist](#) (struct [\\_userheader](#) \*pusrhdr)  
*Sets user defined mail header list.*
- int [pop3\\_login](#) (long pop3\_host, char \*username, char \*pwd)  
*Login to pop3 server.*
- int [pop3\\_deletemail](#) (int mailnumber)  
*Deletes mail from pop3 mailbox.*
- int [pop3\\_getmailboxstate](#) (int \*numberofmails, long \*total\_size)  
*Gets number of mails and mailbox size value from pop3 server.*
- [\\_mail](#) \* [pop3\\_receivemail](#) (int mailnumber, int \*status)  
*Receives mail from pop3 mail server.*
- [\\_maillist](#) \* [pop3\\_getmaillist](#) (int \*status)  
*Reads mail list from pop3 server.*
- int [pop3\\_logout](#) (void)  
*Terminates connection from pop3 server.*
- void [pop3\\_registerauthcallback](#) (int(\*funpt)())  
*Registers pop3 authentication callback routine.*

### 8.28.2 Define Documentation

#### 8.28.2.1 #define MAX\_LINE\_SIZE 1024

Definition for maximum size of mail header

#### **8.28.2.2 #define POP3\_FILE\_ERROR -13**

File creation error value, This value is returned if there is any error in opening file

**See also:**

[pop3\\_receivemail](#)

#### **8.28.2.3 #define POP3\_INSUFFICIENT\_MEMORY -1**

Insufficient memory error value

**See also:**

[pop3\\_receivemail](#)

[pop3\\_login](#)

[pop3\\_getmaillist](#)

#### **8.28.2.4 #define POP3\_INVALID\_MAILNUMBER -6**

Invalid mail number error value

**See also:**

[pop3\\_receivemail](#)

#### **8.28.2.5 #define POP3\_INVALID\_USER\_PASSWORD -10**

Invalid User or Password error value

**See also:**

[pop3\\_login](#)

#### **8.28.2.6 #define POP3\_LIBRARY\_IS\_NOT\_CONFIGURED -11**

pop3 library is not configured error value, this value will be returned if pop3 host information is not configured

**See also:**

[pop3\\_login](#)

#### **8.28.2.7 #define POP3\_MAXATTACHMENTSIZ 5**

Definition for maximum number of attachments

**See also:**

[pop3\\_receivemail](#)



#### **8.28.2.8 #define POP3\_MAXUSERHEADERSIZE 20**

Definition for maximum number of user headers

**See also:**

[pop3\\_receivemail](#)

#### **8.28.2.9 #define POP3\_NOT\_CONNECTED -12**

This error value is returned if connection was not established with pop3 server.

**See also:**

[pop3\\_deletemail](#)  
[pop3\\_getmailboxstate](#)  
[pop3\\_getmaillist](#)  
[pop3\\_receivemail](#)  
[pop3\\_logout](#)

#### **8.28.2.10 #define POP3\_RECEIVEMAIL\_ERROR -5**

Receive mail error value

**See also:**

[pop3\\_receivemail](#)

#### **8.28.2.11 #define POP3\_SOCKET\_ERROR -7**

Socket error value

**See also:**

[pop3\\_receivemail](#)  
[pop3\\_login](#)  
[pop3\\_logout](#)  
[pop3\\_getmailboxstate](#)  
[pop3\\_getmaillist](#)

#### **8.28.2.12 #define POP3\_STATUS\_SUCCESS 0**

pop3 Status success value, this value is returned when operation is completed successfully.

**See also:**

[pop3\\_receivemail](#)

#### 8.28.2.13 #define POP3\_VERSION 2

Version number associated with this header file. Should be the same as the version number returned by the [pop3\\_version](#) function.

See also:

[pop3\\_version](#)

### 8.28.3 Typedef Documentation

#### 8.28.3.1 typedef struct [\\_mail mail](#)

Structure for mail that contains standard mail header, user mail header, message and attachment filename list

#### 8.28.3.2 typedef struct [\\_mailheader mailheader](#)

Structure for standard mail header holds standard mail header values

#### 8.28.3.3 typedef struct [\\_maillist maillist](#)

Structure for maillist

#### 8.28.3.4 typedef struct [\\_pop3\\_session pop3\\_session](#)

Structure for pop3\_session

#### 8.28.3.5 typedef struct [\\_userheader userheader](#)

Structure for user defined mail header contains user header name list and user header value list

### 8.28.4 Function Documentation

#### 8.28.4.1 int pop3\_deletemail (int *mailnumber*)

Deletes mail from pop3 mailbox.

This function sets delete mark against the input message number. The server will delete all the messages marked with delete mark after disconnecting from client. pop3 login operation must be performed before calling this function.

**Parameters:**

*mailnumber* Message number to set delete mark

**Returns:**

[POP3\\_STATUS\\_SUCCESS](#), if everything is successful. Otherwise, one of the following error values:

- [POP3\\_NOT\\_CONNECTED](#)
- [POP3\\_RECEIVEMAIL\\_ERROR](#)
- [POP3\\_INVALID\\_MAILNUMBER](#)

**See also:**

[pop3\\_getmaillist](#)  
[pop3\\_receivemail](#)

**8.28.4.2 int pop3\_getmailboxstate (int \* *numberofmails*, long \* *total\_size*)**

Gets number of mails and mailbox size value from pop3 server.

This function returns number of messages in mailbox and returns total size of the message. pop3 login operation must be performed before calling this function.

**Parameters:**

*numberofmails* points to address location where number of mails value will be stored

*total\_size* points to address location where total mailsize value will be stored

**Returns:**

[POP3\\_STATUS\\_SUCCESS](#), if everything is successful. Otherwise, one of the following error values:

- [POP3\\_NOT\\_CONNECTED](#)
- [POP3\\_RECEIVEMAIL\\_ERROR](#)

**See also:**

[pop3\\_getmaillist](#)  
[pop3\\_receivemail](#)

**8.28.4.3 struct [\\_maillist](#)\* pop3\_getmaillist (int \* *status*)**

Reads mail list from pop3 server.

This function returns list of mail numbers and size of each mail. pop3 login operation must be performed before calling this function.

**Parameters:**

*status* points to address location where status of pop3 function will be stored

**Returns:**

NULL if pop3\_getmaillist failed. Otherwise, returns pointer to maillist object

**See also:**

[pop3\\_login](#)  
[pop3\\_logout](#)  
[pop3\\_getmailboxstate](#)  
[pop3\\_deletemail](#)

**8.28.4.4 void pop3\_init (void)**

Initializes pop3 library.

This function initializes the internal data structures of pop3 library. This function should be called first before calling any other functions of pop3 library

**8.28.4.5 int pop3\_login (long pop3\_host, char \* username, char \* pwd)**

Login to pop3 server.

This function performs authentication with pop3 server and enters into transaction state. It does plain text authentication by default. user can override authentication method by registering their own authentication callback through pop3\_registerauthcallback function.

**Parameters:**

*pop3\_host* IP4 address structure, return value of [in\\_addr](#)

*username* user name value

*pwd* password value

**Returns:**

[POP3\\_STATUS\\_SUCCESS](#) if the operation is completed successfully. Otherwise, one of the following error values:

- [POP3\\_LIBRARY\\_IS\\_NOT\\_CONFIGURED](#)
- [POP3\\_INSUFFICIENT\\_MEMORY](#)
- [POP3\\_SOCKET\\_ERROR](#)
- [POP3\\_RECEIVEMAIL\\_ERROR](#)
- [POP3\\_INVALID\\_USER\\_PASSWORD](#)

**See also:**

[pop3\\_logout](#)

#### 8.28.4.6 `int pop3_logout (void)`

Terminates connection from pop3 server.

This function terminates connection with pop3 server. If pop3 login operation was not performed, it returns error.

**Returns:**

[POP3\\_STATUS\\_SUCCESS](#), if everything is successful. Otherwise, one of the following error values:

- [POP3\\_NOT\\_CONNECTED](#)
- [POP3\\_RECEIVEMAIL\\_ERROR](#)

**See also:**

[pop3\\_login](#)

#### 8.28.4.7 `struct _mail* pop3_receivemail (int mailnumber, int * status)`

Receives mail from pop3 mail server.

This function receives mail from POP3 server and returns pointer to mail object that contains standard mailheader value, user mail header value, message and attachment file list for received mail. this function supports both base64 and quoted-printable encryption/decryption types for both attachments and message. all the attachments will be directly stored in filesystem and mail object retains filenames of attachments.

**NOTE:** Memory for returned mailobject is allocated by this function. If user will not free the memory, then, the memory for mailobject will be re-allocated when pop3\_receivemail function is called next time

**NOTE:** User mail header name list should be set to retrieve user mail header values. Otherwise, this function will ignore user mail header values.

**Parameters:**

*mailnumber* Message number of mail to retrieve

*status* points to address location where status of pop3 function will be stored

**Returns:**

NULL if pop3\_receivemail function failed. Otherwise, returns pointer to mail object

**See also:**

[pop3\\_login](#)  
[pop3\\_logout](#)  
[pop3\\_getmaillist](#)  
[pop3\\_deletemail](#)

#### 8.28.4.8 void pop3\_registerauthcallback (int(\*)() *funpt*)

Registers pop3 authentication callback routine.

This function registers user defined authentication callback function with pop3 library. when pop3\_login function is called from application, user authentication routine will be called with pop3\_session object parameter that contains username,password and socket handler.

**NOTE:** User authentication callback function has to have the following function prototype to receive pop3\_session structure pointer.

int [authentication call back function name] (pop3\_session \*pop3\_handle)

**NOTE:** User authentication callback routine should return [POP3\\_STATUS\\_SUCCESS](#) value for successful authentication. for invalid user or password, [POP3\\_INVALID\\_USER\\_PASSWORD](#) error value should be returned.

**See also:**

[pop3\\_login](#)  
[pop3\\_session](#)

#### 8.28.4.9 void pop3\_setuserheaderlist (struct [\\_userheader](#) \* *pusrhdr*)

Sets user defined mail header list.

This function stores address of user mail header list structure in pop3 library global variable. the user mail header value will be retrieved for all user defined mail header names.

**Parameters:**

*pusrhdr* pointer to the user mail header list. if user mail header name list is less than [POP3\\_MAXUSERHEADERSIZE](#), the last item of user mail header namelist should be NULL.

**See also:**

[pop3\\_receivemail](#)

## 8.29 tini400\_smtp.h File Reference

### 8.29.1 Detailed Description

SMTP Library functions for DS80C400 processor.

This library contains functions for sending mails to smtp mailhost server

For detailed information on the DS80C400 please see the [High-Speed Microcontroller User's Guide: DS80C400 Supplement](#).

**Warning:**

The functions in this library are **NOT** multi-process safe—that is, if you call the same method from two different processes at the same time, the parameters to the function may be destroyed, yielding unpredictable results.

**Data Structures**

- struct [\\_hostinfo](#)
- struct [\\_mailheader](#)
- struct [\\_userheader](#)

**Defines**

- #define [MAX\\_LINE\\_SIZE](#) 1024
- #define [SMTP\\_VERSION](#) 3
- #define [SMTP\\_MAXATTACHMENTSIZ](#) 5
- #define [SMTP\\_MAXUSERHEADERSIZ](#) 20
- #define [SMTP\\_INSUFFICIENT\\_MEMORY](#) -1
- #define [SMTP\\_MAILHOST\\_NOT\\_FOUND](#) -3
- #define [SMTP\\_FILE\\_NOT\\_FOUND](#) -4
- #define [SMTP\\_SOCKET\\_ERROR](#) -7
- #define [SMTP\\_MAIL\\_QUEUED](#) -8
- #define [SMTP\\_INVALID\\_MAILNODE\\_ADDRESS](#) -9
- #define [SMTP\\_LIBRARY\\_IS\\_NOT\\_CONFIGURED](#) -11
- #define [SMTP\\_STATUS\\_SUCCESS](#) 0

**Typedefs**

- typedef [\\_hostinfo](#) hostinfo
- typedef [\\_mailheader](#) mailheader

*Structure for standard mail header holds standard mail header values.*

- typedef [\\_userheader](#) userheader

*Structure for user defined mail header contains user header name list and user header value list.*

**Functions**

- void [smtp\\_init](#) (void)  
*Initializes smtp library.*

- void [smtp\\_sethostinfo](#) (struct [\\_hostinfo](#) \*phostinfo)  
*Sets the host information object with smtp library.*
- void [smtp\\_setclientsockettimeout](#) (long milli\_sec)  
*Sets SMTP client socket timeout value.*
- long [smtp\\_getclientsockettimeout](#) (void)  
*Returns the current value for SMTP client socket timeout.*
- void [smtp\\_setdefaulttheadervalue](#) (struct [\\_mailheader](#) \*pmhdr)  
*Sets the default value for standard mail headers.*
- void [smtp\\_setuserheaderlist](#) (struct [\\_userheader](#) \*pusrhdr)  
*Sets user defined mail header list.*
- int [smtp\\_sendmail](#) (struct [\\_mailheader](#) mail\_header, char \*msg, char \*attachmentlist[SMTP\_MAXATTACHMENTSIZ], char queuemail\_flag, unsigned long \*mailnodeaddress)  
*Sends mail to mail host.*
- int [smtp\\_removemailfromqueue](#) (unsigned long pmailnode\_address)  
*Removes queued mail from mail queue list.*
- int [smtp\\_getqueuedmailstatus](#) (unsigned long pmailnode\_address)  
*Returns the status of queued mail.*

## 8.29.2 Define Documentation

### 8.29.2.1 #define MAX\_LINE\_SIZE 1024

Definition for maximum size of mail header

### 8.29.2.2 #define SMTP\_FILE\_NOT\_FOUND -4

File not found error value

See also:

[smtp\\_sendmail](#)  
[smtp\\_getqueuedmailstatus](#)



#### **8.29.2.3 #define SMTP\_INSUFFICIENT\_MEMORY -1**

Insufficient memory error value

See also:

[smtp\\_sendmail](#)

#### **8.29.2.4 #define SMTP\_INVALID\_MAILNODE\_ADDRESS -9**

Invalid mailnode address error value.

See also:

[smtp\\_getqueuedmailstatus](#)

[smtp\\_removemailfromqueue](#)

#### **8.29.2.5 #define SMTP\_LIBRARY\_IS\_NOT\_CONFIGURED -11**

smtp library is not configured error value, this value will be returned if smtp host information is not configured.

See also:

[smtp\\_sendmail](#)

#### **8.29.2.6 #define SMTP\_MAIL\_QUEUED -8**

Mail is queued error value

See also:

[smtp\\_sendmail](#)

[smtp\\_getqueuedmailstatus](#)

#### **8.29.2.7 #define SMTP\_MAILHOST\_NOT\_FOUND -3**

Mail host is not found error value

See also:

[smtp\\_sendmail](#)

[smtp\\_getqueuedmailstatus](#)

#### **8.29.2.8 #define SMTP\_MAXATTACHMENTSIZ 5**

Definition for maximum number of attachments

See also:

[smtp\\_sendmail](#)

#### 8.29.2.9 **#define SMTP\_MAXUSERHEADERSIZE 20**

Definition for maximum number of user headers

See also:

[smtp\\_sendmail](#)

#### 8.29.2.10 **#define SMTP\_SOCKET\_ERROR -7**

Socket error value

See also:

[smtp\\_sendmail](#)

#### 8.29.2.11 **#define SMTP\_STATUS\_SUCCESS 0**

smtp Status success value, this value is returned when operation is completed successfully.

See also:

[smtp\\_sendmail](#)

#### 8.29.2.12 **#define SMTP\_VERSION 3**

Version number associated with this header file. Should be the same as the version number returned by the [smtp\\_version](#) function.

See also:

[smtp\\_version](#)

### 8.29.3 Typedef Documentation

#### 8.29.3.1 **typedef struct [\\_hostinfo](#) hostinfo**

Structure for host configuration information that has to be registered with smtp library

### 8.29.4 Function Documentation

#### 8.29.4.1 **long smtp\_getclientsockettimeout (void)**

Returns the current value for SMTP client socket timeout.

This function returns the current value for SMTP client socket timeout

This function is safe to be called from multiple processes at the same time.

**Returns:**

The current value for http client socket timeout

**8.29.4.2 int smtp\_getqueuedmailstatus (unsigned long *pmailnode\_address*)**

Returns the status of queued mail.

This function returns the status of mail which was queued by [smtp\\_sendmail](#).

**Parameters:**

*pmailnode\_address* - address of mailnode. this value should be same value returned by smtp\_sendmail function when queueing mail.

**Returns:**

if mail is still in queue, returns the status of mail. [SMTP\\_INVALID\\_MAILNODE\\_ADDRESS](#) if invalid mail node address is passed or mail has been already sent to mailhost

**See also:**

[smtp\\_removemailfromqueue](#)  
[smtp\\_sendmail](#)

**8.29.4.3 void smtp\_init (void)**

Initializes smtp library.

This function initializes the internal data structures of smtp library. This function should be called first before calling any other functions of smtp library.

**NOTE:** Other libraries don't need to be initialized before smtp library initialization.

**8.29.4.4 int smtp\_removemailfromqueue (unsigned long *pmailnode\_address*)**

Removes queued mail from mail queue list.

This function removes the mail which was queued by [smtp\\_sendmail](#).

**Parameters:**

*pmailnode\_address* - address of mailnode to delete. this value should be same value returned by smtp\_sendmail function when queueing mail.

**Returns:**

[SMTP\\_STATUS\\_SUCCESS](#) if mailnode was deleted successfully. [SMTP\\_INVALID\\_MAILNODE\\_ADDRESS](#) if invalid mail node address is passed or mail has been already sent to mailhost

See also:

[smtp\\_getqueuedmailstatus](#)  
[smtp\\_sendmail](#)

**8.29.4.5** `int smtp_sendmail (struct \_mailheader mail_header, char * msg, char * attachmentlist[SMTP_MAXATTACHMENTSIZ], char queueemail_flag, unsigned long * mailnodeaddress)`

Sends mail to mail host.

This function sends mail to mailhost. if smtp host IP address is zero, this function uses dns library to get IP address of target mailhost. if mail host is down and application sets queueemail\_flag=1, Mail will be queued to resend later. this function uses base64 MIME encryption for sending attachments. it does not use any encryption for message

**Parameters:**

*mail\_header* standard mail header object. any uninitialized field name in this structure should be set with NULL value. if default mail header value was initialized and mail\_header field value is NULL, the default mail header value will be used.

*msg* pointer to mail message.

*attachmentlist* array of string holds attachment filelist. if attachment list is less than SMTP\_MAXATTACHMENTSIZ, last element of list should be NULL to indicate end of the list. if there is no attachment to send, then, this argument can be NULL.

*queueemail\_flag* flag to indicate whether mail to be queued or not. if mail host is down and application sets queueemail\_flag=1, mail will be queued.

*mailnodeaddress* address of mail which was queued to resend. this reference value has to be passed to get status of queued mail or to delete it from queue.

**Returns:**

[SMTP\\_STATUS\\_SUCCESS](#) if the operation is completed successfully Otherwise, one of the following error values

- [SMTP\\_LIBRARY\\_IS\\_NOT\\_CONFIGURED](#)
- [SMTP\\_INSUFFICIENT\\_MEMORY](#)
- [SMTP\\_MAILHOST\\_NOT\\_FOUND](#)
- [SMTP\\_MAIL\\_QUEUED](#)
- [SMTP\\_SOCKET\\_ERROR](#) if there is any socket error, or one of smtp server negative response error code which is listed out in rfc821

See also:

[smtp\\_removemailfromqueue](#)  
[smtp\\_getqueuedmailstatus](#)

#### 8.29.4.6 void smtp\_setclientsockettimeout (long *milli\_sec*)

Sets SMTP client socket timeout value.

This function sets SMTP client socket timeout value. The default SMTP client Time out value after initializing smtp library is 30 seconds.

##### **Warning:**

This function is not multi-process safe. If two processes try to call this function at the same time, its parameters may be destroyed, yielding unpredictable results.

##### **Parameters:**

*milli\_sec* timeout value in milliseconds

#### 8.29.4.7 void smtp\_setdefaultheadervalue (struct [\\_mailheader](#) \* *pmhdr*)

Sets the default value for standard mail headers.

This function stores address of mail header structure in smtp library global variable. smtp\_sendmail function uses pmhdr value by default, user can override these values by passing valid standard mail header value while calling smtp\_sendmail function.

**NOTE:** default mail header value is **not mandatory** for sending mail. It is optional feature.

##### **Parameters:**

*pmhdr* pointer to the mail header structure

##### **See also:**

[smtp\\_sendmail](#)

#### 8.29.4.8 void smtp\_sethostinfo (struct [\\_hostinfo](#) \* *phostinfo*)

Sets the host information object with smtp library.

This function stores address of host configuration information structure in smtp library global variable. Then, configures dns library by setting primary and secondary dns server ip addresses. host configuration information is used to connect with SMTP servers.

##### **Parameters:**

*phostinfo* - pointer to the host information structure

##### **See also:**

[smtp\\_sendmail](#)

#### 8.29.4.9 void smtp\_setuserheaderlist (struct [\\_userheader](#) \* *pusrhdr*)

Sets user defined mail header list.

This function stores address of user mail header list structure in smtp library global variable. user defined mail headers will be added while sending mail messages.

**NOTE:** user mail header list is **not mandatory** for sending mail. It is optional feature.

#### Parameters:

*pusrhdr* pointer to the user mail header list. if user mail header name list is less than SMTP\_MAXUSERHEADERSIZE, the last item of user mail header name list should be NULL.

#### See also:

[smtp\\_sendmail](#)

## 8.30 tini400\_spi.h File Reference

### 8.30.1 Detailed Description

SPI library for the TINIm400 module.

"Bit Bang" software SPI library for use with the TINIm400. This is a full featured SPI library for sending and receiving data. It supports 4 SPI\_CLK polarity and phase modes, slave select with optional inversion and optional synching, 8 and 16 bit transfer modes, bit reordering and SPI\_CLK delays.

Port pins used by this SPI library can be specified in spimacros.inc.

For detailed information on the DS80C400 please see the [High-Speed Microcontroller User's Guide: DS80C400 Supplement](#).

#### Warning:

The functions in this library are **NOT** multi-process safe—that is, if you call the same method from two different processes at the same time, the parameters to the function may be destroyed, yielding unpredictable results. However, SPI pins are a system resource and should not be shared among different processes.

#### Defines

- #define [TINI400\\_SPI\\_VERSION](#) 2
- #define [SPI\\_CKPOL\\_MASK](#) 0x01  
*CKPOL MASK.*
- #define [SPI\\_CKPHA\\_MASK](#) 0x02  
*CKPHA MASK.*

- #define [SPI\\_WORD\\_MASK](#) 0x04  
*Word mode MASK.*
- #define [SPI\\_SKEW\\_MASK](#) 0x08  
*No Skew MASK.*
- #define [SPI\\_USESS\\_MASK](#) 0x10  
*Use SS MASK.*
- #define [SPI\\_SYNCHSS\\_MASK](#) 0x20  
*Synch SS MASK.*
- #define [SPI\\_INVERTSS\\_MASK](#) 0x40  
*Invert SS MASK.*

## Functions

- void [spi\\_init](#) (void)  
*Initialize the SPI library.*
- int [spi\\_reverseBits](#) (int length, int wordSize, unsigned char \*dataptr)  
*Reverse bits in buffer.*
- void [spi\\_xmit](#) (unsigned char \*dataptr, int length, unsigned char delay, unsigned char options)  
*Transmit SPI data.*
- unsigned int [spi\\_version](#) (void)  
*Returns the version number of this SPI library.*

## 8.30.2 Define Documentation

### 8.30.2.1 #define [SPI\\_CKPHA\\_MASK](#) 0x02

CKPHA MASK.

See also:

[SPI\\_CKPOL\\_MASK](#)  
[spi\\_xmit](#)

#### 8.30.2.2 **#define SPI\_CKPOL\_MASK 0x01**

CKPOL MASK.

The four SPI clock (SPI\_CLK) modes supported by this library are defined by CKPHA and CKPOL. The CKPOL bit defines the idle state of the SPI clock, CKPOL = 0 forces SPI\_CLK to idle low while CKPOL = 1 forces SPI\_CLK to idle high. CKPHA changes the edge used to signal transfer of data. When CKPHA = 0 the first edge of SPI\_CLK specifies when the slave and master should sample their input. With CKPHA = 1 the second edge of SPI\_CLK specifies when to sample. When CKPHA = 1, the master and slave should present their data on their output during the first SPI\_CLK edge, this allows the data sufficient hold time. When CKPHA = 0, data should become valid when the Slave Select (SS) line goes active. Note that most devices require the SS line to be used when CKPHA = 0 to allow proper timing while SS may be optional when CKPHA = 1.

**See also:**

[SPI\\_CKPHA\\_MASK](#)

[spi\\_xmit](#)

#### 8.30.2.3 **#define SPI\_INVERTSS\_MASK 0x40**

Invert SS MASK.

Most SPI devices expect the active state for SS to be low, but others require high as the active state.

**See also:**

[spi\\_xmit](#)

#### 8.30.2.4 **#define SPI\_SKEW\_MASK 0x08**

No Skew MASK.

To facilitate atomic transfers, interrupts may be disabled while transmitting.

**See also:**

[spi\\_xmit](#)

#### 8.30.2.5 **#define SPI\_SYNCHSS\_MASK 0x20**

Synch SS MASK.

Some SPI devices expect the SS signal to go inactive after each word transfer in order to synchronize.



**See also:**

[spi\\_xmit](#)

#### **8.30.2.6 #define SPI\_USESS\_MASK 0x10**

Use SS MASK.

The SS signal is optional as it may not be required for all SPI setups.

**See also:**

[spi\\_xmit](#)

#### **8.30.2.7 #define SPI\_WORD\_MASK 0x04**

Word mode MASK.

Data is sent to the SPI library as a character array in data memory. When in 8 bit word mode these bytes will be transferred one at a time. In 16 bit word mode 2 bytes will be transferred but this operation will only consume 1 transfer of the number requested. Note that in this library, "word" may be 8 or 16 bits in length depending on the selected mode. Using this mask activates 16 bit word mode

**See also:**

[spi\\_xmit](#)

#### **8.30.2.8 #define TINI400\_SPI\_VERSION 2**

Version number associated with this header file. Should be the same as the version number returned by the [spi\\_version](#) function.

**See also:**

[spi\\_version](#)

### **8.30.3 Function Documentation**

#### **8.30.3.1 int spi\_reverseBits (int *length*, int *wordSize*, unsigned char \* *dataptr*)**

Reverse bits in buffer.

This function can be called to reverse the bits in the passed buffer. It reorders the based on the word mode 8 bit words or 16 bit words. This can be used to convert data for Least Significant Bit (LSB) transfers.

**Parameters:**

***length*** Number of words to bit reverse. Note that for 16 bit words this must be a even value, SPI library does not check this.

**wordSize** Size of the word to reverse. Only 8 and 16 are valid.

**dataptr** Pointer to the data to be reversed, after calling this function the data in this buffers will be bit reversed.

**Returns:**

int 1 for success, -1 if error occurred

### 8.30.3.2 unsigned int spi\_version (void)

Returns the version number of this SPI library.

**Returns:**

int Version number of this SPI library.

### 8.30.3.3 void spi\_xmit (unsigned char \* dataptr, int length, unsigned char delay, unsigned char options)

Transmit SPI data.

Transmits the data passed in over the SPI port, reads and returns any data read back.

**Parameters:**

**dataptr** Pointer to the data to be transmitted, received data is written over transmit data during transfer,

**length** Amount of data to transfer

**delay** Amount of time to delay clock edges, in usec. In order to interface to slower SPI slaves a SPI\_CLK stretch can be used to increase the SPI\_CLK period by 1  $\mu$ sec per stretch.

**options** SPI configuration options defined as:

- bit 0 - CPOL - Set to 1 - SPICLK idles high
- bit 1 - CPHA - Set to 1 - Transfers on second edge
- bit 2 - wordMode - Set to 1 - 16 bit transfers
- bit 3 - noskew - Set to 1 - turn off interrupts during transfer
- bit 4 - useSS - Set to 1 - Use the SS line during transfers
- bit 5 - synchSS - Set to 1 - Takes SS to inactive after every word
- bit 6 - invertSS - Set to 1 - SS line is active high

## 8.31 tini400\_time.h File Reference

### 8.31.1 Detailed Description

Date/Time utilities, tailored for the DS80C400 C Libraries.

This library contains functions that provide simple time utilities in conjunction with the RTC C Library. The time base is variable for this library, meaning that the value '0 seconds' can be assigned to 12:00:00am of January 1st for a specific year. Note that this library does not currently support daylight savings time computations or the concept of time zones.

Note that this library will not return correct values for dates before the year 1901 or after the year 2099.

For detailed information on the DS80C400 please see the [High-Speed Microcontroller User's Guide: DS80C400 Supplement](#).

#### **Warning:**

The functions in this library are **NOT** multi-process safe—that is, if you call the same method from two different processes at the same time, the parameters to the function may be destroyed, yielding unpredictable results. Consult each individual function's documentation for details on which functions are multi-process safe.

#### **Data Structures**

- struct [tm](#)

#### **Defines**

- #define [TINI400\\_TIME\\_VERSION](#) 2

#### **Typedefs**

- typedef unsigned long [time\\_t](#)

#### **Functions**

- unsigned int [time\\_version](#) (void)  
*Returns the version number of this [TIME](#) library.*
- void [time\\_settimebase](#) (unsigned int year)  
*Sets the time base year for the RTC.*
- [time\\_t](#) [mktime](#) (struct [tm](#) \*timeptr)  
*mktime*
- [time\\_t](#) [time](#) ([time\\_t](#) \*timer)  
*time*

- `tm * gmtime (time_t *timer)`  
*gmtime*

## 8.31.2 Define Documentation

### 8.31.2.1 `#define TINI400_TIME_VERSION 2`

Version number associated with this header file. Should be the same as the version number returned by the *time\_version* function.

See also:

*time\_version*

## 8.31.3 Typedef Documentation

### 8.31.3.1 `typedef unsigned long time_t`

Type used for representing time. Our RTC is assumed to be 4 bytes of seconds.

See also:

*time*

## 8.31.4 Function Documentation

### 8.31.4.1 `struct tm* gmtime (time_t * timer)`

*gmtime*

Converts the native time formatted input into a calendar representation.

**Parameters:**

*timer* Native representation of the time to be converted to calendar format.

**Returns:**

Calendar format of the input time.

### 8.31.4.2 `time_t mktime (struct tm * timeptr)`

*mktime*

Converts a *tm* structure (calendar time) into the native time representation of *time\_t*. The time is computed using the hour, minute, second, day of month, month, and year fields of the input structure. The day of year, day of week, and daylight savings time flag are ignored. No bounds checking is performed on the input data.

**Parameters:**

*timeptr* Calendar time to be converted to native time representation

**Returns:**

Native time representation of the calendar.

**8.31.4.3 `time_t` time (`time_t * timer`)**

time

Gets the current time in its native representation format. Use the function [gmtime](#) to get a calendar representation of this time.

**Parameters:**

*timer* If non-null, this is also filled in with the return value

**Returns:**

Native time representation of the current time.

**8.31.4.4 `void` time\_settimebase (`unsigned int year`)**

Sets the time base year for the RTC.

Sets the time base year for the real time clock. The recommended time base is the year 2000. The time base must be set before meaningful calculations can occur.

**Parameters:**

*year* base year which will be used for time computations

**8.31.4.5 `unsigned int` time\_version (`void`)**

Returns the version number of this [TIME](#) library.

**Returns:**

Version number of this [TIME](#) library.

## **8.32 `tini400_xnetboot.h` File Reference**

### **8.32.1 Detailed Description**

External NetBoot library for the DS80C400.

The External Netboot library contains netboot code that can be invoked independently from the ROM. This library provides the latest NetBoot code that adds the following

features: Improves TBIN2 loading to work with files larger than 64KB, disables all multicast traffic reception to improve reliability, supports the DS2502 and the DS1982 to hold a MAC ID (in addition to the DS2502-E48), supports setting the clock multiplier for improved performance, supports acquiring a DHCP IP from the Netgear WGT624 router.

**This library works with IPv4 only.**

The External Netboot library cannot reprogram the same flash chip it is running from, i.e. you need two separate flash memories.

You can use the library from assembly language - set r7 to the desired clock multiplier and jump to the *XNETBOOT* symbol.

```
EXTERN ECODE(XNETBOOT) mov r7, #2 ljmp XNETBOOT
```

**Warning:**

Note that debug symbols have to be turned off in order to avoid a linker error (the linker cannot handle line numbers greater than 65534 and will return an "L220" error when debug symbols are enabled).

**Defines**

- #define [TINI400\\_XNETBOOT\\_VERSION](#) 2

**Functions**

- unsigned int [xnetboot\\_version](#) (void)  
*Returns the version number of this XNETBOOT library.*
- void [xnetboot\\_boot](#) (unsigned char multiplier)  
*Starts NetBoot.*

**8.32.2 Define Documentation**

**8.32.2.1 #define TINI400\_XNETBOOT\_VERSION 2**

Version number associated with this header file. Should be the same as the version number returned by the [xnetboot\\_version](#) function.

**See also:**

[xnetboot\\_version](#)

### 8.32.3 Function Documentation

#### 8.32.3.1 void xnetboot\_boot (unsigned char *multiplier*)

Starts NetBoot.

This function starts NetBoot and does not return to the caller.

**Parameters:**

*multiplier* The argument *multiplier* sets the clock multiplier (1, 2, or 4).

#### 8.32.3.2 unsigned int xnetboot\_version (void)

Returns the version number of this XNETBOOT library.

**Returns:**

Version number of this XNETBOOT library.

## 8.33 tini\_i2c.h File Reference

### 8.33.1 Detailed Description

I2C function library.

This library contains functions for communicating to I2C devices via user specified port pins.

For detailed information on the DS80C400 please see the [High-Speed Microcontroller User's Guide: DS80C400 Supplement](#).

**Warning:**

The functions in this library are **NOT** multi-process safe—that is, if you call the same method from two different processes at the same time, the parameters to the function may be destroyed, yielding unpredictable results. However, I2C pins are a system resource and should not be shared among different processes.

**Defines**

- #define [TINI\\_I2C\\_VERSION](#) 1
- #define [I2C\\_SDA](#) P3\_4
- #define [I2C\\_SCL](#) P3\_5
- #define [I2C\\_ENABLE\\_SCL\\_WAIT\\_FOR\\_SLOW\\_SLAVES](#) 0
- #define [I2C\\_MAXIMUM\\_SCL\\_WAITCOUNT](#) 10000
- #define [I2C\\_DELAY\\_LOOP\\_COUNT](#) 0

## Functions

- int [i2c\\_version](#) ()  
*Return the library version.*
- void [i2c\\_delay](#) (void)  
*Delay function.*
- void [i2c\\_start](#) (void)  
*Performs an I2C start condition.*
- void [i2c\\_bit](#) (unsigned char singlebit)  
*Performs an I2C bit write.*
- unsigned char [i2c\\_readbit](#) (void)  
*Performs an I2C bit read.*
- void [i2c\\_stop](#) (void)  
*Performs an I2C stop condition.*
- unsigned char [i2c\\_readbyte](#) (unsigned char doACK)  
*Performs an I2C byte read.*
- unsigned char [i2c\\_writebyte](#) (unsigned char singlebyte)  
*Performs an I2C byte write.*
- unsigned char [i2c\\_select](#) (unsigned char address)  
*Perform I2C start, address selection.*
- unsigned char [i2c\\_writeblock](#) (unsigned char address, unsigned char \*barr, int length)  
*Perform I2C start, address selection, write specified bytes and I2C stop.*
- unsigned char [i2c\\_readblock](#) (unsigned char address, unsigned char \*barr, int length)  
*Perform I2C start, address selection, read specified number of bytes and I2C stop.*
- unsigned char [i2c\\_writereadblock](#) (unsigned char address, unsigned char \*barr1, int length1, unsigned char \*barr2, int length2)  
*Perform I2C start, address selection, write specified bytes, I2C start, address selection, read bytes and I2C stop.*



### 8.33.2 Define Documentation

#### 8.33.2.1 `#define I2C_DELAY_LOOP_COUNT 0`

Number of loops to wait between any host SCL and SDA transitions

#### 8.33.2.2 `#define I2C_ENABLE_SCL_WAIT_FOR_SLOW_SLAVES 0`

Enable communication with slow slave devices. Value of 1 enables SCL waiting/flow control.

#### 8.33.2.3 `#define I2C_MAXIMUM_SCL_WAITCOUNT 10000`

Number of loops to wait for SCL to return high if SCL flow control is used.

#### 8.33.2.4 `#define I2C_SCL P3_5`

Define SCL (clock) line to talk to the DS1672 on the TINIm400

#### 8.33.2.5 `#define I2C_SDA P3_4`

Define SDA (data) line to talk to the DS1672 on the TINIm400

#### 8.33.2.6 `#define TINI_I2C_VERSION 1`

Version number associated with this header file. Should be the same as the version number returned by the [\*i2c\\_version\*](#) function.

See also:

[\*i2c\\_version\*](#)

### 8.33.3 Function Documentation

#### 8.33.3.1 `void i2c_bit (unsigned char singlebit)`

Performs an I2C bit write.

**Parameters:**

*singlebit* Bit to write on I2C bus

#### 8.33.3.2 `unsigned char i2c_readbit (void)`

Performs an I2C bit read.

**Returns:**

Value of SDA line during read timeslot

#### **8.33.3.3 unsigned char i2c\_readblock (unsigned char *address*, unsigned char \**barr*, int *length*)**

Perform I2C start, address selection, read specified number of bytes and I2C stop.

##### **Parameters:**

*address* Address of device to select. Upper 7 bits are address, LSbit automatically set to 1 by function.

*barr* Array destination for read bytes

*length* Number of bytes to read

##### **Returns:**

0 if device acknowledged address selection and data transfer

#### **8.33.3.4 unsigned char i2c\_readbyte (unsigned char *doACK*)**

Performs an I2C byte read.

##### **Parameters:**

*doACK* Set to 1 to assert acknowledge after reading 8 bits, or 0 to not assert ACK.

##### **Returns:**

Value of SDA line during read timeslot

#### **8.33.3.5 unsigned char i2c\_select (unsigned char *address*)**

Perform I2C start, address selection.

##### **Parameters:**

*address* Address of device to select. Upper 7 bits are address, LSbit denotes read if 1 and write if 0.

##### **Returns:**

0 if device acknowledged address selection

#### **8.33.3.6 unsigned char i2c\_writeblock (unsigned char *address*, unsigned char \**barr*, int *length*)**

Perform I2C start, address selection, write specified bytes and I2C stop.

##### **Parameters:**

*address* Address of device to select. Upper 7 bits are address, LSbit automatically set to 0 by function.

*barr* Array of bytes to write  
*length* Number of bytes to write

**Returns:**

0 if device acknowledged address selection and data transfer

**8.33.3.7 unsigned char i2c\_writebyte (unsigned char *singlebyte*)**

Performs an I2C byte write.

**Parameters:**

*singlebyte* Value to write to bus.

**Returns:**

0 if byte was acknowledged

**8.33.3.8 unsigned char i2c\_writereadblock (unsigned char *address*, unsigned char \* *barr1*, int *length1*, unsigned char \* *barr2*, int *length2*)**

Perform I2C start, address selection, write specified bytes, I2C start, address selection, read bytes and I2C stop.

**Parameters:**

*address* Address of device to select. Upper 7 bits are address, LSbit automatically set to 0 by function.

*barr1* Array of bytes to write

*length1* Number of bytes to write

*barr2* Array destination for read bytes

*length2* Number of bytes to read

**Returns:**

0 if device acknowledged address selection and data transfer

## **8.34 tini\_rtc.h File Reference**

### **8.34.1 Detailed Description**

RTC function library.

This library contains RTC functions for the DS1672U, the real time clock included in the TINIm400 reference module.

For detailed information on the DS1672U, please see the [Low-Voltage Serial Timekeeping Chip](#).

**Warning:**

The functions in this library are **NOT** multi-process safe—that is, if you call the same method from two different processes at the same time, the parameters to the function may be destroyed, yielding unpredictable results.

**Defines**

- #define `DEVICE_ADDRESS` 0xD0
- #define `COUNTER_ADDRESS` 0x00
- #define `CONTROL_ADDRESS` 0x04
- #define `TRICKLECHARGER_ADDRESS` 0x05
- #define `TRICKLECHARGER_DISABLE` 0xF0
- #define `START_CLOCK` 0x7F
- #define `STOP_CLOCK` 0x80
- #define `NODIODE_250OHM` 0xA5
- #define `ONEDIODE_250OHM` 0xA9
- #define `NODIODE_2KOHM` 0xA6
- #define `ONEDIODE_2KOHM` 0xAA
- #define `NODIODE_4KOHM` 0xA7
- #define `ONEDIODE_4KOHM` 0xAB
- #define `TINI_RTC_VERSION` 1

**Functions**

- int `rtc_version` ()  
*Return the library version.*
- int `rtc_startclock` ()  
*Start oscillator to count clock by setting MSB of control register to 0.*
- int `rtc_stopclock` ()  
*Stop oscillator to pause clock by setting MSB of control register to 1.*
- int `rtc_setcontrolregister` (unsigned char newvalue)  
*Write value to 8 bit control register.*
- int `rtc_getcontrolregister` (unsigned char \*)  
*Fetch value of 8 bit control register.*
- int `rtc_disabletricklecharger` ()  
*Disable trickle charger register by setting 4 LSB's to 0.*

- int [rtc\\_enabletricklecharger0diode250ohm](#) ()  
*Set trickle charger register to work no diode and with 250ohm.*
- int [rtc\\_enabletricklecharger1diode250ohm](#) ()  
*Set trickle charger register to work 1 diode and with 250ohm.*
- int [rtc\\_enabletricklecharger0diode2kohm](#) ()  
*Set trickle charger register to work no diode and with 2Kohm.*
- int [rtc\\_enabletricklecharger1diode2kohm](#) ()  
*Set trickle charger register to work 1 diode and with 2Kohm.*
- int [rtc\\_enabletricklecharger0diode4kohm](#) ()  
*Set trickle charger register to work no diode and with 4Kohm.*
- int [rtc\\_enabletricklecharger1diode4kohm](#) ()  
*Set trickle charger register to work 1 diode and with 4Kohm.*
- int [rtc\\_settricklechargerregister](#) (unsigned char newvalue)  
*Set trickle charger register new value.*
- int [rtc\\_gettricklechargerregister](#) (unsigned char \*)  
*Fetch 8 bit trickle charger register content.*
- int [rtc\\_getclock](#) (long \*)  
*Convert char array to long integer after fetch from 32 bit counter of RTC.*
- int [rtc\\_setclock](#) (long newvalue)  
*Convert long integer to char array and write to 32 bit counter of RTC.*

### 8.34.2 Define Documentation

#### 8.34.2.1 #define CONTROL\_ADDRESS 0x04

Address of Control register.

See also:

[rtc\\_setcontrolregister](#)  
[rtc\\_getcontrolregister](#)

#### **8.34.2.2 #define COUNTER\_ADDRESS 0x00**

Starting address of 32 bits RTC counter.

See also:

[rtc\\_getclock](#)  
[rtc\\_setclock](#)

#### **8.34.2.3 #define DEVICE\_ADDRESS 0xD0**

Device address.

#### **8.34.2.4 #define NODIODE\_250OHM 0xA5**

Value of Trickle Charger register that connects Vcc & Vbackup via no diode and 250 ohm resistor when Trickle Charger is enabled .

See also:

[rtc\\_enabletricklecharger0diode250ohm](#)

#### **8.34.2.5 #define NODIODE\_2KOHM 0xA6**

Value of Trickle Charger register that connects Vcc & Vbackup via no diode and 2K ohm resistor when Trickle Charger is enabled .

See also:

[rtc\\_enabletricklecharger0diode2kohm](#)

#### **8.34.2.6 #define NODIODE\_4KOHM 0xA7**

Value of Trickle Charger register that connects Vcc & Vbackup via no diode and 4K ohm resistor when Trickle Charger is enabled .

See also:

[rtc\\_enabletricklecharger0diode4kohm](#)

#### **8.34.2.7 #define ONEDIODE\_250OHM 0xA9**

Value of Trickle Charger register that connects Vcc & Vbackup via one diode and 250 ohm resistor when Trickle Charger is enabled .

See also:

[rtc\\_enabletricklecharger1diode250ohm](#)

#### **8.34.2.8 #define ONEDIODE\_2KOHM 0xAA**

Value of Trickle Charger register that connects Vcc & Vbackup via one diode and 2K ohm resistor when Trickle Charger is enabled .

See also:

[rtc\\_enabletricklecharger1diode2kohm](#)

#### **8.34.2.9 #define ONEDIODE\_4KOHM 0xAB**

Value of Trickle Charger register that connects Vcc & Vbackup via one diode and 4K ohm resistor when Trickle Charger is enabled .

See also:

[rtc\\_enabletricklecharger1diode4kohm](#)

#### **8.34.2.10 #define START\_CLOCK 0x7F**

Value of Control register that will start oscillator.

See also:

[rtc\\_startclock](#)

#### **8.34.2.11 #define STOP\_CLOCK 0x80**

Value of Control register that will stop oscillator.

See also:

[rtc\\_startclock](#)

#### **8.34.2.12 #define TINI\_RTC\_VERSION 1**

Version number associated with this header file. Should be the same as the version number returned by the [rtc\\_version](#) function.

See also:

[rtc\\_version](#)

#### **8.34.2.13 #define TRICKLECHARGER\_ADDRESS 0x05**

Address of Trickle Charger register.

See also:

[rtc\\_gettricklechargerregister](#)  
[rtc\\_settricklechargerregister](#)

#### 8.34.2.14 `#define TRICKLECHARGER_DISABLE 0xF0`

Value of Trickle Charger register that will disable it.

See also:

[rtc\\_disabletricklecharger](#)

### 8.34.3 Function Documentation

#### 8.34.3.1 `int rtc_disabletricklecharger ()`

Disable trickle charger register by setting 4 LSB's to 0.

**Returns:**

0 if pass, -1 if fail

See also:

[rtc\\_enabletricklecharger0diode250ohm](#)

#### 8.34.3.2 `int rtc_enabletricklecharger0diode250ohm ()`

Set trickle charger register to work no diode and with 250ohm.

**Returns:**

0 if pass, -1 if fail

See also:

[rtc\\_disabletricklecharger](#)

[rtc\\_enabletricklecharger1diode250ohm](#)

#### 8.34.3.3 `int rtc_enabletricklecharger0diode2kohm ()`

Set trickle charger register to work no diode and with 2Kohm.

**Returns:**

0 if pass, -1 if fail

See also:

[rtc\\_disabletricklecharger](#)

[rtc\\_enabletricklecharger1diode2kohm](#)



#### **8.34.3.4 int rtc\_enabletricklecharger0diode4kohm ()**

Set trickle charger register to work no diode and with 4Kohm.

**Returns:**

0 if pass, -1 if fail

**See also:**

[rtc\\_disabletricklecharger](#)

[rtc\\_enabletricklecharger1diode4kohm](#)

#### **8.34.3.5 int rtc\_enabletricklecharger1diode250ohm ()**

Set trickle charger register to work 1 diode and with 250ohm.

**Returns:**

0 if pass, -1 if fail

**See also:**

[rtc\\_disabletricklecharger](#)

[rtc\\_enabletricklecharger0diode2kohm](#)

#### **8.34.3.6 int rtc\_enabletricklecharger1diode2kohm ()**

Set trickle charger register to work 1 diode and with 2Kohm.

**Returns:**

0 if pass, -1 if fail

**See also:**

[rtc\\_disabletricklecharger](#)

[rtc\\_enabletricklecharger0diode4kohm](#)

#### **8.34.3.7 int rtc\_enabletricklecharger1diode4kohm ()**

Set trickle charger register to work 1 diode and with 4Kohm.

**Returns:**

0 if pass, -1 if fail

**See also:**

[rtc\\_disabletricklecharger](#)

[rtc\\_enabletricklecharger0diode250ohm](#)

#### **8.34.3.8 int rtc\_getclock (long \*)**

Convert char array to long integer after fetch from 32 bit counter of RTC.

**Returns:**

0 if pass, -1 if fail

**See also:**

[rtc\\_setclock](#)

#### **8.34.3.9 int rtc\_getcontrolregister (unsigned char \*)**

Fetch value of 8 bit control register.

**Returns:**

0 if pass, -1 if fail

**See also:**

[rtc\\_setcontrolregister](#)

#### **8.34.3.10 int rtc\_gettricklechargerregister (unsigned char \*)**

Fetch 8 bit trickle charger register content.

**Returns:**

0 if pass, -1 if fail

**See also:**

[rtc\\_settricklechargerregister](#)

#### **8.34.3.11 int rtc\_setclock (long *newvalue*)**

Convert long integer to char array and write to 32 bit counter of RTC.

**Parameters:**

*newvalue* Value in long integer.

**Returns:**

0 if pass, -1 if fail

**See also:**

[rtc\\_getclock](#)

#### 8.34.3.12 `int rtc_setcontrolregister (unsigned char newvalue)`

Write value to 8 bit control register.

**Parameters:**

*newvalue* Value to set.

**Returns:**

0 if pass, -1 if fail

**See also:**

[rtc\\_getcontrolregister](#)

#### 8.34.3.13 `int rtc_settricklechargerregister (unsigned char newvalue)`

Set trickle charger register new value.

**Parameters:**

*newvalue* Value to set

**Returns:**

0 if pass, -1 if fail

**See also:**

[rtc\\_gettricklechargerregister](#)

#### 8.34.3.14 `int rtc_startclock ()`

Start oscillator to count clock by setting MSB of control register to 0.

**Returns:**

RTC version.

**See also:**

[rtc\\_stopclock](#)

#### 8.34.3.15 `int rtc_stopclock ()`

Stop oscillator to pause clock by setting MSB of control register to 1.

**Returns:**

0 if pass, -1 if fail

**See also:**

[rtc\\_startclock](#)

#### **8.34.3.16 int rtc\_version ()**

Return the library version.

**See also:**

[rtc\\_startclock](#)

## Index

- `_getkey`
  - `stdio.h`, [120](#)
- `accept`
  - `rom400_sock.h`, [46](#)
- `acceptqueue`
  - `rom400_sock.h`, [62](#)
- `AF_INET`
  - `rom400_sock.h`, [46](#)
- `AF_INET6`
  - `rom400_sock.h`, [46](#)
- `arp_cacherequest`
  - `rom400_sock.h`, [46](#)
- `arp_generaterequest`
  - `rom400_sock.h`, [47](#)
- `avail`
  - `rom400_sock.h`, [47](#)
- `bind`
  - `rom400_sock.h`, [47](#)
- `bogus_ptr`
  - `sockaddr`, [6](#)
  - `sockaddr_in`, [6](#)
- `cleanup`
  - `rom400_sock.h`, [48](#)
- `clear_param_buffers`
  - `rom400_sock.h`, [62](#)
- `clearerr`
  - `stdio.h`, [120](#)
- `closesocket`
  - `rom400_sock.h`, [48](#)
- `connect`
  - `rom400_sock.h`, [49](#)
- `crypt_sha1`
  - `tini400_crypt.h`, [140](#)
- `crypt_version`
  - `tini400_crypt.h`, [140](#)
- `dhcp_init`
  - `rom400_dhcp.h`, [11](#)
- `dhcp_registernotify`
  - `rom400_dhcp.h`, [11](#)
- `dhcp_status`
  - `rom400_dhcp.h`, [12](#)
- `DHCP_STATUS_BOUND`
  - `rom400_dhcp.h`, [9](#)
- `DHCP_STATUS_INIT`
  - `rom400_dhcp.h`, [10](#)
- `DHCP_STATUS_INITREBOOT`
  - `rom400_dhcp.h`, [10](#)
- `DHCP_STATUS_REBINDING`
  - `rom400_dhcp.h`, [10](#)
- `DHCP_STATUS_REBOOTING`
  - `rom400_dhcp.h`, [10](#)
- `DHCP_STATUS_RENEWING`
  - `rom400_dhcp.h`, [10](#)
- `DHCP_STATUS_REQUESTING`
  - `rom400_dhcp.h`, [10](#)
- `DHCP_STATUS_SELECTING`
  - `rom400_dhcp.h`, [11](#)
- `dhcp_stop`
  - `rom400_dhcp.h`, [12](#)
- `dhcp_version`
  - `rom400_dhcp.h`, [12](#)
- `dns_enableipv6queries`
  - `tini400_dns.h`, [142](#)
- `dns_getmx`
  - `tini400_dns.h`, [142](#)
- `dns_getprimary`
  - `tini400_dns.h`, [143](#)
- `dns_getsecondary`
  - `tini400_dns.h`, [143](#)
- `dns_gettimeout`
  - `tini400_dns.h`, [144](#)
- `dns_init`
  - `tini400_dns.h`, [144](#)
- `dns_setprimary`
  - `tini400_dns.h`, [144](#)
- `dns_setsecondary`
  - `tini400_dns.h`, [144](#)
- `dns_settimeout`
  - `tini400_dns.h`, [145](#)
- `dns_version`
  - `tini400_dns.h`, [145](#)

EOF	stdio.h, <a href="#">117</a>	FLAG_IO_WAIT	rom400_task.h, <a href="#">80</a>
error	file_structure, <a href="#">3</a>	FLAG_SLEEPING	rom400_task.h, <a href="#">81</a>
ETH_STATUS_LINK	rom400_sock.h, <a href="#">49</a>	Flags	TCB, <a href="#">7</a>
fclose	stdio.h, <a href="#">121</a>	flags	file_structure, <a href="#">3</a>
fd	file_structure, <a href="#">3</a>	flash_eraseblock	rom400_flash.h, <a href="#">15</a>
feof	stdio.h, <a href="#">121</a>	flash_programbyte	rom400_flash.h, <a href="#">15</a>
ferror	stdio.h, <a href="#">121</a>	flash_version	rom400_flash.h, <a href="#">16</a>
fexists	stdio.h, <a href="#">122</a>	flockfile	stdio.h, <a href="#">125</a>
fflush	stdio.h, <a href="#">122</a>	fopen	stdio.h, <a href="#">125</a>
fgetc	stdio.h, <a href="#">122</a>	fopen_fd	stdio.h, <a href="#">126</a>
fgetpos	stdio.h, <a href="#">123</a>	FOPEN_MAX	stdio.h, <a href="#">118</a>
fgets	stdio.h, <a href="#">123</a>	fpos_t	stdio.h, <a href="#">120</a>
FILE	stdio.h, <a href="#">120</a>	fputc	stdio.h, <a href="#">126</a>
FILE_FLAGS_EOF	stdio.h, <a href="#">117</a>	fputs	stdio.h, <a href="#">127</a>
FILE_FLAGS_TEMP	stdio.h, <a href="#">118</a>	fread	stdio.h, <a href="#">127</a>
file_structure, <a href="#">2</a>		freadbytes	stdio.h, <a href="#">128</a>
error, <a href="#">3</a>		freopen	stdio.h, <a href="#">128</a>
fd, <a href="#">3</a>		FS_VERSION	stdio.h, <a href="#">118</a>
flags, <a href="#">3</a>		fseek	stdio.h, <a href="#">129</a>
type, <a href="#">3</a>		fseeko	stdio.h, <a href="#">129</a>
FILE_TYPE_TINIFS	stdio.h, <a href="#">118</a>	fsetpos	stdio.h, <a href="#">130</a>
FILENAME_MAX	stdio.h, <a href="#">118</a>	ftell	stdio.h, <a href="#">130</a>
filesystem_version	stdio.h, <a href="#">124</a>	ftello	stdio.h, <a href="#">131</a>
finit	stdio.h, <a href="#">124</a>		
FLAG_DHCP_WAIT	rom400_task.h, <a href="#">80</a>		

- fttrylockfile
  - stdio.h, [131](#)
- funlockfile
  - stdio.h, [132](#)
- fwrite
  - stdio.h, [132](#)
- fwritebytes
  - stdio.h, [133](#)
- getc
  - stdio.h, [133](#)
- getchar
  - stdio.h, [134](#)
- getethernetstatus
  - rom400\_sock.h, [49](#)
- getfreefsram
  - stdio.h, [134](#)
- gethostbyaddr
  - tini400\_dns.h, [145](#)
- gethostbyname
  - tini400\_dns.h, [146](#)
- getip6params
  - rom400\_sock.h, [50](#)
- getmacid
  - rom400\_sock.h, [50](#)
- getnetworkparams
  - rom400\_sock.h, [50](#)
- getpeername
  - rom400\_sock.h, [51](#)
- gets
  - stdio.h, [134](#)
- getsockname
  - rom400\_sock.h, [51](#)
- getsockopt
  - rom400\_sock.h, [52](#)
- getftpserver
  - rom400\_sock.h, [52](#)
- h\_addr\_list
  - hostent, [4](#)
- h\_addrtype
  - hostent, [3](#)
- h\_aliases
  - hostent, [3](#)
- h\_length
  - hostent, [4](#)
- h\_name
  - hostent, [3](#)
  - mailhostent, [5](#)
- hostent, [3](#)
  - h\_addr\_list, [4](#)
  - h\_addrtype, [3](#)
  - h\_aliases, [3](#)
  - h\_length, [4](#)
  - h\_name, [3](#)
- htons
  - rom400\_sock.h, [53](#)
- i2c\_bit
  - tini\_i2c.h, [155](#)
- i2c\_delay
  - tini\_i2c.h, [155](#)
- I2C\_DELAY\_LOOP\_COUNT
  - tini\_i2c.h, [154](#)
- I2C\_ENABLE\_SCL\_WAIT\_FOR\_-SLOW\_SLAVES
  - tini\_i2c.h, [154](#)
- I2C\_MAXIMUM\_SCL\_WAITCOUNT
  - tini\_i2c.h, [154](#)
- i2c\_readbit
  - tini\_i2c.h, [155](#)
- i2c\_readblock
  - tini\_i2c.h, [155](#)
- i2c\_readbyte
  - tini\_i2c.h, [155](#)
- I2C\_SCL
  - tini\_i2c.h, [154](#)
- I2C\_SDA
  - tini\_i2c.h, [154](#)
- i2c\_select
  - tini\_i2c.h, [155](#)
- i2c\_start
  - tini\_i2c.h, [156](#)
- i2c\_stop
  - tini\_i2c.h, [156](#)
- i2c\_version
  - tini\_i2c.h, [156](#)
- i2c\_writeblock
  - tini\_i2c.h, [156](#)
- i2c\_writebyte
  - tini\_i2c.h, [156](#)
- i2c\_writereadblock

tini_i2c.h, <a href="#">156</a>	rom400_init.h, <a href="#">21</a>
ID	INIT_DIVISOR_4MHZ
TCB, <a href="#">7</a>	rom400_init.h, <a href="#">21</a>
in6_addr, <a href="#">4</a>	INIT_DIVISOR_56MHZ
s6_addr, <a href="#">4</a>	rom400_init.h, <a href="#">21</a>
in_addr, <a href="#">4</a>	INIT_DIVISOR_5MHZ
s_addr, <a href="#">5</a>	rom400_init.h, <a href="#">21</a>
inet_addr	INIT_DIVISOR_64MHZ
rom400_sock.h, <a href="#">63</a>	rom400_init.h, <a href="#">22</a>
inet_ntop	INIT_DIVISOR_6MHZ
rom400_sock.h, <a href="#">63</a>	rom400_init.h, <a href="#">22</a>
inet_pton	INIT_DIVISOR_7MHZ
rom400_sock.h, <a href="#">63</a>	rom400_init.h, <a href="#">22</a>
init_clearSystemRAM	INIT_DIVISOR_80MHZ
rom400_init.h, <a href="#">23</a>	rom400_init.h, <a href="#">22</a>
init_clearXSEG	INIT_DIVISOR_8MHZ
rom400_init.h, <a href="#">24</a>	rom400_init.h, <a href="#">22</a>
init_copyivt	INIT_DIVISOR_96MHZ
rom400_init.h, <a href="#">24</a>	rom400_init.h, <a href="#">22</a>
INIT_CRYSTALFAIL_RESET	init_enableinterrupts
rom400_init.h, <a href="#">19</a>	rom400_init.h, <a href="#">24</a>
INIT_DIVISOR_10MHZ	init_eth
rom400_init.h, <a href="#">19</a>	rom400_init.h, <a href="#">24</a>
INIT_DIVISOR_112MHZ	init_getbootstate
rom400_init.h, <a href="#">19</a>	rom400_init.h, <a href="#">24</a>
INIT_DIVISOR_128MHZ	init_km
rom400_init.h, <a href="#">19</a>	rom400_init.h, <a href="#">25</a>
INIT_DIVISOR_12MHZ	init_mm
rom400_init.h, <a href="#">19</a>	rom400_init.h, <a href="#">25</a>
INIT_DIVISOR_14MHZ	init_netboot
rom400_init.h, <a href="#">20</a>	rom400_init.h, <a href="#">25</a>
INIT_DIVISOR_16MHZ	init_network
rom400_init.h, <a href="#">20</a>	rom400_init.h, <a href="#">26</a>
INIT_DIVISOR_20MHZ	init_ow
rom400_init.h, <a href="#">20</a>	rom400_init.h, <a href="#">26</a>
INIT_DIVISOR_24MHZ	INIT_POWERFAIL_RESET
rom400_init.h, <a href="#">20</a>	rom400_init.h, <a href="#">23</a>
INIT_DIVISOR_28MHZ	init_redirect
rom400_init.h, <a href="#">20</a>	rom400_init.h, <a href="#">27</a>
INIT_DIVISOR_32MHZ	init_rom
rom400_init.h, <a href="#">20</a>	rom400_init.h, <a href="#">27</a>
INIT_DIVISOR_3MHZ	init_sockets
rom400_init.h, <a href="#">21</a>	rom400_init.h, <a href="#">27</a>
INIT_DIVISOR_40MHZ	init_tick
rom400_init.h, <a href="#">21</a>	rom400_init.h, <a href="#">28</a>
INIT_DIVISOR_48MHZ	init_usekeilmonitor



- rom400\_init.h, [28](#)
- init\_version
  - rom400\_init.h, [28](#)
- INIT\_WATCHDOG\_RESET
  - rom400\_init.h, [23](#)
- IPPROTO\_UDP
  - rom400\_sock.h, [53](#)
- ISR\_CAN0
  - tini400\_isr.h, [148](#)
- ISR\_ETHERNET
  - tini400\_isr.h, [148](#)
- ISR\_ETHERNETPOWER
  - tini400\_isr.h, [148](#)
- ISR\_EXTERNALINT0
  - tini400\_isr.h, [148](#)
- ISR\_EXTERNALINT1
  - tini400\_isr.h, [148](#)
- ISR\_EXTERNALINT2345
  - tini400\_isr.h, [148](#)
- isr\_getinterruptvector
  - tini400\_isr.h, [151](#)
- ISR\_POWERFAIL
  - tini400\_isr.h, [149](#)
- ISR\_SERIAL0
  - tini400\_isr.h, [149](#)
- ISR\_SERIAL1
  - tini400\_isr.h, [149](#)
- ISR\_SERIAL2
  - tini400\_isr.h, [149](#)
- isr\_setinterruptvector
  - tini400\_isr.h, [151](#)
- ISR\_TIMER0
  - tini400\_isr.h, [149](#)
- ISR\_TIMER1
  - tini400\_isr.h, [150](#)
- ISR\_TIMER2
  - tini400\_isr.h, [150](#)
- ISR\_TIMER3
  - tini400\_isr.h, [150](#)
- isr\_version
  - tini400\_isr.h, [152](#)
- ISR\_WATCHDOG
  - tini400\_isr.h, [150](#)
- ISR\_WRITEPROTECT
  - tini400\_isr.h, [150](#)
- join
  - rom400\_sock.h, [53](#)
- kmem\_init
  - rom400\_kmem.h, [30](#)
- kmem\_install
  - rom400\_kmem.h, [31](#)
- kmem\_version
  - rom400\_kmem.h, [32](#)
- L\_tmpnam
  - stdio.h, [118](#)
- leave
  - rom400\_sock.h, [54](#)
- listen
  - rom400\_sock.h, [54](#)
- mailhostent, [5](#)
  - h\_name, [5](#)
  - preference, [5](#)
- MAX\_PRIORITY
  - rom400\_task.h, [81](#)
- mem\_free
  - rom400\_mem.h, [33](#)
- mem\_getfreeram
  - rom400\_mem.h, [33](#)
- mem\_malloc
  - rom400\_mem.h, [34](#)
- mem\_mallocdirty
  - rom400\_mem.h, [34](#)
- mem\_sizeof
  - rom400\_mem.h, [35](#)
- mem\_version
  - rom400\_mem.h, [35](#)
- millis
  - TIME, [8](#)
- MIN\_PRIORITY
  - rom400\_task.h, [81](#)
- mkdir
  - stdio.h, [134](#)
- msb
  - TIME, [8](#)
- Next
  - TCB, [7](#)
- NORM\_PRIORITY

rom400_task.h, <a href="#">81</a>	rom400_sock.h, <a href="#">55</a>
nstoh	recvfrom
rom400_sock.h, <a href="#">55</a>	rom400_sock.h, <a href="#">56</a>
NULL	REDIRECT_0
stdio.h, <a href="#">119</a>	rom400_util.h, <a href="#">103</a>
off_t	REDIRECT_DHCPNOTIFY
stdio.h, <a href="#">120</a>	rom400_util.h, <a href="#">103</a>
ow_byte	REDIRECT_FREE
rom400_ow.h, <a href="#">38</a>	rom400_util.h, <a href="#">103</a>
ow_first	REDIRECT_GETFREERAM
rom400_ow.h, <a href="#">38</a>	rom400_util.h, <a href="#">103</a>
ow_getcurrentid	REDIRECT_GETTASKID
rom400_ow.h, <a href="#">38</a>	rom400_util.h, <a href="#">104</a>
ow_next	REDIRECT_GETTHREADID
rom400_ow.h, <a href="#">38</a>	rom400_util.h, <a href="#">104</a>
ow_reset	REDIRECT_GETTIMEMILLIS
rom400_ow.h, <a href="#">39</a>	rom400_util.h, <a href="#">104</a>
OW_RESET_ALARM	REDIRECT_INFOSENDCHAR
rom400_ow.h, <a href="#">37</a>	rom400_util.h, <a href="#">104</a>
OW_RESET_NO_PRESENCE	REDIRECT_IP_-
rom400_ow.h, <a href="#">37</a>	COMPUTECHECKSUM_-
OW_RESET_PRESENCE	SOFTWARE
rom400_ow.h, <a href="#">37</a>	rom400_util.h, <a href="#">104</a>
OW_RESET_SHORT	REDIRECT_KERNELFREE
rom400_ow.h, <a href="#">37</a>	rom400_util.h, <a href="#">105</a>
ow_version	REDIRECT_KERNELMALLOC
rom400_ow.h, <a href="#">39</a>	rom400_util.h, <a href="#">105</a>
P_tmpdir	REDIRECT_MALLOC
stdio.h, <a href="#">119</a>	rom400_util.h, <a href="#">105</a>
PF_INET	REDIRECT_MALLOCDIRTY
rom400_sock.h, <a href="#">55</a>	rom400_util.h, <a href="#">105</a>
preference	REDIRECT_MM_UNDEREF
mailhostent, <a href="#">5</a>	rom400_util.h, <a href="#">105</a>
printf	REDIRECT_OWIP_READCONFIG
stdio.h, <a href="#">135</a>	rom400_util.h, <a href="#">106</a>
Priority	REDIRECT_ROM_TASK_CREATE
TCB, <a href="#">7</a>	rom400_util.h, <a href="#">106</a>
putc	REDIRECT_ROM_TASK_DESTROY
stdio.h, <a href="#">135</a>	rom400_util.h, <a href="#">106</a>
putchar	REDIRECT_ROM_TASK_DUPLICATE
stdio.h, <a href="#">135</a>	rom400_util.h, <a href="#">106</a>
puts	REDIRECT_ROM_TASK_SWITCH_IN
stdio.h, <a href="#">135</a>	rom400_util.h, <a href="#">106</a>
recv	REDIRECT_ROM_TASK_SWITCH_-
	OUT
	rom400_util.h, <a href="#">107</a>

REDIRECT\_SETMACID  
     rom400\_util.h, 107  
 REDIRECT\_SLEEP  
     rom400\_util.h, 107  
 REDIRECT\_THREADIOSLEEP  
     rom400\_util.h, 107  
 REDIRECT\_THREADIOSLEEPNC  
     rom400\_util.h, 107  
 REDIRECT\_THREADRESTORE  
     rom400\_util.h, 108  
 REDIRECT\_THREADRESUME  
     rom400\_util.h, 108  
 REDIRECT\_THREADSAVE  
     rom400\_util.h, 108  
 REDIRECT\_TINIEXPORT\_MM\_-  
     DEREF  
         rom400\_util.h, 108  
 RELOAD\_14\_746  
     rom400\_task.h, 81  
 RELOAD\_18\_432  
     rom400\_task.h, 81  
 RELOAD\_29\_491  
     rom400\_task.h, 82  
 RELOAD\_36\_864  
     rom400\_task.h, 82  
 RELOAD\_58\_982  
     rom400\_task.h, 82  
 RELOAD\_73\_728  
     rom400\_task.h, 82  
 remove  
     stdio.h, 136  
 rename  
     stdio.h, 136  
 rewind  
     stdio.h, 136  
 ROM400\_-  
     ARRAYINDEXOUTOFBOUNDSEXCEPTION,  
         13  
     ROM400\_BINDEXCEPTION, 13  
     ROM400\_-  
         CONNECTEXCEPTION,  
         13  
     ROM400\_ERR\_VERSION, 13  
     ROM400\_INTERNALERROR, 13  
     ROM400\_-  
         INTERRUPTEDIOEXCEPTION,  
         14  
     ROM400\_IOEXCEPTION, 14  
     ROM400\_-  
         NULLPOINTEREXCEPTION,  
         14  
     ROM400\_-  
         OUTOFMEMORYERROR,  
         14  
     ROM400\_SOCKETEXCEPTION,  
         14  
     ROM400\_FLASH\_VERSION  
         rom400\_err.h, 13  
 rom400\_flash.h, 14  
     flash\_eraseblock, 15  
     flash\_programbyte, 15  
     flash\_version, 16  
     ROM400\_FLASH\_VERSION, 15  
 ROM400\_FLASH\_VERSION  
     rom400\_flash.h, 15  
 rom400\_init.h, 16  
 DHCP\_STATUS\_BOUND, 9  
 DHCP\_STATUS\_INIT, 10  
 DHCP\_STATUS\_INITREBOOT, 10  
 DHCP\_STATUS\_REBINDING, 10  
 DHCP\_STATUS\_REBOOTING, 10  
 DHCP\_STATUS\_RENEWING, 10  
 DHCP\_STATUS\_REQUESTING,  
     10  
 DHCP\_STATUS\_SELECTING, 11  
 dhcp\_stop, 12  
 dhcp\_version, 12  
 ROM400\_DHCP\_VERSION, 11  
 ROM400\_DHCP\_VERSION  
     rom400\_dhcp.h, 11  
 rom400\_err.h, 13  
 ROM400\_-  
     ARRAYINDEXOUTOFBOUNDSEXCEPTION,  
         13  
     ROM400\_BINDEXCEPTION, 13  
     ROM400\_-  
         CONNECTEXCEPTION,  
         13  
     ROM400\_ERR\_VERSION, 13  
     ROM400\_INTERNALERROR, 13  
     ROM400\_-  
         INTERRUPTEDIOEXCEPTION,  
         14  
     ROM400\_IOEXCEPTION, 14  
     ROM400\_-  
         NULLPOINTEREXCEPTION,  
         14  
     ROM400\_-  
         OUTOFMEMORYERROR,  
         14  
     ROM400\_SOCKETEXCEPTION,  
         14  
     ROM400\_FLASH\_VERSION  
         rom400\_err.h, 13  
 rom400\_flash.h, 14  
     flash\_eraseblock, 15  
     flash\_programbyte, 15  
     flash\_version, 16  
     ROM400\_FLASH\_VERSION, 15  
 ROM400\_FLASH\_VERSION  
     rom400\_flash.h, 15  
 rom400\_init.h, 16

- init\_clearSystemRAM, [23](#)
- init\_clearXSEG, [24](#)
- init\_copyivt, [24](#)
- INIT\_CRYSTALFAIL\_RESET, [19](#)
- INIT\_DIVISOR\_10MHZ, [19](#)
- INIT\_DIVISOR\_112MHZ, [19](#)
- INIT\_DIVISOR\_128MHZ, [19](#)
- INIT\_DIVISOR\_12MHZ, [19](#)
- INIT\_DIVISOR\_14MHZ, [20](#)
- INIT\_DIVISOR\_16MHZ, [20](#)
- INIT\_DIVISOR\_20MHZ, [20](#)
- INIT\_DIVISOR\_24MHZ, [20](#)
- INIT\_DIVISOR\_28MHZ, [20](#)
- INIT\_DIVISOR\_32MHZ, [20](#)
- INIT\_DIVISOR\_3MHZ, [21](#)
- INIT\_DIVISOR\_40MHZ, [21](#)
- INIT\_DIVISOR\_48MHZ, [21](#)
- INIT\_DIVISOR\_4MHZ, [21](#)
- INIT\_DIVISOR\_56MHZ, [21](#)
- INIT\_DIVISOR\_5MHZ, [21](#)
- INIT\_DIVISOR\_64MHZ, [22](#)
- INIT\_DIVISOR\_6MHZ, [22](#)
- INIT\_DIVISOR\_7MHZ, [22](#)
- INIT\_DIVISOR\_80MHZ, [22](#)
- INIT\_DIVISOR\_8MHZ, [22](#)
- INIT\_DIVISOR\_96MHZ, [22](#)
- init\_enableinterrupts, [24](#)
- init\_eth, [24](#)
- init\_getbootstate, [24](#)
- init\_km, [25](#)
- init\_mm, [25](#)
- init\_netboot, [25](#)
- init\_network, [26](#)
- init\_ow, [26](#)
- INIT\_POWERFAIL\_RESET, [23](#)
- init\_redirect, [27](#)
- init\_rom, [27](#)
- init\_sockets, [27](#)
- init\_tick, [28](#)
- init\_usekeilmonitor, [28](#)
- init\_version, [28](#)
- INIT\_WATCHDOG\_RESET, [23](#)
- ROM400\_INIT\_VERSION, [23](#)
- USE\_KEIL\_MONITOR, [23](#)
- ROM400\_INIT\_VERSION
  - rom400\_init.h, [23](#)
- ROM400\_INTERNALERROR
  - rom400\_err.h, [13](#)
- ROM400\_-
  - INTERRUPTEDIOEXCEPTION
    - rom400\_err.h, [14](#)
- ROM400\_IOEXCEPTION
  - rom400\_err.h, [14](#)
- rom400\_kmem.h, [29](#)
  - kmem\_init, [30](#)
  - kmem\_install, [31](#)
  - kmem\_version, [32](#)
  - ROM400\_KMEM\_MODEL\_-
    - LARGEST, [30](#)
  - ROM400\_KMEM\_MODEL\_-
    - SMALLEST, [30](#)
  - ROM400\_KMEM\_VERSION, [30](#)
- ROM400\_KMEM\_MODEL\_LARGEST
  - rom400\_kmem.h, [30](#)
- ROM400\_KMEM\_MODEL\_-
  - SMALLEST
    - rom400\_kmem.h, [30](#)
- ROM400\_KMEM\_VERSION
  - rom400\_kmem.h, [30](#)
- rom400\_mem.h, [32](#)
  - mem\_free, [33](#)
  - mem\_getfreeram, [33](#)
  - mem\_malloc, [34](#)
  - mem\_mallocdirty, [34](#)
  - mem\_sizeof, [35](#)
  - mem\_version, [35](#)
  - ROM400\_MEM\_VERSION, [33](#)
- ROM400\_MEM\_VERSION
  - rom400\_mem.h, [33](#)
- ROM400\_NULLPOINTEREXCEPTION
  - rom400\_err.h, [14](#)
- ROM400\_OUTOFMEMORYERROR
  - rom400\_err.h, [14](#)
- rom400\_ow.h, [36](#)
  - ow\_byte, [38](#)
  - ow\_first, [38](#)
  - ow\_getcurrentid, [38](#)
  - ow\_next, [38](#)
  - ow\_reset, [39](#)
  - OW\_RESET\_ALARM, [37](#)
  - OW\_RESET\_NO\_PRESENCE, [37](#)
  - OW\_RESET\_PRESENCE, [37](#)

- OW\_RESET\_SHORT, 37
- ow\_version, 39
- ROM400\_OW\_VERSION, 37
- ROM400\_OW\_VERSION
  - rom400\_ow.h, 37
- ROM400\_SCHED\_VERSION
  - rom400\_task.h, 78
- rom400\_sock.h, 39
  - accept, 46
  - acceptqueue, 62
  - AF\_INET, 46
  - AF\_INET6, 46
  - arp\_cacherequest, 46
  - arp\_generaterequest, 47
  - avail, 47
  - bind, 47
  - cleanup, 48
  - clear\_param\_buffers, 62
  - closesocket, 48
  - connect, 49
  - ETH\_STATUS\_LINK, 49
  - getetherstatus, 49
  - getip6params, 50
  - getmacid, 50
  - getnetworkparams, 50
  - getpeername, 51
  - getsockname, 51
  - getsockopt, 52
  - getttftpsrv, 52
  - htons, 53
  - inet\_addr, 63
  - inet\_ntop, 63
  - inet\_pton, 63
  - IPPROTO\_UDP, 53
  - join, 53
  - leave, 54
  - listen, 54
  - nstoh, 55
  - PF\_INET, 55
  - recv, 55
  - recvfrom, 56
  - ROM400\_SOCK\_SYNCH\_VERSION, 56
  - ROM400\_SOCK\_VERSION, 56
  - send, 57
  - sendto, 57
  - setmacid, 58
  - setnetworkparams, 58
  - setsockopt, 59
  - setttftpsrv, 59
  - SO\_BINDADDR, 60
  - SO\_LINGER, 60
  - SO\_TIMEOUT, 60
  - SOCK\_DGRAM, 60
  - SOCK\_STREAM, 61
  - sock\_version, 61
  - socket, 61
  - SOCKET\_TYPE\_DATAGRAM, 61
  - SOCKET\_TYPE\_STREAM, 62
  - syn\_accept, 64
  - syn\_arp\_cacherequest, 64
  - syn\_arp\_generaterequest, 65
  - syn\_avail, 65
  - syn\_bind, 65
  - syn\_cleanup, 66
  - syn\_closesocket, 66
  - syn\_connect, 67
  - syn\_getetherstatus, 67
  - syn\_getip6params, 67
  - syn\_getmacid, 68
  - syn\_getnetworkparams, 68
  - syn\_getpeername, 69
  - syn\_getsockname, 69
  - syn\_getsockopt, 69
  - syn\_getttftpsrv, 70
  - syn\_join, 70
  - syn\_leave, 71
  - syn\_listen, 71
  - syn\_recv, 72
  - syn\_recvfrom, 72
  - syn\_send, 73
  - syn\_sendto, 73
  - syn\_setDatagramAddress, 74
  - syn\_setmacid, 74
  - syn\_setnetworkparams, 75
  - syn\_setsockopt, 75
  - syn\_setttftpsrv, 76
  - syn\_socket, 76
  - syn\_version, 77
  - TCP\_NODELAY, 62
  - udpavailable, 77
  - ROM400\_SOCK\_SYNCH\_VERSION

- rom400\_sock.h, 56
- ROM400 SOCK\_VERSION
  - rom400\_sock.h, 56
- ROM400\_SOCKETEXCEPTION
  - rom400\_err.h, 14
- rom400\_task.h, 77
  - FLAG\_DHCP\_WAIT, 80
  - FLAG\_IO\_WAIT, 80
  - FLAG\_SLEEPING, 81
  - MAX\_PRIORITY, 81
  - MIN\_PRIORITY, 81
  - NORM\_PRIORITY, 81
  - RELOAD\_14\_746, 81
  - RELOAD\_18\_432, 81
  - RELOAD\_29\_491, 82
  - RELOAD\_36\_864, 82
  - RELOAD\_58\_982, 82
  - RELOAD\_73\_728, 82
  - ROM400\_SCHED\_VERSION, 78
  - ROM400\_TASK\_VERSION, 82
  - ROM\_SAVESIZE, 83
  - task\_entercritsection, 84
  - task\_fork, 84
  - task\_genesis, 85
  - task\_getcurrent, 85
  - task\_getpriority, 85
  - task\_gettaskid, 86
  - task\_getthreadid, 86
  - task\_gettickreload, 87
  - task\_gettimemillis, 87
  - task\_kill, 87
  - task\_leavecritsection, 88
  - task\_setpriority, 88
  - task\_settickreload, 89
  - task\_signal, 89
  - task\_sleep, 83
  - task\_suspend, 90
  - task\_synch\_sleep, 90
  - task\_synch\_wait, 91
  - task\_threadiosleep, 91
  - task\_threadiosleepnc, 92
  - task\_threadrestore, 92
  - task\_threadresume, 93
  - task\_threadsave, 93
  - task\_version, 94
  - task\_wait, 83
- ROM400\_TASK\_VERSION
  - rom400\_task.h, 82
- rom400\_tftp.h, 94
  - ROM400\_TFTP\_VERSION, 95
  - tftp\_first, 96
  - tftp\_getdata, 96
  - tftp\_init, 96
  - TFTP\_LAST\_SEGMENT, 95
  - TFTP\_MORE\_DATA, 95
  - tftp\_next, 97
  - tftp\_version, 97
- ROM400\_TFTP\_VERSION
  - rom400\_tftp.h, 95
- rom400\_useriopoll.h, 97
  - ROM400\_USERIOPOLL\_-  
VERSION, 98
  - useriopoll\_getlistsize, 99
  - useriopoll\_getpollroutine, 99
  - useriopoll\_init, 99
  - useriopoll\_isinstalled, 100
  - useriopoll\_registerpollroutine, 100
  - useriopoll\_removepollroutine, 100
  - useriopoll\_version, 101
- ROM400\_USERIOPOLL\_VERSION
  - rom400\_useriopoll.h, 98
- rom400\_util.h, 101
  - REDIRECT\_0, 103
  - REDIRECT\_DHCPNOTIFY, 103
  - REDIRECT\_FREE, 103
  - REDIRECT\_GETFREERAM, 103
  - REDIRECT\_GETTASKID, 104
  - REDIRECT\_GETTHREADID, 104
  - REDIRECT\_GETTIMEMILLIS,  
104
  - REDIRECT\_INFOSENDCHAR,  
104
  - REDIRECT\_IP\_-  
COMPUTECHECKSUM\_-  
SOFTWARE, 104
  - REDIRECT\_KERNELFREE, 105
  - REDIRECT\_KERNELMALLOC,  
105
  - REDIRECT\_MALLOC, 105
  - REDIRECT\_MALLOCDIRTY, 105
  - REDIRECT\_MM\_UNDEREF, 105

REDIRECT_OWIP_-	s6_addr
READCONFIG, <a href="#">106</a>	in6_addr, <a href="#">4</a>
REDIRECT_ROM_TASK_-	s_addr
CREATE, <a href="#">106</a>	in_addr, <a href="#">5</a>
REDIRECT_ROM_TASK_-	scanf
DESTROY, <a href="#">106</a>	stdio.h, <a href="#">137</a>
REDIRECT_ROM_TASK_-	SEEK_CUR
DUPLICATE, <a href="#">106</a>	stdio.h, <a href="#">119</a>
REDIRECT_ROM_TASK_-	SEEK_END
SWITCH_IN, <a href="#">106</a>	stdio.h, <a href="#">119</a>
REDIRECT_ROM_TASK_-	SEEK_SET
SWITCH_OUT, <a href="#">107</a>	stdio.h, <a href="#">119</a>
REDIRECT_SETMACID, <a href="#">107</a>	send
REDIRECT_SLEEP, <a href="#">107</a>	rom400_sock.h, <a href="#">57</a>
REDIRECT_THREADIOSLEEP,	sendto
<a href="#">107</a>	rom400_sock.h, <a href="#">57</a>
REDIRECT_-	setmacid
THREADIOSLEEPNC, <a href="#">107</a>	rom400_sock.h, <a href="#">58</a>
REDIRECT_THREADRESTORE,	setnetworkparams
<a href="#">108</a>	rom400_sock.h, <a href="#">58</a>
REDIRECT_THREADRESUME,	setsockopt
<a href="#">108</a>	rom400_sock.h, <a href="#">59</a>
REDIRECT_THREADSAVE, <a href="#">108</a>	setttftpsrv
REDIRECT_TINIEXPORT_MM_-	rom400_sock.h, <a href="#">59</a>
DEREF, <a href="#">108</a>	sin_addr
ROM400_UTIL_VERSION, <a href="#">108</a>	sockaddr, <a href="#">6</a>
util_crc16, <a href="#">109</a>	sockaddr_in, <a href="#">6</a>
util_getpseudorandom, <a href="#">109</a>	sin_family
util_infosendchar, <a href="#">109</a>	sockaddr, <a href="#">6</a>
util_installhook, <a href="#">109</a>	sockaddr_in, <a href="#">7</a>
util_memclear, <a href="#">110</a>	sin_port
util_memcompare, <a href="#">110</a>	sockaddr, <a href="#">6</a>
util_memcopy, <a href="#">111</a>	sockaddr_in, <a href="#">6</a>
util_setrandomseed, <a href="#">111</a>	sin_zero
util_version, <a href="#">111</a>	sockaddr_in, <a href="#">6</a>
ROM400_UTIL_VERSION	size_t
rom400_util.h, <a href="#">108</a>	stdio.h, <a href="#">120</a>
rom400_xnetstack.h, <a href="#">112</a>	SO_BINDADDR
ROM400_XNETSTACK_-	rom400_sock.h, <a href="#">60</a>
VERSION, <a href="#">112</a>	SO_LINGER
xnetstack_install, <a href="#">112</a>	rom400_sock.h, <a href="#">60</a>
xnetstack_version, <a href="#">112</a>	SO_TIMEOUT
ROM400_XNETSTACK_VERSION	rom400_sock.h, <a href="#">60</a>
rom400_xnetstack.h, <a href="#">112</a>	SOCK_DGRAM
ROM_SAVE_SIZE	rom400_sock.h, <a href="#">60</a>
rom400_task.h, <a href="#">83</a>	SOCK_STREAM

- rom400\_sock.h, [61](#)
- sock\_version
  - rom400\_sock.h, [61](#)
- sockaddr, [5](#)
  - bogus\_ptr, [6](#)
  - sin\_addr, [6](#)
  - sin\_family, [6](#)
  - sin\_port, [6](#)
- sockaddr\_in, [6](#)
  - bogus\_ptr, [6](#)
  - sin\_addr, [6](#)
  - sin\_family, [7](#)
  - sin\_port, [6](#)
  - sin\_zero, [6](#)
- socket
  - rom400\_sock.h, [61](#)
- SOCKET\_TYPE\_DATAGRAM
  - rom400\_sock.h, [61](#)
- SOCKET\_TYPE\_STREAM
  - rom400\_sock.h, [62](#)
- sprintf
  - stdio.h, [137](#)
- sscanf
  - stdio.h, [137](#)
- StatePtr
  - TCB, [7](#)
- StateSize
  - TCB, [7](#)
- stdio.h, [113](#)
  - \_getkey, [120](#)
  - clearerr, [120](#)
  - EOF, [117](#)
  - fclose, [121](#)
  - feof, [121](#)
  - ferror, [121](#)
  - fexists, [122](#)
  - fflush, [122](#)
  - fgetc, [122](#)
  - fgetpos, [123](#)
  - fgets, [123](#)
  - FILE, [120](#)
  - FILE\_FLAGS\_EOF, [117](#)
  - FILE\_FLAGS\_TEMP, [118](#)
  - FILE\_TYPE\_TINIFS, [118](#)
  - FILENAME\_MAX, [118](#)
  - filesystem\_version, [124](#)
  - finit, [124](#)
  - flockfile, [125](#)
  - fopen, [125](#)
  - fopen\_fd, [126](#)
  - FOPEN\_MAX, [118](#)
  - fpos\_t, [120](#)
  - fputc, [126](#)
  - fputs, [127](#)
  - fread, [127](#)
  - freadbytes, [128](#)
  - freopen, [128](#)
  - FS\_VERSION, [118](#)
  - fseek, [129](#)
  - fseeko, [129](#)
  - fsetpos, [130](#)
  - ftell, [130](#)
  - ftello, [131](#)
  - ftrylockfile, [131](#)
  - funlockfile, [132](#)
  - fwrite, [132](#)
  - fwritebytes, [133](#)
  - getc, [133](#)
  - getchar, [134](#)
  - getfreefsram, [134](#)
  - gets, [134](#)
  - L\_tmpnam, [118](#)
  - mkdir, [134](#)
  - NULL, [119](#)
  - off\_t, [120](#)
  - P\_tmpdir, [119](#)
  - printf, [135](#)
  - putc, [135](#)
  - putchar, [135](#)
  - puts, [135](#)
  - remove, [136](#)
  - rename, [136](#)
  - rewind, [136](#)
  - scanf, [137](#)
  - SEEK\_CUR, [119](#)
  - SEEK\_END, [119](#)
  - SEEK\_SET, [119](#)
  - size\_t, [120](#)
  - sprintf, [137](#)
  - sscanf, [137](#)
  - tempnam, [137](#)
  - TMP\_MAX, [120](#)



- tmpfile, 138
- tmpnam, 138
- ungetchar, 139
- vprintf, 139
- vsprintf, 139
- syn\_accept
  - rom400\_sock.h, 64
- syn\_arp\_cacherequest
  - rom400\_sock.h, 64
- syn\_arp\_generaterequest
  - rom400\_sock.h, 65
- syn\_avail
  - rom400\_sock.h, 65
- syn\_bind
  - rom400\_sock.h, 65
- syn\_cleanup
  - rom400\_sock.h, 66
- syn\_closesocket
  - rom400\_sock.h, 66
- syn\_connect
  - rom400\_sock.h, 67
- syn\_getetherstatus
  - rom400\_sock.h, 67
- syn\_getip6params
  - rom400\_sock.h, 67
- syn\_getmacid
  - rom400\_sock.h, 68
- syn\_getnetworkparams
  - rom400\_sock.h, 68
- syn\_getpeername
  - rom400\_sock.h, 69
- syn\_getsockname
  - rom400\_sock.h, 69
- syn\_getsockopt
  - rom400\_sock.h, 69
- syn\_gettftpserver
  - rom400\_sock.h, 70
- syn\_join
  - rom400\_sock.h, 70
- syn\_leave
  - rom400\_sock.h, 71
- syn\_listen
  - rom400\_sock.h, 71
- syn\_rcv
  - rom400\_sock.h, 72
- syn\_rcvfrom
  - rom400\_sock.h, 72
- syn\_send
  - rom400\_sock.h, 73
- syn\_sendto
  - rom400\_sock.h, 73
- syn\_setDatagramAddress
  - rom400\_sock.h, 74
- syn\_setmacid
  - rom400\_sock.h, 74
- syn\_setnetworkparams
  - rom400\_sock.h, 75
- syn\_setsockopt
  - rom400\_sock.h, 75
- syn\_settftpserver
  - rom400\_sock.h, 76
- syn\_socket
  - rom400\_sock.h, 76
- syn\_version
  - rom400\_sock.h, 77
- task\_entercritsection
  - rom400\_task.h, 84
- task\_fork
  - rom400\_task.h, 84
- task\_genesis
  - rom400\_task.h, 85
- task\_getcurrent
  - rom400\_task.h, 85
- task\_getpriority
  - rom400\_task.h, 85
- task\_gettaskid
  - rom400\_task.h, 86
- task\_getthreadid
  - rom400\_task.h, 86
- task\_gettickreload
  - rom400\_task.h, 87
- task\_gettimemillis
  - rom400\_task.h, 87
- task\_kill
  - rom400\_task.h, 87
- task\_leavecritsection
  - rom400\_task.h, 88
- task\_setpriority
  - rom400\_task.h, 88
- task\_settickreload
  - rom400\_task.h, 89

- task\_signal
  - rom400\_task.h, [89](#)
- task\_sleep
  - rom400\_task.h, [83](#)
- task\_suspend
  - rom400\_task.h, [90](#)
- task\_synch\_sleep
  - rom400\_task.h, [90](#)
- task\_synch\_wait
  - rom400\_task.h, [91](#)
- task\_threadiosleep
  - rom400\_task.h, [91](#)
- task\_threadiosleepnc
  - rom400\_task.h, [92](#)
- task\_threadrestore
  - rom400\_task.h, [92](#)
- task\_threadresume
  - rom400\_task.h, [93](#)
- task\_threadsave
  - rom400\_task.h, [93](#)
- task\_version
  - rom400\_task.h, [94](#)
- task\_wait
  - rom400\_task.h, [83](#)
- TCB, [7](#)
  - Flags, [7](#)
  - ID, [7](#)
  - Next, [7](#)
  - Priority, [7](#)
  - StatePtr, [7](#)
  - StateSize, [7](#)
  - WakeupTime, [7](#)
- TCP\_NODELAY
  - rom400\_sock.h, [62](#)
- tempnam
  - stdio.h, [137](#)
- tftp\_first
  - rom400\_tftp.h, [96](#)
- tftp\_getdata
  - rom400\_tftp.h, [96](#)
- tftp\_init
  - rom400\_tftp.h, [96](#)
- TFTP\_LAST\_SEGMENT
  - rom400\_tftp.h, [95](#)
- TFTP\_MORE\_DATA
  - rom400\_tftp.h, [95](#)
- tftp\_next
  - rom400\_tftp.h, [97](#)
- tftp\_version
  - rom400\_tftp.h, [97](#)
- TIME, [8](#)
  - millis, [8](#)
  - msb, [8](#)
- tini400\_crypt.h, [139](#)
  - crypt\_sha1, [140](#)
  - crypt\_version, [140](#)
  - TINI400\_CRYPT\_VERSION, [140](#)
- TINI400\_CRYPT\_VERSION
  - tini400\_crypt.h, [140](#)
- tini400\_dns.h, [140](#)
  - dns\_enableipv6queries, [142](#)
  - dns\_getmx, [142](#)
  - dns\_getprimary, [143](#)
  - dns\_getsecondary, [143](#)
  - dns\_gettimeout, [144](#)
  - dns\_init, [144](#)
  - dns\_setprimary, [144](#)
  - dns\_setsecondary, [144](#)
  - dns\_settimeout, [145](#)
  - dns\_version, [145](#)
  - gethostbyaddr, [145](#)
  - gethostbyname, [146](#)
  - TINI400\_DNS\_VERSION, [142](#)
- TINI400\_DNS\_VERSION
  - tini400\_dns.h, [142](#)
- tini400\_isr.h, [146](#)
  - ISR\_CAN0, [148](#)
  - ISR\_ETHERNET, [148](#)
  - ISR\_ETHERNETPOWER, [148](#)
  - ISR\_EXTERNALINT0, [148](#)
  - ISR\_EXTERNALINT1, [148](#)
  - ISR\_EXTERNALINT2345, [148](#)
  - isr\_getinterruptvector, [151](#)
  - ISR\_POWERFAIL, [149](#)
  - ISR\_SERIAL0, [149](#)
  - ISR\_SERIAL1, [149](#)
  - ISR\_SERIAL2, [149](#)
  - isr\_setinterruptvector, [151](#)
  - ISR\_TIMER0, [149](#)
  - ISR\_TIMER1, [150](#)
  - ISR\_TIMER2, [150](#)
  - ISR\_TIMER3, [150](#)

- isr\_version, [152](#)
- ISR\_WATCHDOG, [150](#)
- ISR\_WRITEPROTECT, [150](#)
- TINI400\_ISR\_VERSION, [151](#)
- TINI400\_ISR\_VERSION
  - tini400\_isr.h, [151](#)
- tini\_i2c.h, [152](#)
  - i2c\_bit, [155](#)
  - i2c\_delay, [155](#)
  - I2C\_DELAY\_LOOP\_COUNT, [154](#)
  - I2C\_ENABLE\_SCL\_WAIT\_FOR\_-  
SLOW\_SLAVES, [154](#)
  - I2C\_MAXIMUM\_SCL\_-  
WAITCOUNT, [154](#)
  - i2c\_readbit, [155](#)
  - i2c\_readblock, [155](#)
  - i2c\_readbyte, [155](#)
  - I2C\_SCL, [154](#)
  - I2C\_SDA, [154](#)
  - i2c\_select, [155](#)
  - i2c\_start, [156](#)
  - i2c\_stop, [156](#)
  - i2c\_version, [156](#)
  - i2c\_writeblock, [156](#)
  - i2c\_writebyte, [156](#)
  - i2c\_writereadblock, [156](#)
  - TINI\_I2C\_VERSION, [154](#)
- TINI\_I2C\_VERSION
  - tini\_i2c.h, [154](#)
- TMP\_MAX
  - stdio.h, [120](#)
- tmpfile
  - stdio.h, [138](#)
- tmpnam
  - stdio.h, [138](#)
- type
  - file\_structure, [3](#)
- udpavailable
  - rom400\_sock.h, [77](#)
- ungetchar
  - stdio.h, [139](#)
- USE\_KEIL\_MONITOR
  - rom400\_init.h, [23](#)
- useriopoll\_getlistsize
  - rom400\_useriopoll.h, [99](#)
- useriopoll\_getpollroutine
  - rom400\_useriopoll.h, [99](#)
- useriopoll\_init
  - rom400\_useriopoll.h, [99](#)
- useriopoll\_isinstalled
  - rom400\_useriopoll.h, [100](#)
- useriopoll\_registerpollroutine
  - rom400\_useriopoll.h, [100](#)
- useriopoll\_removepollroutine
  - rom400\_useriopoll.h, [100](#)
- useriopoll\_version
  - rom400\_useriopoll.h, [101](#)
- util\_crc16
  - rom400\_util.h, [109](#)
- util\_getpseudorandom
  - rom400\_util.h, [109](#)
- util\_infosendchar
  - rom400\_util.h, [109](#)
- util\_installhook
  - rom400\_util.h, [109](#)
- util\_memclear
  - rom400\_util.h, [110](#)
- util\_memcompare
  - rom400\_util.h, [110](#)
- util\_memcpy
  - rom400\_util.h, [111](#)
- util\_setrandomseed
  - rom400\_util.h, [111](#)
- util\_version
  - rom400\_util.h, [111](#)
- vprintf
  - stdio.h, [139](#)
- vsprintf
  - stdio.h, [139](#)
- WakeupTime
  - TCB, [7](#)
- xnetstack\_install
  - rom400\_xnetstack.h, [112](#)
- xnetstack\_version
  - rom400\_xnetstack.h, [112](#)