



Detailed Description

ASIST provides application developers with C-language functions to build commands and parse reports. Structures are provided for each message in individual include files. Host applications use these structures to set up the data as required and ASIST creates the encoded binary command message to send to the VCO/4K. In reverse, the report messages from the host are unpacked by the ASIST parsing functions.

The V5.x system software includes support for two host API modes: standard and extended. The extended API mode supports the expanding capabilities offered in VCO/4K and V5.x system software, including extended port addresses and the addition of tone plans. The standard host API mode provides backward compatibility to previous system software releases.

ASIST supports both standard and extended API modes. An ASIST function allows you to specify which form of the messages should be used.



Note

Cisco Systems recommends using the extended mode host API for all new application development.

API Message Format

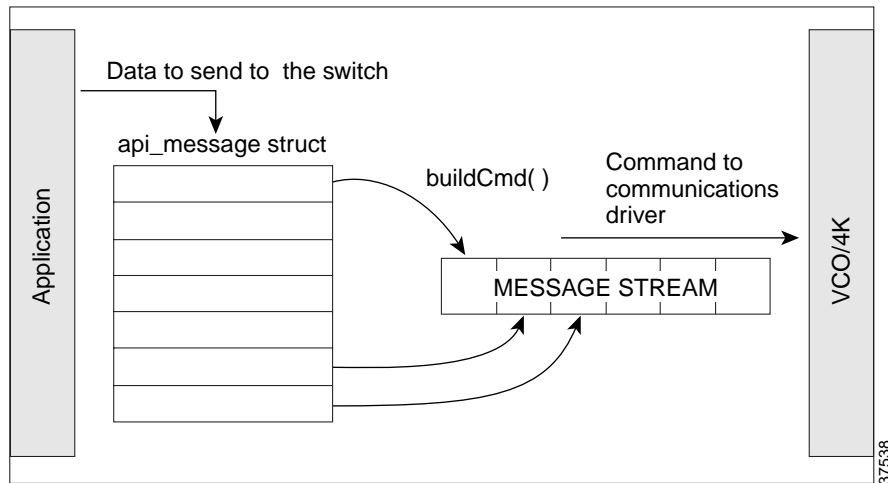
A data structure called **api_message** (defined in the **api_msg.h** file) serves as the interface to all of the command and report functions in ASIST. The **api_message** structure follows the guidelines in both the *Cisco VCO/4K Standard Programming Reference* and the *Cisco VCO/4K Extended API Programming Reference*. The **api_message** structure implements the command/report data field via a union of all the primary command and report data structures defined in the include files for each message type. This union, called **api_cmd_rep**, with the header data in **api_message**, is defined in **api_msg.h**.

Each command buildup function does the following:

- Refers to the appropriate structure pointed to by **msg**.
- Extracts the various data elements.
- Copies them into the correct **buf** location.
- Returns the length of the **buf** to the caller.

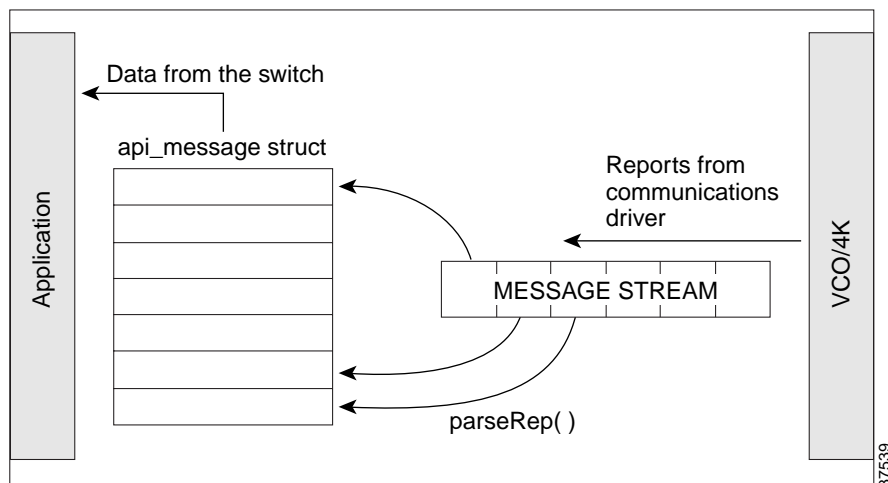
At this point, the **buf** array is ready to be sent to a communication driver and transmitted to the system. Figure 2-1 shows the command build up capability.

Figure 2-1 Command Buildup Flow



The report parsing functions basically perform the inverse of the command buildup functions. They take a byte array filled with a report just received by the communications driver, parse the array byte-by-byte, and initialize the data elements in the appropriate report structure. Figure 2-2 shows the report parsing capability.

Figure 2-2 Report Parsing Flow



Standard versus Extended Operational Mode Host API

The V5.x system software supports 4096 ports and multiple tone plans. An extended mode host application programming interface (API) accommodates these capabilities. The extended mode host API is a superset of the API provided in systems prior to V5.x—the standard mode host API. The same fields, in the same order, are provided. Differences between the two modes include the following:

- Larger field sizes for specific data types.
- Bit flags realigned.

- Additional fields to support new capabilities (i.e., tone plans).

ASIST supports several variations of systems, which include:

- V3.x systems (refer to the *Cisco VCO/4K Standard Programming Reference*)
- V4.x systems (refer to the *Cisco VCO/4K Standard Programming Reference*)
- V5.x (and beyond) systems operating with the standard mode host API
- V5.x (and beyond) systems operating with the extended mode host API

The operational mode which dictates the API, either standard, or extended, is set in the VCO/4K switch at install time for V5.x systems. The API cannot be dynamically changed during operation. Systems using software versions prior to V5.0 support the standard mode API only.



Note

The host API mode used by the application and ASIST **must** match the mode set in the switch.

You set ASIST at run time to support either API by invoking the SetMessageMode() function described in the “Set Message Mode” section on page 2-5. ASIST builds or parses messages according to the mode you specify. This allows the same host application to support both forms of the host API. All function and data definition names are the same.

The data structures within previous releases of ASIST must be expanded to accommodate the extended mode host API. This includes additional fields and larger field sizes. ASIST provides both versions of the structures in order to remain backward compatible. The smaller structures (V4.x and earlier) are referred to as V4.x structures and the updated expanded structures (V5.x and beyond) are referred to as V5.x structures.

V5.x structures must be used when the extended mode API is used. You can use either structure set with the standard mode API. If the V5.x structures are used with the standard mode API, unused fields, such as tone plan, are ignored, and the larger fields sizes are masked in the message being built. Initialize any unused fields to 0.

Instead of providing two physically separate sets of files to handle the two structure sets, the files are combined. Conditional compiling specifies which structure set to use. Set constants to STANDARD to specify the V4.x structures. Set constants to EXTENDED to specify the V5.x structures. These constants are set up on a per message basis. This allows you to migrate from the V4.x structures to the V5.x structures one message at a time. The constants may also be set as a group by setting DEFAULT_API_MODE to the desired mode (see Table 2-1). The constants are provided in asist.h.

Table 2-1 ASIST Structures According to Standard or Extended API

Structures in ASIST as Set by DEFAULT_API_MODE	VCO/4K Set to Standard API at Install Set Message Mode (STANDARD)	VCO/4K Set to Extended API at Install Set Message Mode (EXTENDED)
Standard (V4.x structure).	OK. Provided for backward compatibility.	Not allowed. Extended (V5.x) structures must be used for extended API.
Extended (V5.x structure).	OK. Provided for backward compatibility.	Recommended setting.

Figure 2-3 shows an example of a structure definition set up with the conditional compile information. The figure shows a portion of api_dcc.h set to use the V5.x structure set.

Figure 2-3 Example of an ASIST Structure

```

typedef struct
{
    unsigned inp_level           :4; /* Input level to be adjusted */
    unsigned align1             :2; /* For alignment purpose only */
    unsigned is_olevel_dec      :2; /* Is output level of conference port to
                                     be decremented by 3 dB? */
    unsigned is_voice_1way      :1; /* Is a 2-way or 1-way voice path to be
                                     setup? */
    unsigned is_prt_idle        :1; /* Is line/trunk to be idled through PSC
                                     when deleted from conference
                                     */
    unsigned is_voice_immd      :1; /* Is voice path of outgoing path to be
                                     established immediately on
                                     receipt of command? */

#if CR_DCC_CTRL == EXTENDED
    unsigned align2             :5; /* For alignment purpose only */
    Paddr dcc_port;             /* Address of conference port */
#else
    unsigned align2             :2; /* For alignment purpose only */
    unsigned dcc_port           :11; /* Address of conference port */
#endif
} PortCtrl_t;

typedef struct
{
#if CR_DCC_CTRL == EXTENDED
    SpacerByte spacer_byte;     /* User defined command identifier */
    TonePlan tone_plan;         /* Tone plans */
    ushort confr_no;            /* Conference number */
#else
    uchar confr_no;             /* Conference number */
#endif
    unsigned is_resrve_conf      :1; /* Is conference to be reserved? */
    unsigned is_start_conf      :1; /* Is conference to be started? */
    unsigned is_tear_conf       :1; /* Is conference to be torn down? */
    unsigned is_add_conf        :1; /* Are one or more line/trunk ports to be
                                     added to a conference? */
    unsigned is_del_conf        :1; /* Are one or more line/trunk ports to be
                                     deleted from a conference? */
    unsigned is_level_ad        :1; /* Is the input/output level adjustment
                                     necessary? */
    unsigned align1             :2; /* For alignment purpose only */
    uchar port_count;           /* Number of ports affected */
    PortCtrl_t port_ctrl[MAX_DCC_PORTS]; /* Port Control Codes for ports involved
                                     in conferencing */
}cr_dcc_ctrl;

```

37540

Unique Field Data Types in V4.x and V5.x

To provide flexibility, consistency, and future growth, unique data types are available for fields which differ between the V4.x and V5.x structure sets. They are specified in types.h and are as follows:

- **Paddr**—All extended mode port addresses are declared Paddr data types. Paddr is an unsigned long integer. Do not change the definition of the Paddr type or the port address fields.
- **SpacerByte**—All spacer_byte fields are declared SpacerByte data types. SpacerByte is an unsigned long integer. Do not change the definition of the SpacerByte type or the spacer_byte field.

- **TonePlan**—All `tone_plan` fields are declared `TonePlan` data types. `TonePlan` is an unsigned long integer. Do not change the definition of the `TonePlan` type or the `tone_plan` fields.
- **Rule**—All extended mode impulse/outputpulse rule numbers are declared `Rule` data types. `Rule` is an unsigned short integer. Do not change the definition of the `Rule` type or the impulse/outputpulse rule number fields.
- **Group**—Some extended mode resource group number fields are declared `Group` data types—in reports that have a resource group field, such as the `$DA` and `$DB` reports. `Group` is an unsigned short integer. Do not change the definition of the `Group` type or the resource group number fields.

External Functions

This section specifies the functions called by the application. The functions in turn access the internal message processing described in the “Message Processing” section on page 2-11.

For the following externals functions, the include files are:

- `#include <sys/types.h>`
- `#include “types.h”`
- `#include “api_msg.h”`

Set Message Mode

Function

`int SetMessageMode (int mode).`

The message mode determines whether the library will build and parse standard messages or extended messages. The `setMessageMode` function may be called to adjust the message mode. The parameter may be `STANDARD` in all systems, or `EXTENDED` in VCO V5.0 systems only.

The message mode defaults to `STANDARD`. The `setMessageMode` function returns the selected message mode when it is successful, and a `- 1` when it is unsuccessful.

Parse Report

Function

`int parseRep(unchar *buf, int rep_len, api_message *msg)`

This function is a single function call interface to all the report parsing functions. `parseRep()` uses `msg->func_id` to select the appropriate report parsing function. It then calls the report parsing function and returns the report in the `api_message` structure pointed to by `msg`. If `msg->func_id` is invalid, the function returns a value greater than zero; if invalid, a `- 1`.

Retrieve Incoming Port

Function

- `ushort getIport(api_message *msg)`
- `Paddr getExtIport(api_message *msg)`

This function retrieves the `ctrl_port` variable of the report structure identified by `msg->func_id` and returns it to the caller; if the `func_id` is not valid, it returns a `-1`. This function only accesses the following report structures:

- `rr_mf_digits`
- `rs_port_stats`
- `rr_oport_cos`
- `rr_iport_cos`
- `rs_res_limit`
- `rc_dvc_status`
- `rr_isdn_pcos`
- `rr_isdn_irule`
- `rr_ipulse_rule`
- `rr_psc`
- `rs_port_status`
- `rr_spoken_dig`
- `rr_dtmf_digits`

Retrieve Outgoing Port

Function

- `ushort getOport(api_message *msg)`
- `Paddr getExtOport(api_message *msg)`

This function retrieves the resource or outgoing port of the report structure identified by `msg->func_id` and returns it to the caller; if the `func_id` is not valid, it returns a `-1`. This function only accesses the following report structures:

- `rr_mf_digits`
- `rr_dtmf_digits`
- `rr_spoken_dig`
- `rc_dvc_status`
- `rr_isdn_pcos`
- `rr_oport_cos`

Build Command

Function

```
int buildCmd(api_message *msg, unchar *buf)
```

This function is a single function call interface to all the command buildup functions. `buildCmd()` uses `msg->func_id` to select the appropriate command buildup function. It then calls the command buildup function and returns the command in the `buf` array and the length, in bytes, of the array. If `msg->func_id` is invalid, the function returns a value greater than zero; if invalid, a `-1`.

Substitute Incoming Port

Function

- `int subsIport(api_message *msg, ushort iport)`
- `int subsExtIport(api_message *msg, Paddr iport)`

This function initializes the `ctrl_port` or `iport` variable of the command identified by `msg->func_id`, with the value of the `iport` parameter. The function returns `TRUE` if the `func_id` is valid, otherwise `FALSE`. This function only affects the following command structures:

- `cd_path_ctrl`
- `cr_isdn_ctrl`
- `cr_mf_ctrl`
- `cr_oport_ctrl`
- `cr_iport_ctrl`
- `cr_dtmf_ctrl`
- `cr_dvc_ctrl`
- `cr_src_ctrl`
- `cd_psupv_ctrl`
- `cr_ch_iport`

Substitute Outgoing Port

Function

- `int subsOport(api_message *msg, ushort oport)`
- `int subsExtOport(api_message *msg, Paddr oport)`

This function initializes the outgoing port variable of the command identified by `msg->func_id`, with the value of the `oport` parameter. The function returns `TRUE` if the `func_id` is valid, otherwise `FALSE`. This function only affects the following command structures:

- `cd_path_ctrl`
- `cr_oport_ctrl`
- `cr_ch_iport`, `cr_isdn_ctrl`

Parse Command

Function

```
int parseCmd (uchar *buf, int buflen, api_message *msg)
```

This function processes a command (pointed to by `buf` and returned by the system) and populates the appropriate command data structure in `msg`. It returns a value greater than zero if successful; if an invalid command is passed to it, it returns `-1`.

Processing ISDN IEs: Calling/Called Party Number

The functions in this section can be called directly to parse or build up ISDN message information elements (IEs). Refer to the “Controlling ISDN Primary Rate Interfaces” section on page 2-52 for ISDN message processing.

- `#include "isdn_ie.h"`

- `#include <stdio.h>`

Find Information Element

This function performs the following actions:

- Searches for the IE identifier, **ieid**, in the received IE message segment pointed to by **iebuf**.
- Returns the index into **iebuf** where **ieid** is found; else it returns `-1` if it fails to find **ieid**.

Function

```
findIE(unchar ieid, unchar *iebuf, int seg_count)
```

Mode Constant

None

Parameters

The parameters for this message are in Table 2-2.

Table 2-2 Find Information Element Parameters

Parameter	Type	Description
ieid	unsigned char	IE identifier to search for in iebuf
iebuf	pointer, unsigned char	Contains the received IE message from the system
seg_count	integer	The number of IE segments contained in iebuf

Calling/Called Party Number IE Buildup

This function performs the following actions:

- Translates an API IE message (IE_MSG) into a Calling Party Number or Called Party Number information element, depending upon the **ieid**.
- Allows the user to specify the number type and plan to be used with the address digits.
- If the Calling Party Number IE is specified, the user has access to the number presentation and screening parameters. Set the **ps_ind_flag** to TRUE in this case.
- Allows the user to specify the number of digits and the address digits.
- Returns the length, in bytes, of the IE message contained in **iebuf**.

Function

```
IE_buildCallNum(IE_MSG *iemsgp, unchar *iebuf)
```

Mode Constant

None

Parameters

The parameters for this message are in Table 2-3.

Table 2-3 Calling/Called Party Number IE Buildup Parameters

Parameter	Type	Description
iemsgp->ieid	unsigned char	IE identifier.
type	unsigned char	Number type.
plan	unsigned char	Number plan.
ps_ind_flag	unsigned char	Indicates if presentation and screening are required.
presentation_ind	unsigned char	Is calling party number presented to called user?
screening_ind	unsigned char	Is the calling party number screened?
digit_count	unsigned char	Number of digits.
digits	character	IA5 (ASCII) formatted digits.
iebuf	pointer, unsigned char	Contains the Calling/Called Party IE message.

Calling/Called Party Number IE Parsing

This function performs the following actions:

- Translates a Calling Party Number or Called Party Number information element into API IE message (IE_MSG).
- Provides the user with the number type and plan of the address digits.
- If the Calling Party Number IE is received, the user may be provided the number presentation and screening parameters. The **ps_ind_flag** is set to TRUE in this case.
- Provides the user with the number of digits and the address digits.

Function

IE_parseCallNum(unchar *iebuf, int ielen, IE_MSG *iemsgp)

Mode Constant

None

Parameters

The parameters for this message are in Table 2-4.

Table 2-4 Calling/Called Party Number IE Parameters

Parameters	Type	Description
iebuf	pointer, unsigned char	Contains the Calling/Called Party IE.
ielen	integer	Length, in bytes, of message.
iemsgp->ieid	unsigned char	IE identifier.
type	unsigned char	Number type.

Table 2-4 Calling/Called Party Number IE Parameters (continued)

Parameters	Type	Description
plan	unsigned char	Number plan.
ps_ind_flag	unsigned char	Indicates presentation and screening are set.
presentation_ind	unsigned char	Is calling party number presented to called user?
screening_ind	unsigned char	Is the calling party number screened?
digit_count	unsigned char	Number of digits.
digits	character	IA5 (ASCII) formatted digits.

Network Facility Information Element

This function translates Network Facility Information Elements into API IE message (IE_MSG). No parsing function is available at this time.

Function

IE_buildNetFacil(IE_MSG *iemsgp, unchar *iebuf)

Mode Constant

None

Parameters

The parameters for this message are in Table 2-5.

Table 2-5 Network Facility Information Element Parameters

Parameter	Type	Description
ie_id	unchar	The ID number of the IE, network-specific facility messages use a value of 0x20.
ie_len	unchar	Length of the information element beginning with octet number 3. The maximum length is 25 octets, per TR 41449.
netfacil.length	unchar	The length of network identification in octets, which include octet 3.1 and optional octet(s) 3.2.
netfacil.ext	unchar	Extention.
netfacil.type	unchar	Type of network identification.
netfacil.plan	unchar	Network identification plan.
netfacil.ident[]	unchar	Network identification.
netfacil.par_bin	unchar	Parameters/binary.
netfacil.expan	unchar	Expansion.
netfacil.ftr_svc	unchar	Requested facility is feature/service.
netfacil.codeval	unchar	Facility coding value.
netfacil.param[]	unchar	Parameters.

Message Processing

This section defines the supported command and report messages. The messages are grouped by functional areas.

For each functional area, the include files which describe the associated structures are specified for reference. These include files are included in `api_msg.h` and as a result, they do not need to be called out individually.

For each message, the following data is included:

- Description
- Prototype for the function supporting this message
- The mode constant used in the conditional compile of the structure
- Parameter definitions

For parameters that are different between standard and extended structure definitions, the type specified in the table is as described in the “Standard versus Extended Operational Mode Host API” section on page 2-2. The actual size varies.

For example, the parameter definitions in all cases of port addresses are specified as type `Paddr`. If you compile the structure in standard mode, the type is actually either an unsigned short or an 11-bit field.

For all of the following externals functions, the include files are:

- `#include <sys/types.h>`
- `#include “types.h”`
- `#include “api_msg.h”`

Internal Functions

Individual internal functions perform the actual parsing and building of messages. These functions are not called directly.

The naming conventions adhered to by ASIST functions are as follows:

`fXY_funcname`:

If `X` is:

- `r`—Decoding a report from the system.
- `c`—Sending a command to the system.

And `Y` is:

- `r`—Resource control command/report.
- `c`—Configuration control command/report.
- `s`—System status command/report.
- `d`—System diagnostics command.
- `m`—System maintenance command.

Similarly, the naming conventions for data structures are as follows:

`XY_structname`—where `X` and `Y` have the same meaning as defined above.

Structname and funcname are identical for the data structure and function that they identify. Table 2-6 maps the commands and reports to their corresponding abbreviations.

Table 2-6 Commands/Reports and Function Names

Hex	Command/Report	Function/Structure Name for Command Buildup	Function/Structure Name for Report Parsing
\$65	Subrate Path Control	(f)cr_subrate_ctrl	(f)rr_subrate_ctrl
\$67	DTMF Collection Control	(f)cr_dtmf_ctrl	(f)rr_dtmf_ctrl
\$D1	DTMF Digit Collection	—	(f)rr_dtmf_digits
\$68	MF Collection Control	(f)cr_mf_ctrl	(f)rr_mf_ctrl
\$DO	MF Digit Collection	—	(f)rr_mf_digits
\$6C	DVC Port Control	(f)cr_dvc_ctrl	(f)rr_dvc_ctrl
\$DE	DVC Port Status	—	(f)rc_dvc_status
\$91	Voice Prompt Maintenance	(f)cr_vpm_ctrl	(f)rr_vpm_ctrl
\$6D	Conference Control	(f)cr_dcc_ctrl	(f)rr_dcc_ctrl
\$70	Port Hook State Control	(f)cd_hook_ctrl	(f)rd_hook_ctrl
\$69	Outgoing Port Control	(f)cr_oport_ctrl	(f)rr_oport_ctrl
\$DA	Outgoing Port Change of State	—	(f)rr_oport_cos
\$6A	Incoming Port Control (Macro)	(f)cr_iport_ctrl	(f)_iport_ctrl
\$DB	Incoming Port Change of State	—	(f)rr_iport_cos
\$DD	Inpulse Rule Complete (Macro)	—	(f)rr_impulse_rule
\$72	Port Supervision Control	(f)cr_psupv_ctrl	(f)rr_psupv_ctrl
\$66	Voice Path Control	(f)cd_path_ctrl	(f)rd_path_ctrl
\$6B	Change Incoming Port	(f)cr_ch_iport	(f)rr_ch_iport
\$D5	Route Action	—	(f)rr_route_action
\$C0 00	Configure VCA/Set System Clock	(f)cc_config_vca	—
\$C0 01	Change Active Controllers	(f)cc_ch_sysctrl	—
\$C0 02	T1 Synchronization Control	(f)cc_config_t1	(f)rc_config_t1
\$C0 03	Set/Reset Host Alarms	(f)cc_set_alarms	—
\$C0 04	Host Load Control	(f)cc_load_ctrl	—
\$C0 05	Host Assume/Relinquish Port Control	(f)cc_port_ctrl	(f)rc_port_ctrl
\$DC	Active/Standby Mode	—	(f)rc_act_sby
\$80	Resource Allocation	(f)cs_res_alloc	(f)rs_res_alloc
\$81	Hardware Allocation	(f)cs_hw_alloc	(f)rs_hw_alloc
\$82	Card Status	(f)cs_card_statreq	(f)rs_card_statreq
\$83	Change Port Status	(f)cs_port_statreq	(f)rs_port_statreq
\$D2	Permanent Signal Condition	—	(f)rr_psc
\$90	Change Port Status	(f)cm_ch_pstatus	(f)rm_ch_pstatus

Table 2-6 Commands/Reports and Function Names (continued)

Hex	Command/Report	Function/Structure Name for Command Buildup	Function/Structure Name for Report Parsing
\$D3	Port Status	—	(f)rs_port_status
\$D6	Resource Limitation	—	(f)rs_res_limit
\$D9	Card Status	—	(f)rs_card_status
\$F0	Alarm Condition	—	(f)rs_alarm_cond
\$49	ISDN Port Control	(f)cr_isdn_ctrl	(f)rr_isdn_ctrl
\$EA	ISDN Port Change of State	—	(f)rr_isdn_pcos
\$ED	ISDN Inpulse Rule Complete	—	(f)rr_isdn_irule

Subrate Switching

```
#include "api_sub_switch.h"
```

Subrate Path Control (\$65) Command and Report

Subrate switching gives the system the ability to connect portions of DSO links to other DSO links, called paths. The width of a subrate path can be from 8 kilobits to 64 kilobits, in 8-kilobit increments. The subrate switch command is structured with a single source end point and one or more destination end points. An endpoint is the combination of a port address and a bit offset.



Note

When ASIST is operating in the extended mode, and more than 98 destination endpoints are specified, the built command is too long. Limit the number of destination endpoints to 98 or fewer.

Function

```
int fcr_subrate_ctrl(api_message *msg, char *buf)
int frr_subrate_ctrl(unchar *buf, int len, api_message *msg)
```

Mode Constant

```
CR_SUBRATE_SWITCH
```

Parameters

The parameters for this message are in Table 2-7 and Table 2-8.

Table 2-7 Subrate Switch (\$65) Parameters

Parameter	Type	Description
spacer_byte	SpacerByte	User-definable spacer bytes—compiled in extended mode only.
tone_plan	TonePlan	Reserved for tone plans—compiled in extended mode only.
detach_bearer	unsigned:1	Detach bearer.
idle_channel	unsigned:1	Idle channel.
is_multi_dest_mode	unsigned:1	Multiple destination mode?
is_bulk_mode	unsigned:1	Bulk mode?
is_tearardown	unsigned:1	Tear path down?
path_control	unsigned:2	Path control.
parameter	unsigned short	Parameter.
width	unchar	Subrate path width.
source	endpoint_struct	Source end port address and offset.
dest []	endpoint_struct	Destination port addresses and offsets, up to 166.

Table 2-8 Subrate Switch (\$65) Endpoint and Structure

Parameter	Type	Description
port_address	Paddr	Port address of the end point
offset	unchar	Bit offset at which the path begins

DTMF Digit Collection

```
#include api_dtmf.h
#include api_tone.h
```

DTMF Collection Control (\$67) (Standard and Enhanced) Command and Report

Use these functions to perform the following:

- Generate the DTMF Collection Control (\$67) (Standard and Enhanced) command to be sent to the system, and decode the report.
- Instruct the system to collect DTMF digits sent over any line or trunk circuit without an impulse rule
- Collect dial pulse (DP) digits on a SLIC or DID circuit.
- Attach/detach DTMF receivers to/from a trunk (SLIC, DID, and UTC cards have an onboard DTMF receiver per port).
- Instruct the IPRC to play a voice prompt/announcement via an attached Enhanced Voice Port Control (\$6C) command.

Function

```
fcr_dtmf_ctrl(api_message *msg, char *buf)
frr_dtmf_ctrl(unchar *buf, int len, api_message *msg)
```

Mode Constant

CR_DTMF_CTRL

Parameters

For parameters that are different between standard and extended structure definitions, the type specified in the table is as described in the “Standard versus Extended Operational Mode Host API” section on page 2-2. The actual size varies.

If `msg->cmd_rep.dtmf_ctrl.type = DTMF_STD`, then parameters are as in Table 2-9.

If `msg->cmd_rep.dtmf_ctrl.type = DTMF_ENH`, then parameters are as in Table 2-10.

Table 2-9 DTMF Collection Control (Standard) (\$67) Parameters

Parameter	Type	Description
spacer_byte	SpacerByte	User-definable spacer bytes—compiled in extended mode only.
tone_plan	TonePlan	Reserved for tone plans—compiled in extended mode only.
ctrl_port	Paddr	Controlling port address.
is_switch_reqd	unsigned:1	Is switching action required?
is_port_attach	unsigned:1	Is port to be attached or detached?
is_dtmf_rgrp	unsigned:1	Is a specific DTMF receiver to be used or one from DTMF receiver resource group?
is_dtmf_retain	unsigned:1	Is DTMF receiver to be retained after report?
dtmf_port	Paddr	DTMF receiver port address. If controlling port address resides on DID, UTC or SLIC, then \$00.
is_dtmf_enable	unsigned:1	Is the DTMF receiver to be enabled?
max_digits	unsigned:6	Maximum number of digits to be collected.
reenter_digits	unchar	Reenter digits.
end_digits	unchar	End of string digits.
col_timeout	unchar	Number of seconds allowed for the user to enter max_digits.
is_reenter_beep	unsigned:1	Is a beep tone connected when the user enters reenter code?
is_strend_beep	unsigned:1	Is a beep tone connected when the end of string code is detected?
is_enable_beep	unsigned:1	Is a beep tone connected when DTMF receiver is enabled?

Table 2-10 DTMF Collection Control (Enhanced) (\$67) Parameters

Parameter	Type	Description
spacer_byte	SpacerByte	User definable spacer bytes—compiled in extended mode only.
tone_plan	TonePlan	Reserved for tone plans—compiled in extended mode only.
ctrl_port	Paddr	Controlling port address.
is_switch_reqd	unsigned:1	Is switching action required?
is_port_attach	unsigned:1	Is port to be attached or detached?
is_dtmf_rgrp	unsigned:1	Is a specific DTMF receiver to be used or one from DTMF receiver resource group?
is_dtmf_retain	unsigned:1	Is DTMF receiver to be retained after report?
dtmf_port	Paddr	DTMF receiver port address. If controlling port address resides on DID, UTC or SLIC, then \$00.
is_dtmf_enable	unsigned:1	Is the DTMF receiver to be enabled?
is_enhanced	unsigned:1	Should always be 1.
is_4th_col_enable	unsigned:1	Is fourth column DTMF enabled?
is_tmr_seg	unsigned:1	Is Collection Timers segment attached?
is_eos_seg	unsigned:1	Is Reenter/End of String segment attached?
is_fdig_seg	unsigned:1	Is First Digit Processing String segment attached?
is_enopt_seg	unsigned:1	Is Enabling Options Segment attached?
max_digits	unsigned:6	Maximum number of digits to be collected
is_store_dig	unsigned:1	Are digits to be stored in ports digit field?
is_app_dig	unsigned:1	Are digits to be appended in ports digit field?
dig_field	unsigned:3	Fields in which digits stored/appended.
If is_tmr_seg = TRUE		
fdig_tout	unchar	First Digit Timeout.
idig_tout	unchar	Inter Digit Timeout.
fldig_tout	unchar	Field Timeout.
If is_eos_seg = TRUE		
is_1dig_reenter	unsigned:1	Is a single-digit reenter code used?
is_2dig_reenter	unsigned:1	Is a two-digit reenter code used?
is_1dig_eos	unsigned:1	Is a single-digit end of string code used?
is_2dig_eos	unsigned:1	Is a two-digit end of string code used?
is_rent_notone	unsigned:1	Is no tone on reenter code detection to be connected?
is_eos_notone	unsigned:1	Is no tone on end of string code detection to be connected?
rent_dig_code	unsigned:1	One- or two-digit DTMF digit reenter code.
eos_code	unsigned:1	One- or two-digit DTMF digit end of string code.
rent_tone_code	unsigned:1	Reenter tone on detection of reenter code.
eos_tone_code	unsigned:1	Reenter tone on detection of end of string code.

Table 2-10 DTMF Collection Control (Enhanced) (\$67) Parameters (continued)

Parameter	Type	Description
field_tout	unchar	Field Timeout.
If is_fdig_seg = TRUE		
is_rep_fdig	unsigned:1	Is DTMF digit report sent to host on detection of first digit?
is_fdig_tone	unsigned:1	Is a tone to be presented on detection of first digit?
is_fdig_wink	unsigned:1	Is a wink to be presented on detection of first digit?
is_fdig_abort	unsigned:1	Is DVC voice prompt being presented to be aborted on detection of first digit?
is_detach_og	unsigned:1	Is line/trunk attached to port to be detached on detection of first digit?
fdig_tone	unchar	Tone to be presented on detection of first digit.
If is_enopt_seg = TRUE		
is_enab_rec	unsigned:1	Is receiver to be enabled immediately or after condition satisfied?
is_enh_dvc	unsigned:1	Is an enhanced \$6C segment attached?
is_tone_renb	unsigned:1	Is a tone to be presented when receiver enabled?
is_wink_renb	unsigned:1	Is a wink to be presented when receiver enabled?
is_tmr_pause	unsigned:1	Pause before starting first digit timer?
is_tmr_sup	unsigned:1	Wait for supervision event before starting digit timer?
is_tmr_dvc	unsigned:1	Present (up to 14) voice prompts before starting digit timer?
If is_enh_dvc = TRUE		
dvc_ctrl	cr_dvc_ctrl	\$6C segment (refer to the “DVC Port Control (\$6C) Command and Report” section on page 2-20).

DTMF Digit Collection (\$D1) (Standard and Enhanced) Report Parsing

This function does the following:

- Analyzes the DTMF Digit Collection (\$D1) report sent from the system.
- Transfers DTMF/DP digit collection information to the host application.
- Generates a report indicating whether digit report is valid, and the line/trunk for which digits were collected. If a timeout occurs, any digits collected up to that point are returned.
- Collects a maximum of 40 digits.

The report produced by this function also indicates:

- Report generated for first digit receipt.
- DVC prompt being presented was aborted.
- Timeout occurred while waiting for supervision.
- Digit field overflow occurred.
- Receiver port not available at first request (hunt only).

Function

```
frr_dtmf_digits(unchar *buf, int len, api_message *msg)
```

Mode Constant

```
RR_DTMF_DIGITS
```

Parameters

The parameters for this message are in Table 2-11.

Table 2-11 DTMF Digit Collection Report (\$D1) Parsing Parameters

Parameter	Type	Description
ctrl_port	Paddr	Controlling port address.
is_prmt_abort	unsigned:1	Was DVC prompt aborted after the user entered the first digit?
was_out_detach	unsigned:1	Was the outgoing port detached on first digit detection?
did_sup_fire	unsigned:1	Was digit collection aborted and the receiver removed because the supervision timer fired?
did_dig_flow	unsigned:1	Did digit field overflow?
why_sup_fired	unchar	Indicates why supervision timer fired.
dtmf_port	Paddr	DTMF receiver port address. Identical to controlling port address for SLICs, DIDs and UTCs.
is_enh_dtmf	unsigned: 1	0 (zero) if report following is old style. 1 (one) if report following is enhanced report format.
did_idig_fire	unsigned:1	Did the interdigit timer fire?
is_fdig_rep	unsigned:1	Is it the first digit report?
was_rec_avail	unsigned:1	Was DTMF receiver available on initial request?
did_ddig_fire	unsigned:1	Did digit collection timer fire?
did_fdig_fire	unsigned:1	Did first digit timer fire?
is_rep_valid	unsigned:1	Is DTMF digit report valid?
field_id	unchar	Field in which the system stores reported digits when Enable Digit Field Reporting feature is enabled.
digits	array, unchar	Pointer to array containing digits collected.
digit_count	unchar	Number of digits collected.

MF Digit Collection

```
#include "api_mf.h"
```

MF Collection Control (\$68) Command and Report

These functions do the following:

- Generate the MF Collection Control (\$68) command that is sent to the system, and parse the reports.

- Enable the host to collect MF digits sent over a trunk.
- Attach/detach MF receivers to/from a trunk. Up to 40 MF digits can be collected in a 30-second period.
- Allow call to remain in active state if garbled digits are received or no KP/ST is detected. The default is to tear down the call.

Function

```
fcr_mf_ctrl(api_message *msg, char *buf)
frr_mf_ctrl(unchar *buf, int len, api_message *msg)
```

Mode Constant

CR_MF_CTRL

Parameters

The parameters for this message are in Table 2-12.

Table 2-12 MF Collection Control (\$68) Parameters

Parameter	Type	Description
spacer_byte	SpacerByte	User-definable spacer bytes—compiled in extended mode only.
tone_plan	TonePlan	Reserved for tone plans—compiled in extended mode only.
ctrl_port	Paddr	Controlling port address.
is_switch_reqd	unsigned:l	Is switching action required?
is_port_attach	unsigned:l	Is port to be attached or detached?
is_mf_rgrp	unsigned:l	Is a specific MF receiver to be used or one from MF receiver resource group?
is_mf_retain	unsigned:l	Is MF receiver to be retained after report?
is_tearcall	unsigned:l	Is the call to be torn down upon digit collection failure?
mf_port	unsigned:11	MF receiver port address.
spacer_byte	unchar	Spacer byte.
mf_enable	unchar	MF receiver can either be enabled or not (MF_ENABLE or MF_DISABLE).

MF Digit Collection (\$D0) Report Parsing

This function does the following:

- Analyzes the MF Digit Collection (\$D0) report sent from the system.
- Transfers MF digit collection information from the system to the host application.
- Generates a report indicating whether the digit report is valid, and the incoming port from which digits were collected. The report also indicates the present state of controlling port (CP_SETUP or forced to idle) and whether it detected garbled MF digits.
- Collects a maximum of 40 digits.

Function

```
frr_mf_digits(unchar *buf, int len, api_message *msg)
```

Mode Constant

```
RR_MF_DIGITS
```

Parameters

The parameters for this function are in Table 2-13.

Table 2-13 MF Digit Collection (\$D0) Report Parsing Parameters

Parameter	Type	Description
ctrl_port	Paddr	Incoming port address.
spacer_bytes	ushort	Spacer bytes.
mf_port	Paddr	MF receiver port address.
is_rep_garbled	unsigned:1	Is MF report garbled?
is_port_idle	unsigned:1	If MF report is garbled, has the controlling port been forced to idle state or placed in setup state?
is_mf_avail	unsigned:1	Is MF receiver available when initially requested?
did_mf_fire	unsigned:1	Did the MF digit collection timer fire?
is_rep_valid	unsigned:1	Is MF digit report valid?
field_id	unchar	Field in which the system stores reported digits when Enable Digit Field Reporting feature is enabled.
digits	array, unchar	Array containing digits collected.
digit_count	unchar	Number of digits collected.

Playing Digitized Voice Prompts

```
#include "api_dvc.h"
```

DVC Port Control (\$6C) Command and Report

Use these functions do the following:

- Generate the DVC Port Control (\$6C) command that is sent to the system, and parse the report returned from the system.
- Instruct the system to play up to 14 prompts (or 20, with Enhanced \$6C command) to a line or trunk port. All prompts are downloaded to the Digital Voice Card (DVC) at system boot.
- Link or remove a DVC port to or from a call's resource chain.
- Serve as a command segment in an Incoming Port Control (\$6A) command.
- Play or record voice prompts.

Function

```
fcr_dvc_ctrl(api_message *msg, char *buf)
frr_dvc_ctrl(unchar *buf, int len, api_message *msg)
```

Mode Constant

```
CR_DVC_CTRL
```

Parameters

The parameters for this message are in Table 2-14.

Table 2-14 DVC Port Control (\$6C) Parameters

Parameter	Type	Description
spacer_byte	SpacerByte	User-definable spacer bytes—compiled in extended mode only.
tone_plan	TonePlan	Reserved for tone plans—compiled in extended mode only.
ctrl_port	Paddr	Controlling port address.
is_switch_reqd	unsigned:1	Is switching action required?
is_port_attach	unsigned:1	Is a DVC port to be attached or detached?
is_dvc_rgrp	unsigned:1	Is a specific DVC port to be used or one from a resource group?
is_dvc_rel	unsigned:1	Is DVC port to be released after prompts have been played?
dvc_port	Paddr	DVC port address group to search for.
is_play_prompt	unsigned:1	Is a prompt to be played on a line/trunk?
is_genrep_prmt	unsigned:1	Is a \$DE report to be generated when all the prompts have been played?
enh_dvc	unsigned:1	Is this an enhanced DVC command?
seg_attach	unsigned:1	Is play (0) or record (1) segment attached (enhanced only)?
no_prompts	unsigned:4	Number of prompts to be played (maximum 14).
phrases	char pointer	Pointer to a char array of phrase numbers to be played.
enh_record_seg	rec_seg	Record segment for (enhanced only).
enh_play_seg	play_seg	Play segment for (enhanced only).

DVC Port Status (\$DE) Report Parsing

This function analyzes a DVC Port Status (\$DE) report sent from the system and indicates when all voice prompts specified in a DVC Control (\$6C) command have completed.

Function

```
frc_dvc_status(unchar *buf, int len, api_message *msg)
```

Mode Constant

```
RC_DVC_STATUS
```

Parameters

The parameters for this message are in Table 2-15.

Table 2-15 DVC Port Status (\$DE) Parsing Parameters

Parameter	Type	Description
ctrl_port	Paddr	Address of incoming port to which voice prompts were played.
spacer_bytes	ushort	Spacer bytes.
dvc_port	Paddr	Address of DVC port used to present prompts.
status	unchar	Indicated status of the digit report.

Voice Prompt Maintenance (\$91) Command and Report

These functions generate the Voice Prompt Maintenance Control (\$91) command that is sent to the system, and parse the report returned from the system. The \$91 command provides a mechanism for the host to:

- Upload voice prompt information from one or more IPRCs
- Download prompt information to one or more IPRCs.

Function

```
fcr_vpm_ctrl(api_message *msg, char *buf)
```

```
frr_vpm_ctrl(unchar *buf, int len, api_message *msg)
```

Mode Constant

```
CR_VPM_CTRL
```

Parameters

The parameters for this message are in Table 2-16.

Table 2-16 Voice Prompt Maintenance (\$91) Parameters

Parameters	Type	Description
spacer_byte	SpacerByte	User-definable spacer byte—compiled in extended mode only.
tone_plan	TonePlan	Reserved for tone plans—compiled in extended mode only.
control_code	unchar	0 = download prompt information; 1 = upload.
access_code	unchar	0 = Access card containing port specified by port address code; 1 = access card specified by RLS code; 2 = access all cards supporting specified prompt library (download only).
rls_code	unchar	See Hardware Allocation (\$81) report for RLS code specification; set to \$00 if accessing by port address.
ctrl_port	Paddr	Card that contains port address will be accessed. Set to \$0000 if accessing by RLS code.
source_library	unchar	Hexadecimal representation of library ID (\$00 to \$0F).

Table 2-16 Voice Prompt Maintenance (\$91) Parameters (continued)

Parameters	Type	Description
temp_prompt	unsigned: 1	Is this a temporary prompt?
source_prompt_id	ushort	Source prompt ID.
dest_lib	uchar	Destination library (upload only) (\$00 to \$0F).
dest_prompt_id	ushort	Destination prompt ID (upload only) (\$0001 to \$00FF).

Controlling Multi-Party Conferences

```
#include "api_dcc.h"
```

Conference Control (\$6D) Command and Report

These functions generate the Conference Control (\$6D) command that is sent to the system, and the report returned by the system. It controls conferencing features. Up to eight conference ports can be used for a conference. The system supports up to 128 simultaneous conferences.

Function

```
fcr_dcc_ctrl(api_message *msg, char *buf)
frr_dcc_ctrl(uchar *buf, int len, api_message *msg)
```

Mode Constant

```
CR_DCC_CTRL
```

Parameters

The parameters for this message are in Table 2-17.

Table 2-17 Conference Control (\$6D) Parameters

Parameter	Type	Description
spacer_byte	SpacerByte	User-definable spacer byte—compiled in extended mode only.
tone_plan	TonePlan	Reserved for tone plans—compiled in extended mode only.
confr_no	unsigned short	Conference number.
is_reserve_conf	unsigned:1	Is conference to be reserved?
is_start_conf	unsigned:1	Is conference to be started?
is_tear_conf	unsigned:1	Is conference to be torn down?
is_add_conf	unsigned:1	Are one or more line/trunk ports to be added to conference?
is_del_conf	unsigned:1	Are one or more line/trunk ports to be deleted from a conference?
is_level_adj	unsigned:1	Is the input/output level adjustment necessary?
port_count	uchar	Number of ports affected.

Table 2-17 Conference Control (\$6D) Parameters (continued)

Parameter	Type	Description
For Each Port (port_count):		
inp_level_adj	unchar	Input level adjustment (00 to 15).
is_olevel_dec	unsigned:1	Is output level of conference port associated with the line/trunk port to be decremented by 3 dB?
is_voice_2way	unsigned:1	Is a two-way or one-way voice path to be set up?
dcc_port	Paddr	Address of port involved in conference.

Port Hook Control

```
#include "api_hook.h"
```

Port Hook State Control (\$70) Command and Report

The Port Hook State Control (\$70) command provides the host with the ability to cause onhook and offhook processing on a line or trunk port. The host may also start impulse or outpulse rule processing with this command. Hook state control is useful when hook state events are received external to the switch, such as SS7.

Function

```
fcd_hook_ctrl(api_message *msg, char *buf)
frd_hook_ctrl(unchar *buf, int len, api_message *msg)
```

Mode Constant

```
CD_HOOK_CTRL
```

Parameters

The parameters for this message are in Table 2-18.

Table 2-18 Port Hook State Control (\$70) Parameters

Parameter	Type	Description
spacer_byte	SpacerByte	User-definable spacer bytes—compiled in extended mode only.
tone_plan	TonePlan	Reserved for tone plans—compiled in extended mode only.
ctrl_port	Paddr	Address of the port being affected.
supp_dadb	unsigned:1	Suppress DA and DB reports.
hook_state	unsigned:1	Hook State: <ul style="list-style-type: none"> • 0 = On hook. • 1 = Off hook.

Table 2-18 Port Hook State Control (\$70) Parameters (continued)

Parameter	Type	Description
class	uchar	Class of Service: <ul style="list-style-type: none"> • 0 = Use port's configured COS. • 1 = Force port to incoming. • 2 = Force port to outgoing.
do_irule	unsigned:1	Inpulse rule control.
do_orule	unsigned:1	Outpulse rule control.
rule_id	Rule	Rule number.

Controlling Line/Trunk Network Interfaces

```
#include "api_dtmf.h"
#include "api_mf.h"
#include "api_src.h"
#include "api_dvc.h"
#include "api_net.h"
```

Outgoing Port Control (\$69) Command and Report

Use these functions to do the following:

- Generate the Outgoing Port Control (\$69) command that is sent to the system, and parse the report from the system.
- Link or remove outgoing circuits to or from a call's resource chain.
- Begin outputpulse/inpulse rule processing for an outgoing port.
- Overwrite digit strings contained in call record fields and supply new digits.
- Disconnect call at teardown.

Function

```
fcr_oport_ctrl(api_message *msg, uchar *buf)
frr_oport_ctrl(uchar *buf, int len, api_message *msg)
```

Mode Constant

```
CR_OPORT_CTRL
```

Parameters

The parameters for this message are in Table 2-19.

Table 2-19 Outgoing Port Control (\$69) Parameters

Parameter	Type	Description
spacer_byte	SpacerByte	User-definable spacer bytes—compiled in extended mode only.
tone_plan	TonePlan	Reserved for tone plans—compiled in extended mode only.
ctrl_port	Paddr	Incoming port address.
is_switch_reqd	unsigned:1	Is switching action required?
is_port_attach	unsigned:1	Is port to be attached or detached?
is_oport_rgrp	unsigned:1	Is a specific outgoing port to be used or one from a resource group?
twoway_path	unsigned:2	Defer 2-way path until end of outpulse rule (DEFER_OPUL), defer until outgoing answers (DEFER_OANS), or cut 2-way speech instantly (CUT_SPEECH).
oport	Paddr	Outgoing port address/resource group to search for.
is_opul_exec	unsigned:1	Is an outpulse rule to be executed?
is_discon_byte	unsigned:1	Is Disconnect Control byte included in the command?
is_ipul_exec	unsigned:1	Is an inpulse rule to be executed?
rule_number	Rule	Inpulse/outpulse number.
is_irep_sup	unsigned:1	Is onhook report for incoming port to be suppressed if outgoing port goes onhook first?
is_ic_setup	unsigned:1	Is incoming port to return to CP_SETUP if outgoing port goes onhook first?
is_orep_sup	unsigned:1	Is onhook report for outgoing port to be suppressed?
field_no	unsigned:3	Call record field to receive digit string.
no_digits	unsigned:5	Number of digits in string between 0 to 40 (0 to 12 for ANI).
dig_string	array, unchar	Digit string.

Outgoing Port Change of State (\$DA) Report Parsing

This function does the following:

- Analyzes Outgoing Port Change of State (\$DA) report sent from the system.
- Informs host of a change in hardware state of an outgoing the system port.
- Indicates whether an outpulse rule has been successfully completed.



Note In order for the system to generate a report indicating outpulse rule completion, a REP END token must be contained in the outpulse rule.

- Indicates supervision errors in change byte **oport_change**.
- Indicates when a rehunt of an outgoing port is performed.

Function

```
frr_oport_cos(unchar *buf, int len, api_message *msg)
```

Mode Constant

```
RR_OPORT_COS
```

Parameters

The parameters for this message are in Table 2-20.

Table 2-20 Outgoing Port Change of State Report (\$DA) Parameters

Parameter	Type	Description
res_group	Group	Resource group number.
oport_change	unchar	Change occurring on outgoing port. Can be ACT_OPORT, INACT_OPORT, SUPRERR_OPORT, SUPDET_OPORT, SUPOPUL_OPORT, OUTPUT_OPORT or HUNT_OPORT.
oport	Paddr	Outgoing port address.
iport	Paddr	Incoming port address.
supv_code	ushort	Answer Supervision Code.
is_og_ans	unsigned:1	Is outgoing port considered answered?
supv_tmplate	unsigned:6	Answer supervision template used.
new_oport	Paddr	Address of new outgoing port selected by the system as a result of rehunt.

Incoming Port Control (Macro) (\$6A) Command and Report

These functions do the following:

- Generate the Incoming Port Control (\$6A) command that is sent to the system, and parse the report from the system.
- Instruct the system to force call origination or disconnect, begin an impulse or outpulse rule, or execute one of the following:
 - Port Supervision Control (\$72) command
 - DTMF Collection Control (Standard or Enhanced) (\$67) command
 - MF Collection Control (\$68) command
 - Outgoing Port Control (\$69) command
 - DVC Port Control (\$6C) command
 - Outpulse Control segment

**Note**

Specify only one impulse rule, outpulse rule or command segment in a single command. You can include up to five outpulse control segments in a single command when you specify an outpulse or impulse rule for an incoming port.

Function

```
fcr_iport_ctrl(api_message *msg, char *buf)
frr_iport_ctrl(unchar *buf, int len, api_message *msg)
```

Mode Constant

```
CR_IPORT_CTRL
```

Parameters

The parameters for this message are in Table 2-21.

Table 2-21 Incoming Port Control (Macro) (\$6A) Command and Report Parameters

Parameter	Type	Description
spacer_byte	SpacerByte	User-definable spacer bytes—compiled in extended mode only.
tone_plan	TonePlan	Reserved for tone plans—compiled in extended mode only.
iport	Paddr	Incoming port address.
oport	Paddr	Outgoing port address/resource group to search for.
is_switch_reqd	unsigned:1	Is switching action required?
is_port_attach	unsigned:1	Is a call to be originated or disconnected?
is_iport_rgrp	unsigned:1	Is a specific incoming port to be used or one from a resource group?
is_ipdcon_sup	unsigned:1	Should incoming go to CP_SETUP state on forced disconnect?
is_opdcon_sup	unsigned:1	Should outgoing go to CP_SETUP state on forced disconnect?
is_ipon_rep	unsigned:1	Are on hooks for incoming ports to be reported?
is_oon_rep	unsigned:1	Are on hooks for outgoing ports to be reported?
is_opul_exec	unsigned:1	Is an outpulse rule to be executed?
is_ipul_exec	unsigned:1	Is an inpulse rule to be executed?
rule_number	Rule	Inpulse/outpulse number.
is_psup_seg	unsigned:1	Is Port Supervision Control Command segment attached?
is_dtmf_seg	unsigned:1	Is DTMF Collection Control Command segment attached?
is_mf_seg	unsigned:1	Is MF Collection Control Command segment attached?
is_dvc_seg	unsigned:1	Is DVC Port Control Command segment attached?
is_ogcon_seg	unsigned:1	Is Outgoing Port Control Command segment attached?
is_iccon_seg	unsigned:1	Is Incoming Port Control Command segment attached?
is_output_seg	unsigned:1	Is Outpulse Rule Control Command segment attached?
If is_psup_seg=TRUE		
is_psup_exe	unsigned:1	Is supervision action to be executed or cancelled?
psup_action	unsigned:3	Port supervision action.
If is_mf_seg = TRUE		
is_switch_reqd	unsigned:1	Is switching action required?

Table 2-21 Incoming Port Control (Macro) (\$6A) Command and Report Parameters (continued)

Parameter	Type	Description
is_port_attach	unsigned:1	Is port to be attached or detached?
is_mf_rgrp	unsigned:1	Is a specific MF receiver to be used or one from MF receiver resource group?
is_mf_retain	unsigned:1	Is MF receiver to be retained after report?
is_tearcall	unsigned:1	Is the call to be torn down upon digit collection failure?
mf_port	Paddr	MF receiver port address.
mf_enable	uchar	MF receiver enabled?
If is_dtmf_seg = TRUE:		
If the Standard DTMF Collection segment is used:		
is_switch_reqd	unsigned:1	Is switching action required?
is_port_attach	unsigned:1	Is port to be attached or detached?
is_dtmf_rgrp	unsigned:1	Is a specific DTMF receiver to be used or one from DTMF receiver resource group?
is_dtmf_retain	unsigned:1	Is DTMF receiver to be retained after report?
dtmf_port	Paddr	DTMF receiver port address. If controlling port address resides on DID, UTC or SLIC, then \$00.
is_dtmf_enable	unsigned:1	Is the DTMF receiver to be enabled?
max_digits	unsigned:6	Maximum number of digits to be collected.
reenter_digits	uchar	Reenter digits.
end_digits	uchar	End of string digits.
col_timeout	uchar	Number of seconds allowed for the user to enter max_digits.
is_reenter_beep	unsigned:1	Is a beep tone connected when user enters reenter code?
is_strend_beep	unsigned:1	Is a beep tone connected when end of string code detected?
is_enable_beep	unsigned:1	Is a beep tone connected when DTMF receiver is enabled?
If Enhanced DTMF Collection Segment Used:		
is_switch_reqd	unsigned:1	Is switching action required?
is_port_attach	unsigned:1	Is port to be attached or detached?
is_dtmf_rgrp	unsigned:1	Is a specific DTMF receiver to be used or one from DTMF receiver resource group?
is_dtmf_retain	unsigned:1	Is DTMF receiver to be retained after report?
dtmf_port	Paddr	DTMF receiver port address. If controlling port address resides on DID, UTC or SLIC, then \$00.
is_dtmf_enable	unsigned:1	Is the DTMF receiver to be enabled?
is_enhanced	unsigned:1	Should always be 1.
is_tmr_seg	unsigned:1	Is Collection Timers segment attached?
is_eos_seg	unsigned:1	Is Reenter/End of String segment attached?
is_fdig_seg	unsigned:1	Is First Digit Processing String segment attached?

Table 2-21 Incoming Port Control (Macro) (\$6A) Command and Report Parameters (continued)

Parameter	Type	Description
is_enopt_seg	unsigned:1	Is the Enabling Options segment attached?
max_digits	unsigned:6	Maximum number of digits to be collected.
is_store_dig	unsigned:1	Are digits to be stored in ports digit field?
is_app_dig	unsigned:1	Are digits to be appended in ports digit field?
dig_field	unsigned:3	Fields in which digits stored/appended.
If is_tmr_seg = TRUE:		
fdig_tout	unchar	First Digit Timeout.
idig_tout	unchar	Inter Digit Timeout.
fldig_tout	unchar	Field Timeout.
If is_eos_seg = TRUE:		
is_1dig_reenter	unsigned:1	Is a single-digit reenter code used?
is_2dig_reenter	unsigned:1	Is a two-digit reenter code used?
is_1dig_eos	unsigned:1	Is a single-digit end of string code used?
is_2dig_eos	unsigned:1	Is a two-digit end of string code used?
is_rent_notone	unsigned:1	Is no tone on reenter code detection to be connected?
is_eos_notone	unsigned:1	Is no tone on end of string code detection to be connected?
rent_dig_code	unsigned:1	One- or two-digit DTMF digit reenter code.
eos_code	unsigned:1	One- or two-digit DTMF digit end of string code.
rent_tone_code	unsigned:1	Reenter tone on detection of reenter code.
eos_tone_code	unsigned:1	Reenter tone on detection of end of string code.
field_tout	unchar	Field Timeout.
If is_fdig_seg = TRUE:		
is_rep_fdig	unsigned:1	Is DTMF digit report sent to host on detection of first digit?
is_fdig_tone	unsigned:1	Is a tone to be presented on detection of first digit?
is_fdig_wink	unsigned:1	Is a wink to be presented on detection of first digit?
is_fdig_abort	unsigned:1	Is a DVC voice prompt being presented to be aborted on detection of first digit?
is_detach_og	unsigned:1	Is line/trunk attached to port to be detached on detection of first digit?
fdig_tone	unchar	Tone to be presented on detection of first digit.
If is_enopt_seg = TRUE		
is_enab_rec	unsigned:1	Is receiver to be enabled immediately or after condition satisfied?
is_tone_renb	unsigned:1	Is a tone to be presented when the receiver is enabled?
is_wink_renb	unsigned:1	Is a wink to be presented when the receiver is enabled?
is_tmr_pause	unsigned:1	Pause before starting first digit timer?
is_tmr_sup	unsigned:1	Wait for supervision event before starting digit timer?

Table 2-21 Incoming Port Control (Macro) (\$6A) Command and Report Parameters (continued)

Parameter	Type	Description
is_tmr_dvc	unsigned:1	Present (up to 14) voice prompts before starting digit timer?
Incoming Port Control (Macro) Command (\$6A) Buildup		
If is_dvc_seg = TRUE:		
is_switch_reqd	unsigned:1	Is switching action required?
is_port_attach	unsigned:1	Is a DVC port to be attached or detached?
is_dvc_rgrp	unsigned:1	Is a specific DVC port to be used or one from a resource group?
is_dvc_retain	unsigned:1	Is DVC port to be retained after prompts have been played?
dvc_port	Paddr	DVC port address group to search for.
is_play_promp	unsigned:1	Is a prompt to be played on a line/trunk?
is_genrep_prmt	unsigned:1	Is a \$DE report to be generated when all the prompts have been played?
no_prompts	unsigned:4	Number of prompts to be played (maximum 14).
phrases	char pointer	Pointer to a char array of phrase numbers to be played.
If is_ogcon_seg = TRUE:		
is_switch_reqd	unsigned:1	Is switching action required?
is_port_attach	unsigned:1	Is port to be attached or detached?
is_oport_rgrp	unsigned:1	Is a specific outgoing port to be used or one from a resource group?
two_way_path	unsigned:2	Defer 2-way path until end of outpulse rule (DEFER_OPUL), defer until outgoing answers (DEFER_OANS) or cut 2-way speech instantly (CUT_SPEECH).
oport	Paddr	Outgoing port address/resource group to search for.
is_opul_exec	unsigned:1	Is an outpulse rule to be executed?
is_discon_byte	unsigned:1	Is Disconnect Control byte included in the command?
is_ipul_exec	unsigned:1	Is an inpulse rule to be executed?
rule_number	Rule	Inpulse/outpulse rule number.
field_no	unsigned:3	Call record field to receive digit string.
no_digits	unsigned:5	Number of digits in string between 0 to 40 (0 to 12 for ANI)
dig_string	array, unchar	Digit string.
is_irep_sup	input	Is onhook report for incoming port to be suppressed if outgoing port goes onhook first?
is_ic_setup	unsigned:1	Is incoming port to return to CP_SETUP if outgoing port goes onhook first?
is_orep_sup	unsigned:1	Is onhook report for outgoing port to be suppressed?
If is_output_seg = TRUE:		

Table 2-21 Incoming Port Control (Macro) (\$6A) Command and Report Parameters (continued)

Parameter	Type	Description
no_dopul_segs	integer	The number of outpulse control segments contained within the array <code>ipport_segs.dopul_seg[]</code> .
dopul_seg[]	array of typedig_opul_ seg	Contains one to five outpulse control segments.

Incoming Port Change of State (\$DB) Report Parsing

This function does the following:

- Analyzes Incoming Port Change of State (\$DB) report sent from the system.
- Informs host of a change in hardware state of an incoming system port.
- Indicates whether an inpulse rule has been successfully completed.

Function

```
frr_ipport_cos(unchar *buf, int len, api_message *msg)
```

Mode Constant

```
RR_IPPORT_COS
```

Parameters

The parameters for this message are in Table 2-22.

Table 2-22 Incoming Port Change of State (\$DB) Report Parameters

Parameter	Type	Description
res_group	Group	Resource group number.
ipport_change	unchar	Change occurring on incoming port. Can be ACT_IPPORT, INACT_IPPORT, SUPRERR_IPPORT or OUTPUT_IPPORT.
ipport	Paddr	Incoming port address.
supv_code	ushort	Answer supervision code.
supv_tmplate	unsigned:6	Answer supervision template used.

Inpulse Rule Complete (Macro) (\$DD) Report Parsing

This function does the following:

- Analyzes Impulse Rule Complete (\$DD) report sent from the system.
- Informs the host that an inpulse rule has been processed. The content of the report is controlled by the type of reporting specified in the inpulse rule. If REP EACH is specified, the report will indicate only that inpulse rule processing has ended. If REP END is specified, the report is a macro containing resource control reports (segments) to represent all actions taken during inpulse rule execution.

Function

```
frr_ipulse_rule(unchar *buf, int len, api_message *msg)
```

Mode Constant

```
RR_IPULSE_RULE
```

Parameters

The parameters for this message are in Table 2-23.

Table 2-23 Inpulse Rule Complete (Macro) (\$DD) Report Parameters

Parameter	Type	Description
ctrl_port	Paddr	Address of controlling port for which inpulse rule executed.
is_port_inc	unsigned:1	Is port incoming?
is_loop_abort	unsigned:1	Has rule been aborted because of looping?
route_action	unsigned:1	Has a ROUTE token been executed?
seg_count	unsigned:3	Number of segments in inpulse report
is_dvc_avail	unsigned:1	Was DVC port available on initial request?
is_inpul_abort	unsigned:1	Was inpulse processing aborted?
is_no_outh	unsigned:1	Was rule aborted because of exhaustion of outpulse channel?
rule_id	Rule	Inpulse rule executed.
seg_type	unchar	Type of segment—ic port change of state, DTMF collection or MF collection.
If seg_type = FR_IPORT_COS:		
iport_change	unchar	Change occurring on incoming port.
If seg_type = FR_MF_DIGIT:		
mf_port	Paddr	MF receiver port address.
is_rep_garbled	unsigned:1	Is MF receiver garbled?
is_port_idle	unsigned:1	If MF report is garbled, is port in idle state or setup?
is_mf_avail	unsigned:1	Is MF receiver available when initially requested?
did_mf_fire	unsigned:1	Did MF digit collection timer fire?
is_rep_valid	unsigned:1	Is MF digit collection valid?
digits	array, unchar	Array containing MF digits collected.
digit_count	unchar	Number of digits collected.
field_id	unchar	Field number in which digits are stored.
If seg_type = FR_DTMF_DIGIT:		
dtmf_port	Paddr	DTMF receiver port address.
did_idig_fire	unsigned:1	Did interdigit timer fire?
is_fdig_rep	unsigned:1	Is it first digit report?
was_rec_avail	unsigned:1	Is DTMF receiver available when initially requested?

Table 2-23 Impulse Rule Complete (Macro) (\$DD) Report Parameters (continued)

Parameter	Type	Description
did_ddig_fire	unsigned:1	Did digit collection timer fire?
did_fdig_fire	unsigned:1	Did first digit timer fire?
is_rep_valid	unsigned:1	Is MF digit collection valid?
digits	array, unchar	Array containing DTMF digits collected.
digit_count	unchar	Number of digits collected.
field_id	unchar	Field number in which digits stored.

Port Supervision Control (\$72) Command and Report

These functions generate the Port Supervision Control (\$72) command that is sent to the system, and parse the report from the system. It is used for manual host control of outward handshake and supervision signals on both incoming and outgoing circuits.

Function

```
fcr_psupv_ctrl(api_message *msg, char *buf)
frr_psupv_ctrl(unchar *buf, int len, api_message *msg)
```

Mode Constant

```
CD_PSUPV_CTRL
```

Parameters

The parameters for this message are in Table 2-24.

Table 2-24 Port Supervision Control (\$72) Command and Report Parameters

Parameter	Type	Description
spacer_byte	SpacerByte	User-definable spacer bytes—compiled in extended mode only.
tone_plan	TonePlan	Reserved for tone plans—compiled in extended mode only.
ctrl_port	Paddr	Controlling port address.
is_psup_exe	unsigned:1	Is supervision action to be executed (seize/wink) EXEC_PSUP or cancelled (seize) CANCEL_PSUP?
psup_action	unsigned:3	Port supervision action (can be SEIZE_SUP or WINK_PSUP).

Establishing Voice Paths Between Ports

```
#include "api_path.h"
#include "api_tone.h"
```

Voice Path Control (\$66) Command and Report

These functions generate the Voice Path Control (\$66) command that is sent to the system, and parse the report from the system. It is used for immediate setup of voice paths between receivers and senders. A receiver can be an incoming circuit, outgoing circuit, MF receiver, DTMF receiver, or SRC port. A sender can be an incoming circuit, outgoing circuit, system tones, DVC port, or DCC port. The voice path remains established until it is torn down by:

- Release of one of the circuits involved
- A call processing action
- An inpulse or outpulse rule
- Another voice path control command
- Resource Control command

It is possible to tear down a two-way path in only one direction, converting it to a one-way path. It can also be used to set a conference party to listen to a tone. A second command will send it back to conference.

Function

```
fcd_path_ctrl(api_message *msg, char *buf)
```

```
frd_path_ctrl(unchar *buf, int len, api_message *msg)
```

Mode Constant

```
CD_PATH_CTRL
```

Parameters

The parameters for this message are in Table 2-25.

Table 2-25 Voice Path Control (\$66) Parameters

Parameter	Type	Description
spacer_byte	SpacerByte	User-definable spacer bytes—compiled in extended mode only.
path_type	unchar	Type of voice path to be constructed. Can be either <code>BREAK_PATH</code> , <code>ONEWAY_PATH</code> or <code>TWOWAY_PATH</code> (for non-ISDN channels) or <code>BREAK_ISDNHC_PATH</code> , <code>ONEWAY_ISDN_PATH</code> or <code>TWOWAY_ISDNHC_PATH</code> (for ISDN channels).
recv_port	Paddr	Port address of receiver.
send_port	Paddr	Port address of sender.
num_b_chans	unchar	Number of adjacent ISDN B-channels.

Change Incoming Port (\$6B) Command and Report

These functions generate the Change Incoming Port (\$6B) command that is sent to the system, and parse the report returned from the system. It switches all resources for an active call from one incoming port to another. The original port is forced to an idle state.

Function

```
fcr_ch_iport(api_message *msg, char *buf)
frr_ch_iport(unchar *buf, int len, api_message *msg)
```

Mode Constant

```
CR_CH_IPOINT
```

Parameters

The parameters for this message are in Table 2-26.

Table 2-26 Change Incoming Port (\$6B) Command and Report Parameters

Parameter	Type	Description
spacer_byte	SpacerByte	User definable spacer bytes—compiled in extended mode only.
tone_plan	TonePlan	Reserved for tone plans—compiled in extended mode only.
old_iport	Paddr	Old incoming port address.
is_ipdcon_clr	unsigned:1	Are disconnect control bits on old incoming port to be cleared or carried?
is_opdcon_clr	unsigned:1	Are disconnect control bits on outgoing port to be cleared or retained?
is_iport_rgrp	unsigned:1	Is a specific incoming port to be used or one from a resource group?
new_iport	Paddr	New incoming port address group to search for.

Route Action (\$D5) Report Parsing

This function parses a Routing Action (\$D5) report. The \$D5 report provides the host with information about the outcome of a ROUTE inpulse rule operation. This information includes the action (inpulse or outpulse rule) performed, the status of that action, and the connected incoming and outgoing ports.

Function

```
frr_route_action(unchar *buf, int len, api_message *msg)
```

Mode Constant

```
RR_ROUTE_ACTION
```

Parameters

The parameters for this message are in Table 2-27.

Table 2-27 Route Action (\$D5) Report Parameters

Parameter	Type	Description
iport	Paddr	Incoming controlling port.
action	unchar	Routing action performed; either inpulse or outpulse rule executed.

Table 2-27 Route Action (\$D5) Report Parameters (continued)

Parameter	Type	Description
status	unchar	Status of the routing action; subset of network status byte values.
oport	Paddr	Outgoing port involved in the routing action.

Modify System Controller Operation

```
#include api_ctrl.h
```

Configure VCA/Set System Clock (\$C0 00) Command Buildup

This function does the following:

- Generates the Configure VCA/Set System Clock (\$C0 00) command that is used to configure any specific Virtual Communication Addresses (VCA) needed besides global communication address \$DF. Specific VCAs are especially important when a single host controls multiple systems.
- Allows setting of the system real time clock.

Function

```
fcc_confg_vca(api_message *msg, char *buf)
```

Mode Constant

None

Parameters

The parameters for this message are in Table 2-28.

Table 2-28 Configure VCA/Set System Clock (\$C0 00) Command Parameters

Parameter	Type	Description
funcid2	unchar	Second function ID byte.
vca	unchar	Virtual communications address; specifying \$FF leaves it unchanged address to \$DF.
hour_clock	unchar	System clock hour value to set.
min_clock	unchar	System clock minute value to set.
sec_clock	unchar	System clock second value to set.

Change Active Controllers (\$C0 01) Command Buildup

This function generates the Change Active Controllers (\$C0 01) command that is sent to the system. In redundant systems, it transfers system control from the active side to the standby side. Command can be sent to either the active or standby system controller. Optional reset of previously active side is also available.

Function

```
fcc_ch_sysctrl(api_message *msg, char *buf)
```

Mode Constant

None

Parameters

The parameters for this message are in Table 2-29.

Table 2-29 Change Active Controllers (\$C0 01) Command Parameters

Parameter	Type	Description
funcid2	uchar	Second function ID byte.
prev_act_cond	uchar	Condition of transfer for previously active side. Can be NORESET_PREVACT or RESET_PREVACT.

Synchronization Control (\$C0 02) Command and Report

These functions generate the T1 Synchronization Control (\$C0 02) command that is sent to the system, and parses the report returned from the system. It alters Administration Console Master Timing Link parameters as follows:

- Switch to internal synchronization.
- Switch to external synchronization.
- Switch to incoming synchronization and specify/change Master Timing Link.
- Primary only.
- Secondary only.
- Both primary and secondary.

**Note**

T1, E1 and ISDN PRI cards can be selected as the source for incoming timing.

Function

```
fcc_config_t1(api_message *msg, char *buf)
frc_config_t1(uchar *buf, int len, api_message *msg)
```

Mode Constant

CC_CONFIG_T1

Parameters

The parameters for this message are in Table 2-30.

Table 2-30 T1 Synchronization Control (\$C0 02) Command Parameters

Parameter	Type	Description
funcid2	uchar	Second function ID byte.
t1_sync_mode	uchar	Specifies Master Timing Link parameters to be altered. Can be INT_SYNC, EXT_SYNC, PR_MASTER PR_SEC_MASTER or SEC_MASTER.
prim_link_val	Paddr	Port address if t1_sync_mode = PR_SEC_MASTER or PR_MASTER.
sec_link_val	Paddr	Port address if t1_sync_mode = PR_SEC_MASTER or SEC_MASTER.

Set/Reset Host Alarms (\$C0 03) Command Buildup

This function generates the Set/Reset Host Alarms (\$C0 03) command that is sent to the system. Use it to set/clear major, minor and auxiliary alarms controlled by the host.

Function

```
fcc_set_alarms(api_message *msg, char *buf)
```

Mode Constant

None

Parameters

The parameters for this message are in Table 2-31.

Table 2-31 Set/Reset Host Alarms (C0 03) Command Parameters

Parameter	Type	Description
funcid2	uchar	Second function ID byte.
alarm_clear	unsigned:4	Determines if alarm is to be set or cleared. Can be ALARM_SET or ALARM_CLEAR.
alarm_type	unsigned:4	Type of alarm to be set/reset. Can be MAJOR_ALARM, MINOR_ALARM, AUX1_ALARM or AUX2_ALARM.

Host Call Load Control (\$C0 04) Command Buildup

This function generates the Host Call Load Control (\$C0 04) command that is sent to the system. Use it in conjunction with Enable Host Control of Call Load feature. When enabled, this command allows the host to start or stop sending Inpulse Rule Complete (\$DD) reports and Incoming Port Change of State (\$DB) reports. This effectively stops call processing, since calls cannot be completed through the system without host intervention.

Function

```
fcc_load_ctrl(api_message *msg, char *buf)
```

Mode Constant

None

Parameters

The parameters for this message are in Table 2-32.

Table 2-32 Host Call Load Control (\$C0 04) Command Parameters

Parameter	Type	Description
funcid2	uchar	Second function ID byte.
is_rep_suspend	unsigned:1	Determines if system should process incoming calls by reporting them to host.

Host Assume/Relinquish Port Control (\$C0 05) Command

These functions extend the system controller functionality to allow a host process to relinquish control of a call assigned to itself, or assume control of a call that has been assigned to a different host.

Function

```
fcc_port_ctrl (api_message *msg, char *buf)
frc_port_ctrl(uchar *buf, int len, api_message *msg)
```

Mode Constant

CC_PORT_CTRL

Parameters

The parameters for this message are in Table 2-33.

Table 2-33 Host Assume/Relinquish Port Control (\$C0 05) Command Parameters

Parameter	Type	Description
funcid2	uchar	Always 0x05.
port_ctrl	uchar	Port control modifier.
port	Paddr	Extended message mode.

Active/Standby Mode (\$DC) Report Parsing

This function does the following:

- Analyzes Active/Standby mode (\$DC) report sent from the system.
- Informs host of a system boot or transfer in control between the active and standby sides for a redundant system.
- Reports when a link between the host and the system becomes established. In a redundant system, each link sends the report.

Function

```
frc_act_sby(unchar *buf, int len, api_message *msg)
```

Mode Constant

None

Parameters

The parameters for this message are in Table 2-34.

Table 2-34 Active/Standby Mode (\$DC) Report Parameters

Parameter	Type	Description
funcid2	unchar	Second function ID byte.
is_sysboot	unsigned:1	Has system booted/data link established or is it a run time transfer?
is_sds_b	unsigned:1	Has report originated from the A-side of the system or the B-side of the system?
is_sds_act	unsigned:1	Is the system originating the report currently active or standby?

System Status

```
#include api_stat.h
```

Request Resource Allocation (\$80) Command Buildup

This function generates the Request Resource Allocation (\$80) command that is sent to the system. The system returns a bit map of the on-line and off-line status of every port within a given resource group and port address range. The add_range can be RANGE_00, RANGE_512, RANGE_1024, RANGE_1536, RANGE_2048, RANGE_2560, RANGE_3072, or RANGE_3584.

Function

```
fcs_res_alloc(api_message *msg, char *buf)
```

Mode Constant

CS_RES_ALLOC

Parameters

The parameters for this message are in Table 2-35.

Table 2-35 Request Resource Allocation (\$80) Command Parameters

Parameter	Type	Description
add_range	ulong	Port address range for which resource allocation report desired. Can be RANGE_00, RANGE_512, RANGE_1024, RANGE_1536, RANGE_2048, RANGE_2560, RANGE_3072, or RANGE_3584.
res_grp_no	Group	Resource group number.

Resource Allocation (\$80) Report Parsing

This function analyzes the Resource Allocation (\$80) report sent from the system. A bit map is returned containing the status of each port within a given range and specified resource group. The report shows whether the port is of the specified resource group and whether it is on-line or off-line.

Function

```
frs_res_alloc(unchar *buf, int len, api_message *msg)
```

Mode Constant

```
RS_RES_ALLOC
```

Parameters

The parameters for this message are in Table 2-36.

Table 2-36 Resource Allocation (\$80) Report Parameters

Parameter	Type	Description
low_port	Paddr	Lower address range.
high_port	Paddr	Higher address range.
res_grp	Group	Resource group number.
status	char array	128-byte array containing the status byte.

Request Hardware Allocation (\$81) Command Buildup

This function generates the Request Hardware Allocation (\$81) command that is sent to the system. The resulting bit map allows the host to match logical port addresses used in system commands to their corresponding physical rack, level, slot (R-L-S) hardware addresses of all ports within a given port address range. The add_range can be HW_RANGE_00, HW_RANGE_512, HW_RANGE_1024, HW_RANGE_1536, HW_RANGE_2048, HW_RANGE_2560, HW_RANGE_3072, or HW_RANGE_3584.

Function

```
fcs_hw_alloc(api_message *msg, char *buf)
```

Mode Constant

CS_HW_ALLOC

Parameter

The parameter for this message is in Table 2-37.

Table 2-37 Request Hardware Allocation (\$81) Command Parameters

Parameter	Type	Description
add_range	uchar	Port address range for which resource allocation report desired.

Hardware Allocation (\$81) Report Parsing

This function analyzes the Hardware Allocation (\$81) report sent from the system and returns a bit map of ports within a given range. Each byte in the 64-byte array represents eight port addresses.

Function

```
frs_hw_alloc(uchar *buf, int len, api_message *msg)
```

Mode Constant

RS_HW_ALLOC

Parameters

The parameters for this message are in Table 2-38.

Table 2-38 Hardware Allocation (\$81) Report Parameters

Parameter	Type	Description
low_port	Paddr	Lower address range.
high_port	Paddr	Higher address range.
rls	char array	64-byte array containing R-L-S information.

Card Status (\$82) Command Buildup

Use this function to obtain the Card Status (\$82) report. You can specify a single card or a range of cards. Specify the card by its rack, level, and slot (R-L-S) position. Specify a range of cards by encoding the starting R-L-S and ending R-L-S in the command. One Card Status (\$82) report is generated for each card specified in the range. For single cards, the starting R-L-S and the ending R-L-S are the same.

Function

```
fcs_card_statreq(api_message *msg, char *buf)
```

Mode Constant

CS_CARD_STATREQ

Parameters

The parameters for this message are in Table 2-39.

Table 2-39 Card Status (\$82) Command Parameters

Parameter	Type	Description
first_rack	unchar	First rack number.
first_level	unchar	First level number.
first_slot	unchar	First slot number.
first_span	unchar	First span number.
last_rack	unchar	Last rack number.
last_level	unchar	Last level number.
last_slot	unchar	Last slot number.
last_span	unchar	Last span number.

Card Status (\$82) Report Parsing

This function informs the host of the status of a card. The card location is represented by both the port address and the physical rack, level, and slot (RLS) address. The report includes the status of the card and the type of the card. One Card Status (\$82) report is generated for each of the cards specified in the Card Status (\$82) command. In the case of a multispan card, an \$82 report is generated for each span in the slot. The Card Status (\$82) report takes the form of a command returned with a network status byte set to \$01.

Function

```
frs_card_statreq(api_message *msg, char *buf)
```

Mode Constant

```
RS_CARD_STATREQ
```

Parameters

The parameters for this message are in Table 2-40.

Table 2-40 Card Status Report (\$82) Parameters

Parameter	Type	Description
low_port	Paddr	Lower address range.
high_port	Paddr	Higher address range.
hour	unchar	Hour.
min	unchar	Minute.
sec	unchar	Second.
rack	unchar	Rack number.
level	unchar	Level number.

Table 2-40 Card Status Report (\$82) Parameters (continued)

Parameter	Type	Description
slot	unchar	Slot number.
span	unchar	Span number.
card_type	unchar	Card type.
card_state	unchar	Card state.

Port Status (\$83) Command Buildup

This function is used to obtain the Port Status (\$83) report. The command queries the status of a range of ports specified by either a port address or the rack, level, and slot (RLS) encoding of a card, with the specified span or resource group. In the case of a query for ports in a multispan card (through RLS specification), you must also specify the span (interface).

Function

```
fcs_port_statreq(api_message *msg, char *buf)
```

Mode Constant

```
CS_PORT_STATREQ
```

Parameters

The parameters for this message are in Table 2-41.

Table 2-41 Port Status (\$83) Command Parameters

Parameter	Type	Description
more_frags	unsigned	More fragments to follow?
align1	unsigned:3	Unused—for alignment purposes.
pa_range	unsigned:1	Is port address range used?
rg	unsigned:1	Is a resource group used?
rls	unsigned:1	Is rack, level, slot, span used?
align2	unsigned:1	Unused—for alignment purposes.
If pa_range = TRUE:		
start_pa	Paddr	Starting port address.
end_pa	Paddr	Ending port address.
If rg = TRUE:		
align:1	ushort	Unused—for alignment purposes.
rg_num	ushort	Resource group number
align:2	Paddr	Unused—for alignment purposes.
If rls = TRUE:		
rack	unchar	Rack number.

Table 2-41 Port Status (\$83) Command Parameters (continued)

Parameter	Type	Description
level	uchar	Level number.
slot	uchar	Slot number.
span	uchar	Span number.
align1	Paddr	Unused—for alignment purposes.

Port Status (\$83) Report Parsing

This function informs the host of a range of ports. The command, for which the report is generated, forms the leading portion of the report. This leading portion is followed by a series of port status report elements, each of which is three bytes long. The first two bytes specify the port address; the third byte specifies the call processing status of the port. If the port range for which the status report is requested is such that one \$83 report cannot accommodate all the port status report elements, the report is split into as many \$83 reports as necessary. Such fragments are distinguished from each other through a continuity bit. Up to 82 port status elements can be in one \$83 report, considering that the maximum length of the report is 256 bytes.

Function

```
(f)rs_port_statreq(api_message *msg, char *buf)
```

Mode Constant

```
RS_PORT_STATREQ
```

Parameters

The parameters for this message are in Table 2-42 and Table 2-43.

Table 2-42 Port Status (\$83) Report Parameters

Parameter	Type	Description
more_frags	unsigned:1	More fragments to follow?
align1	unsigned:3	Unused—for alignment purposes.
pa_range	unsigned:1	Is port address range used?
rg	unsigned:1	Is a resource group used?
rls	unsigned:1	Is rack, level, slot, span used?
align2	unsigned:1	Unused—for alignment purposes.
if pa_range = TRUE		
start_pa	Paddr	Starting port address.
end_pa	Paddr	Ending port address.
if rg = TRUE		
align1	ushort	Unused—for alignment purposes.
rg_num	ushort	Resource group number.

Table 2-42 Port Status (\$83) Report Parameters (continued)

Parameter	Type	Description
align2	Paddr	Unused—for alignment purposes.
if rls = TRUE		
rack	unchar	Rack number.
level	unchar	Level number.
slot	unchar	Slot number.
span	unchar	Span number.
align1	Paddr	Unused—for alignment purposes.
report_element[]	rep_ele_struct	Report Elements (MAX_REPORT_ELEMENTS). See Table 2-43 .

Table 2-43 Port Status (\$83) Report Elements

Parameter	Type	Description
port_address	Paddr	Port address.
cp_state	unchar	Call processing state.
supp_state	unchar	Supplementary state.
isdn_state	unchar	ISDN state.
isdn_substate	unchar	ISDN sub state.

Permanent Signal Condition (\$D2) Report Parsing

This function analyzes the Permanent Signal Condition (\$D2) report sent from the system. It informs a host that a line/trunk port has not released within 30 seconds of a release by the system. The report is also sent when a line/trunk goes back on hook. This function analyzes the \$D2 report sent from the system.

Function

```
frr_psc(unchar *buf, int len, api_message *msg)
```

Mode Constant

RR_PSC

Parameters

The parameters for this message are in [Table 2-44](#).

Table 2-44 Permanent Signal Condition (\$D2) Report Parameters

Parameter	Type	Description
port	Paddr	Port address.
is_on_hook	unsigned:1	Is PSC due to on hook on one end of stable call?

Table 2-44 Permanent Signal Condition (\$D2) Report Parameters (continued)

Parameter	Type	Description
is_error	unsigned:1	Is PSC due to error condition?
did_host_tout	unsigned:1	Was PSC caused when host timed out because of no response to initial call report?
did_host_dcon	unsigned:1	Was PSC caused when the host forcibly disconnected the port?
no_mf_rcvrs	unsigned:1	Is PSC caused because of MF receiver resource limitation?
is_sds_prob	unsigned:1	Is PSC due to an internal system problem?
is_mf_garbled	unsigned:1	Is PSC due to garbled MF digits?
is_psc	unsigned:1	Does PSC exist?
res_grp	Group	Resource group number.
hour	unchar	Hour of report generation.
min	unchar	Minute of report generation.
sec	unchar	Second of report generation.

Change Port Status (\$90) Command and Report

These functions generate and parse the Change Port Status (\$90) command. They enable the host to activate and deactivate individual ports on an internal service circuit, or network interface card. They also perform the same action as taking ports out-of-service using the system administration Card Maintenance utility.

Function

```
fcm_ch_pstatus(api_message *msg, char *buf)
frm_ch_pstatus(unchar *buf, int len, opi_message *msg)
```

Mode Constant

CM_CH_PSTATUS

Parameters

The parameters for this message are in Table 2-45.

Table 2-45 Change Port Status (\$90) Command and Report Parameters

Parameter	Type	Description
spacer_byte	SpacerByte	User-definable spacer bytes—compiled in extended mode only.
port_deact	unsigned:2	Conditions to activate/deactivate a port.
is_port_seize	unsigned:1	Seize out on a port.
is_upd_disk	unsigned:1	Is port status change to be updated on a disk?
list_ports	unsigned:2	Change status of single port, cluster of ports or ports listed in the port address map.

Table 2-45 Change Port Status (\$90) Command and Report Parameters

Parameter	Type	Description
first_port	Paddr	Port address of the first port affected by change of state.
paddr_map	array, unchar	Port address map.

Port Status (\$D3) Report Parsing

This function analyzes the Port Status (\$D3) report sent from the system. It informs the host of a change in status of an system resource report. Change can be as a result of the following:

- Activation/deactivation of a port through system administration.
- Setting a voice path between ports through system administrative menu.
- Path/Port Reset screen.
- Inward seize detected for a port with COS=O or COS=2 and internal COS=U; port busied out by connected equipment.
- Auto Makebusy feature; port busied out after specified number of supervision errors have been detected.

Function

```
frs_port_status(unchar *buf, int len, api_message *msg)
```

Mode Constant

```
RS_PORT_STATUS
```

Parameters

The parameters for this message are in Table 2-46.

Table 2-46 Port Status (\$D3) Report Parameters

Parameter	Type	Description
port	Paddr	Address of port.
old_status	unsigned:4	Old status of port before change occurred. Can be either RES_UNAVAIL or RES_AVAIL.
new_status	unsigned:4	New status of port before change occurred. Can be either RES_UNAVAIL or RES_AVAIL.
originator	unchar	Specifies whether the system or host originated the change in status and the reason. Can be HOST_UNK, SDS_UNK, SDS_SYSADM, SDS_SPATH, SDS_DEND, SDS_ABUSY or SDS_CARD.
res_grp	Group	Resource group number.
hour	unchar	Hour of report generation.
min	unchar	Minute of report generation.
sec	unchar	Second of report generation.

Resource Limitation (\$D6) Report Parsing

This function analyzes the Resource Limitation (\$D6) report sent from the system. It informs the host when a resource limitation condition is detected or cleared. It is generated only for the first occurrence in a specific group for a limitation condition (in response to a resource control command, impulse rule, outpulse rule) until the condition clears.

Function

```
frs_res_limit(unchar *buf, int len, api_message *msg)
```

Mode Constant

```
RS_RES_LIMIT
```

Parameters

The parameters for this message are in Table 2-47.

Table 2-47 Resource Limitation (\$D6) Report Parameters

Parameter	Type	Description
is_limitation	unsigned:1	Is resource limitation present?
res_grp	Group	Resource group number.
port	Paddr	Requesting port address.
hour	unchar	Hour of report generation.
min	unchar	Minute of report generation.
sec	unchar	Second of report generation.

Card Status (\$D9) Report Parsing

This function analyzes the Card Status (\$D9) report sent from the system. It informs the host of a change in the status of an system resource card. The card location is represented both by the port address and the physical rack-level-slot address. It is generated when status changes are caused by system administration, host command, or physical removal/replacement of a card.

Function

```
frs_card_status(unchar *buf, int len, api_message *msg)
```

Mode Constant

```
RS_CARD_STATUS
```

Parameters

The parameters for this message are in Table 2-48.

Table 2-48 Card Status (\$D9) Report Parameters

Parameter	Type	Description
low_port	Paddr	Address of lowest port affected.
high_port	Paddr	Address of highest port affected.
is_card_off	unsigned:1	Is the card offline?
is_card_del	unsigned:1	Has the card been deleted?
is_card_add	unsigned:1	Has the card been added?
is_sds_resp	unsigned:1	Is system responsible?
is_host_resp	unsigned:1	Is the host responsible?
hour	unchar	Hour of report generation.
min	unchar	Minute of report generation.
sec	unchar	Second of report generation.
card_state	unchar	State of the card. Can be either UNK_STATE,ACT_STATE,MAINT_STATE,DIAG_STATE,OOS_STATE,SBY_STATE (only BRC and DTG) or CON_STATE.
rack	unchar	Rack number.
level	unchar	Level number.
slot	unchar	Slot number.

Alarm Condition (\$F0) Report Parsing

This function performs the following actions:

- Translates the system \$F0 report, pointed to by **buf**, into the Alarm Condition API message, pointed to by **msgp**.
- Provides the user with type of alarm that occurred, the level of severity of that alarm, and the number of alarms that exist.

Function

```
frs_alarm_cond(unchar *buf, int len, api_message *msgp)
```

Mode Constant

None

Parameters

The parameters for this message are in Table 2-49.

Table 2-49 Alarm Condition (\$F0) Report Parameters

Parameter	Type	Description
alarm_type	unsigned integer	Identifies the type of alarm that occurred.
alarm_level	unsigned integer	Indicates the level of severity.
alarm_count	integer	Indicates the number of alarms that exist.
count_level	unsigned integer	Indicates the level of severity for the number of alarms.
almdata_ind	integer	Indicates if alarm data is present: TRUE or FALSE.
alarm_data	unsigned integer	Additional data for some alarm types.

Controlling ISDN Primary Rate Interfaces

```
#include "api_dtmf.h"
#include "api_dvc.h"
#include "api_mf.h"
#include "api_src.h"
#include "api_net.h"
#include "api_isdn.h"
```

ISDN Port Control (\$49) Command and Report

These functions perform the following actions:

- Translates the ISDN Port Control API message, pointed to by **msgp**, into the corresponding system \$49 message, pointed to by **buf**, or parse the message returned from the system.
- Allows the user to specify the controlling and associated ports involved with the call.
- Processes up to five ISDN Outpulse Control Segments, any one of which may contain a Binary Code Digit string, Information Element (IE) data, IE header, or a complete IE message.
- Allows the user to specify whether or not to execute an inpulse or outpulse rule.
- Copies IE segments, if any, into the correct location of the system message. The IE segments are created with the **IE_buildCallNum()** function and then copied into the **ie_segs[]** array.
- Returns the length, in bytes, of the \$49 system message pointed to by **buf**.

Function

```
fcr_isdn_ctrl(api_message *msgp, uchar *buf)
frr_isdn_ctrl(uchar *buf, int len, api_message *msg)
```

Mode Constant

```
CR_ISDN_CTRL
```

Parameters

The parameters for this message are in Table 2-50.

Table 2-50 ISDN Port Control (\$49) Command and Report Parameters

Parameter	Type	Description
spacer_byte	SpacerByte	User definable spacer bytes—compiled in extended mode only.
tone_plan	TonePlan	Reserved for tone plans—compiled in extended mode only.
ctrl_port	Paddr	Controlling port address.
ctrl_callid	unsigned integer	Controlling call identifier.
ctrl_addrld	unsigned integer	Controlling address identifier.
assc_port	unsigned integer	Associated port address.
assc_callid	unsigned integer	Associated call identifier.
assc_addrld	unsigned integer	Associated address identifier.
conn_switch_reqd	unsigned integer	Switching action.
conn_port_attach	unsigned integer	Attach or remove resource.
conn_port_rgrp	unsigned integer	Port or resource group.
conn_path	unsigned integer	Speech path control options.
disconn_d_bit	unsigned integer	Receive report on disconnect.
disconn_r_bit	unsigned integer	Receive report on port release.
disconn_t_bit	unsigned integer	Receive report if release was received.
disconn_i_bit	unsigned integer	Return iport to CP_SETUP when oport releases.
disconn_c_bit	unsigned integer	Send EA report if ISDN oport is released.
disconn_u_bit	unsigned integer	Force oport to IDLE or SETUP.
rule_orule_exec	unsigned integer	Is outpulse rule executed?
rule_irule_exec	unsigned integer	Is inpulse rule executed?
rule_number	Rule	Inpulse or outpulse rule number.
no_opulse_segs	unsigned integer	Outpulse segment count.
no_ie_segs	unsigned integer	Information element count.
ie_seg_len	unsigned integer	Length in bytes of the IE segment.
opulse_segs	ISDN_OPULSE_SEGS array	Outpulse segments.
ie_segs	array, unsigned char	IE segments.

ISDN Port Change of State (\$EA) Report Parsing

This function performs the following actions:

- Translates the system \$EA report, pointed to by **buf**, into the ISDN Port Change of State API message, pointed to by **msgp**.

- Informs the user of a network event or change of state on an ISDN PRI port.
- Provides the affected controlling and associated port.
- Provides the ISDN D-channel message type.
- Provides error status.
- Provides the outpulse rule number that was executed.
- Provides the ISDN Supervision Template number.
- Provides optional IE message segments that may have been collected by the outpulse rule. The IE message segments must be processed by the **IE_parseCallNum()** function.

Function

```
frr_isdn_pcos(uchar *buf, int len, api_message *msgp)
```

Mode Constant

```
RR_ISDN_PCOS
```

Parameters

The parameters for this message are in Table 2-51.

Table 2-51 ISDN Port Change of State (\$EA) Report Parameters

Parameter	Type	Description
ctrl_port	Paddr	Controlling port address.
ctrl_callid	unsigned integer	Controlling call identifier.
ctrl_addrld	unsigned integer	Controlling address identifier.
assc_port	Paddr	Associated port address.
assc_callid	unsigned integer	Associated call identifier.
assc_addrld	unsigned integer	Associated address identifier.
change	unsigned integer	Type of change on the port.
d_channel_msg	unsigned integer	Codeset 0, Q.931 D-channel msgs.
supv_answer	unsigned integer	Is outgoing port answered or not?
supv_template	unsigned integer	ISDN supervision template used.
status	unsigned integer	Error status.
rule_number	Rule	Outpulse rule processed.
no_ie_segs	integer	IE count.
ie_seg_len	integer	Length in bytes of the ie segment.
ie_segs	array, unsigned char	IE segments.
data[5]	array, unsigned char	Optional data field.

ISDN Impulse Rule Complete (\$ED) Report Parsing

This function performs the following actions:

- Translates the system \$ED report, pointed to by **buf**, into the ISDN impulse rule complete API message, pointed to by **msgp**.
- Informs the user of the completion of an impulse rule on a ISDN PRI port.
- Provides the affected controlling port.
- Provides the ISDN D-channel message type.
- Provides error status.
- Provides the impulse rule number that was executed.
- Provides optional DTMF and MF digit reports in the same format as the regular Impulse Rule Complete (\$DD) report.
- Provides optional IE message segments that may have been collected by the outpulse rule. The IE message segments must be processed by the **IE_parseCallNum()** function.

Function

```
frr_isdn_irule(unchar *buf, int len, api_message *msgp)
```

Mode Constant

```
RR_ISDN_IRULE
```

Parameters

The parameters for this message are in Table 2-52.

Table 2-52 ISDN Impulse Rule Complete (\$ED) Report Parameters

Parameter	Type	Description
ctrl_port	Paddr	Controlling port address.
ctrl_callid	unsigned integer	Controlling call identifier.
ctrl_addrld	unsigned integer	Controlling address identifier.
rule_number	Rule	Impulse rule executed.
d_channel_msg	unsigned integer	Received D-channel message type.
no_status_bytes	unsigned integer	Number of impulse rule status bytes.
no_digit_segs	integer	Number of digit segment reports.
no_ie_segs	integer	Number of ie messages that follow.
ie_seg_len	integer	Length in bytes of the IE segment.
irule_status	array, unsigned char	Completion status.
digit_segs	array, ipulse_seg_t	Digit segment storage.
ie_segs	array, unsigned char	Information element segments.

