

SXDEV

COLLABORATORS

	TITLE : SXDEV		
ACTION	NAME	DATE	SIGNATURE
WRITTEN BY		August 22, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	SXDEV	1
1.1	Developing Doors	1
1.2	Structures	1
1.3	Door Port	5
1.4	System-X XIM Functions	6
1.5	JH_LI <Func: 000>	7
1.6	JH_REGISTER <Func: 001>	7
1.7	JH_SHUTDOWN <Func: 002>	7
1.8	JH_WRITE <Func: 003>	8
1.9	JH_SM <Func: 004>	8
1.10	JH_PM <Func: 005>	8
1.11	JH_HK <Func: 006>	8
1.12	JH_SG <Func: 007>	9
1.13	JH_SF <Func: 008>	9
1.14	JH_EF <Func: 009>	9
1.15	JH_CO <Func: 010>	10
1.16	JH_BBSNAME <Func: 011>	10
1.17	JH_SYSOP <Func: 012>	10
1.18	JH_FLAGFILE <Func: 013>	10
1.19	DT_NAME <Func: 100>	11
1.20	DT_PASSWORD <Func: 101>	11
1.21	DT_LOCATION <Func: 102>	11
1.22	DT_PHONENUMBER <Func: 103>	12
1.23	DT_SLOTNUMBER <Func: 104>	12
1.24	DT_ACCESSLEVEL <Func: 105>	12
1.25	DT_RATIOTYPE <Func: 106>	12
1.26	DT_RATIO <Func: 107>	12
1.27	DT_COMPTYPE <Func: 108>	13
1.28	DT_MESSAGESPOSTED <Func: 109>	13
1.29	DT_UPLOADS <Func: 110>	13

1.30 DT_DOWNLOADS <Func: 111>	13
1.31 DT_TIMESCALED <Func: 112>	14
1.32 DT_TIMELASTON <Func: 113>	14
1.33 DT_TIMEUSED <Func: 114>	14
1.34 DT_TIMELIMIT <Func: 115>	15
1.35 DT_TIMETOTAL <Func: 116>	15
1.36 DT_BYTESUPLOAD <Func: 117>	15
1.37 DT_BYTEDOWNLOAD <Func: 118>	15
1.38 DT_DAILYBYTELIMIT <Func: 119>	16
1.39 DT_DAILYBYTEDLD <Func: 120>	16
1.40 DT_EXPERT <Func: 121>	16
1.41 DT_LINELENGTH <Func: 122>	16
1.42 ACTIVE_NODES <Func: 123>	16
1.43 DT_DUMP <Func: 124>	17
1.44 DT_TIMEOUT <Func: 125>	17
1.45 BB_CONFNAME <Func: 126>	17
1.46 BB_CONFLOCAL <Func: 127>	17
1.47 BB_LOCAL <Func: 128>	18
1.48 BB_STATUS <Func: 129>	18
1.49 BB_MAINLINE <Func: 131>	18
1.50 RETURNCOMMAND <Func: 136>	18
1.51 ZMODEMSEND <Func: 137>	19
1.52 ZMODEMRECEIVE <Func: 138>	19
1.53 SCREEN_ADDRESS <Func: 139>	19
1.54 BB_TASKPRI <Func: 140>	19
1.55 RAWSCREEN_ADDRESS <Func: 141>	20
1.56 BB_CHATFLAG <Func: 142>	20
1.57 DT_STAMP_LASTON <Func: 143>	20
1.58 DT_STAMP_CTIME <Func: 144>	20
1.59 DT_CURR_TIME <Func: 145>	21
1.60 DT_CONFACCESS <Func: 146>	21
1.61 BB_NODEID <Func: 149>	21
1.62 BB_CALLERSLOG <Func: 150>	21
1.63 BB_UDLOG <Func: 151>	22
1.64 EXPRESS_VERSION <Func: 152>	22
1.65 BB_CHATSET <Func: 162>	22
1.66 ENVSTAT <Func: 163>	22
1.67 NODE_DEVICE <Func: 503>	22
1.68 NODE_UNIT <Func: 504>	23

1.69	NODE_BAUD <Func: 505>	23
1.70	JH_MCI <Func: 507>	23
1.71	PRV_COMMAND <Func: 508>	23
1.72	BB_CONFNUM <Func: 510>	24
1.73	BB_DROPDTR <Func: 511>	24
1.74	BB_GETTASK <Func: 512>	24
1.75	NODE_BAUDRATE <Func: 516>	24
1.76	BB_LOGONTYPE <Func: 517>	24
1.77	BB_SCRLEFT <Func: 518>	25
1.78	BB_SCRTOP <Func: 519>	25
1.79	BB_SCRWIDTH <Func: 520>	25
1.80	BB_SCRHEIGHT <Func: 521>	25
1.81	BB_PURGELIN <Func: 522>	26
1.82	BB_PURGELINESTART <Func: 523>	26
1.83	BB_PURGELINEEND <Func: 524>	26
1.84	BB_NONSTOPTEXT <Func: 525>	26
1.85	BB_LINECOUNT <Func: 526>	26
1.86	DT_LANGUAGE <Func: 527>	27
1.87	DT_QUICKFLAG <Func: 528>	27
1.88	DT_GOODFILE <Func: 529>	27
1.89	System-X Only Print String	28
1.90	Get UserStruct Pointer (UserData)	28
1.91	Get UserStruct Pointer (SXUser)	28
1.92	Get UserStruct Pointer (Index)	28
1.93	Get the System-X version	29
1.94	Get a pointer to the node structures	29
1.95	MCP	29
1.96	Execute an internal SX Function	32

Chapter 1

SXDEV

1.1 Developing Doors

DEVELOPING DOORS AND UTILS FOR SYSTEM-X

Structures associated with SX
 How the door port works in theory
 List of XIM functions available
 Communication with MCP

```

-----
| System-X uses the XIM door port as its primary door port. This means |
| it's compatable with AmiExpress. If you are intending to write a door |
| especially for System-X, use the XIM port, as it gives you the most  |
| power.                                                                |
-----

```

Note: if you find any developer/starter packs around for AmiExpress, they will work just fine with System-X. There are examples for AmigaE and heaps of ASM examples around. Developing doors for SX can also be done using any of the other door types, but XIM is the best for SX (has the most commands available).

1.2 Structures

The structures are all in 'C' format, but in general:

WORD, short = 16 bit signed word
 UWORD = 16 bit unsigned word
 int, long, LONG = 32 bit signed word
 ULONG = 32 bit unsigned word
 APTR = 32 bit pointer

Any fields with an asterix "*" infront of them are pointers!

```

-----
| User.data or User structure in memory |

```

```

\-----/

struct UserData {
    char  Name[31],
        Pass[9],
        Location[30],
        PhoneNumber[13];
    USHORT Slot_Number;
    USHORT Sec_Status, /* access level */
        Sec_Board, /* unused */
        Sec_Library, /* unused */
        Sec_Bulletin, /* unused */
        Messages_Posted;
    ULONG NewSinceDate,
        ConfRead1,
        ConfRead2,
        ConfRead3,
        ConfRead4,
        ConfRead5,
        ConfRead6,
        ConfRead7,
        ConfRead8,
        ConfRead9;
    char  Conference_Access[10];
    USHORT Uploads,
        Downloads,
        ConfRJoin,
        Times_Called;
    long  Time_Last_On,
        Time_Used,
        Time_Limit,
        Time_Left;
    ULONG Bytes_Download,
        Bytes_Upload,
        Daily_Bytes_Limit,
        Daily_Bytes_Dld;
    char  Expert;
    ULONG ConfYM1,
        ConfYM2,
        ConfYM3,
        ConfYM4,
        ConfYM5,
        ConfYM6,
        ConfYM7,
        ConfYM8,
        ConfYM9;
    long  BeginLogCall;
    UBYTE Protocol,
        UUCPA,
        LineLength,
        New_User;
};

\-----/
| User.Index or UserIndex structure in memory |
\-----/

```

```
struct UserIndexStruct
{
    char handle[31];
    char realname[31];
    UWORD misc;
};
```

```
.------.
| User.SX file or in memory |
\-----/
```

```
struct SXUserStruct
{
    UWORD byteratio;
    UWORD fileratio;
    ULONG flags;
    UWORD freefiles;
    ULONG freebytes;
    ULONG ConfAccess[10];
    UWORD lastfilearea;
    char computer[24];
    char sentbyline[46];
    char password[16];
    long firstcall;
    char reserved[110];
};
```

```
.------.
| Prefs/Confs.DAT |
\-----/
```

```
struct ConfStruct
{
    char name[45];
    char path[55];
    char pass[16];
    char filepath[52];
    UWORD fileareas;
    UWORD uploadarea;
    UBYTE flf;
    UBYTE flags;
    char reserve[82];
};
```

```
.------.
| Prefs/Serial.DAT |
\-----/
```

```
struct SerialStruct
{
    char device[32];
    WORD unit;
    WORD misc;
    long minrate;
```



```

    long dcerate;
    UBYTE sevenwire;
    UBYTE shared;
    char ring[16];
    char connect[24];
    char answer[16];
    char initstr[48];
};

```

```

.------.
| SX:Prefs/Main.DAT |
\-----/

```

```

struct MainStruct
{
    char BBSName[64];
    char BBSPath[64];
    char BBSLoc[64];
    char Sysop[64];
    char DNPath[64];
    char ULPath[64];
    long nodes;
};

```

```

.------.
| SX:LogFiles/Download.LOG & Upload.LOG |
\-----/

```

```

struct XferLog
{
    UWORD user_slot;
    UBYTE conf,
        filearea;
    char filename[32];
    long size,
        baud,
        cps,
        time;
    UBYTE node;
    char res[7];
} Xfer;

```

```

.------.
| SX:LogFiles/Callers.LOG |
\-----/

```

Note: this file has a UWORD at the top before the first structure, which is a pointer to the oldest entry in the log. (its a rotary log of 30)

```

struct CallerLog
{
    UWORD user_slot,
        node;
    long time_login,
        time_logout,

```

```

    seconds_online,
    bytes_uploaded,
    bytes_downloaded;
UWORD files_uploaded,
    files_downloaded,
    messages;
long baud,
    flags;
UBYTE logout_mode;
char res[25];
};

```

```

-----
| NODE STRUCTURE AS PASSED BY MCP-COMMAND-14 OR XIM-COMMAND-1505 |
-----

```

```

struct node_struct
{
    /* offset DEC */
    APTR next;          /* 0 */
    struct UserData *User;      /* 4 */
    struct UserIndexStruct *UserIndex; /* 8 */
    struct SXUserStruct *SXUser; /* 12 */
    char *action;          /* 16 */
    char *filename;        /* 20 */
    long baud;             /* 24 */
    long loginsecs;        /* 28 */
    UWORD number;          /* 32 */
    UBYTE actionnumber;     /* 34 */
    UBYTE active;          /* 35 */
    UBYTE useron;          /* 36 */
    UBYTE misc;            /* 37 */
};

```

1.3 Door Port

There are two ways to program doors for System-X. One is to use AEDoor.library, which is very easy, and the other is to use exec messages, which gives more control. Below is explained the exec message style of programming doors.

--- The message structure ---

```

struct JHMessage
{
    struct Message Msg;    <----- msg structure
    char String[200];      <----- info buffer
    int Data;              <----- Read/Write & result indicator
    int Command;           <----- Command sent from door.
    int NODEID;            <----- reserved
    int LineNum;           <----- reserved
    unsigned long signal;  <----- reserved
    struct Process *task;
    APTR *Semi;

```

```
};
```

The port name is AEDoorPort%d , where %d is the node number. General in 'C' you'd do:

```
-----
char portname[32];
struct MsgPort *port;

sprintf(portname, "AEDoorPort%d", atoi(argv[1]));
port = FindPort(portname);
-----
```

See the examples/ directory for more information.

1.4 System-X XIM Functions

The following list is a list of all System-X HOST Commands which can be used in every program language to get/store information to the AmiExpress Host Port. With this Command you can get Information like BBSNAME, USERNAME etc. Click on the Buttons for more Information.

JH_LI	<Func: 000>	JH_REGISTER	<Func: 001>
JH_SHUTDOWN	<Func: 002>	JH_WRITE	<Func: 003>
JH_SM	<Func: 004>	JH_PM	<Func: 005>
JH_HK	<Func: 006>	JH_SG	<Func: 007>
JH_SF	<Func: 008>	JH_EF	<Func: 009>
JH_CO	<Func: 010>	JH_BBSNAME	<Func: 011>
JH_SYSOP	<Func: 012>	JH_FLAGFILE	<Func: 013>
DT_NAME	<Func: 100>	DT_PASSWORD	<Func: 101>
DT_LOCATION	<Func: 102>	DT_PHONENUMBER	<Func: 103>
DT_SLOTNUMBER	<Func: 104>	DT_ACCESSLEVEL	<Func: 105>
DT_RATIOTYPE	<Func: 106>	DT_RATIO	<Func: 107>
DT_COMPTYPE	<Func: 108>	DT_MESSAGESPOSTED	<Func: 109>
DT_UPLOADS	<Func: 110>	DT_DOWNLOADS	<Func: 111>
DT_TIMESCALED	<Func: 112>	DT_TIMELASTON	<Func: 113>
DT_TIMEUSED	<Func: 114>	DT_TIMELIMIT	<Func: 115>
DT_TIMETOTAL	<Func: 116>	DT_BYTESUPLOAD	<Func: 117>
DT_BYTEDOWNLOAD	<Func: 118>	DT_DAILYBYTELIMIT	<Func: 119>
DT_DAILYBYTEDLD	<Func: 120>	DT_EXPERT	<Func: 121>
DT_LINELENGTH	<Func: 122>	ACTIVE_NODES	<Func: 123>
DT_DUMP	<Func: 124>	DT_TIMEOUT	<Func: 125>
BB_CONFNAME	<Func: 126>	BB_CONFLOCAL	<Func: 127>
BB_LOCAL	<Func: 128>	BB_STATUS	<Func: 129>
BB_MAINLINE	<Func: 131>	RETURNCOMMAND	<Func: 136>
ZMODEMSEND	<Func: 137>	ZMODEMRECEIVE	<Func: 138>
SCREEN_ADDRESS	<Func: 139>	BB_TASKPRI	<Func: 140>
RAWSCREEN_ADDRESS	<Func: 141>	BB_CHATFLAG	<Func: 142>
DT_STAMP_LASTON	<Func: 143>	DT_STAMP_CTIME	<Func: 144>
DT_CURR_TIME	<Func: 145>	DT_CONFACCESS	<Func: 146>
BB_NODEID	<Func: 149>	BB_CALLERSLOG	<Func: 150>
BB_UDLOG	<Func: 151>	EXPRESS_VERSION	<Func: 152>

BB_CHATSET	<Func: 162>	ENVSTAT	<Func: 163>
NODE_DEVICE	<Func: 503>	NODE_UNIT	<Func: 504>
NODE_BAUD	<Func: 505>	JH_MCI	<Func: 507>
PRV_COMMAND	<Func: 508>	BB_CONFNUM	<Func: 510>
BB_DROPDTR	<Func: 511>	BB_GETTASK	<Func: 512>
NODE_BAUDRATE	<Func: 516>	BB_LOGONTYPE	<Func: 517>
BB_SCRLEFT	<Func: 518>	BB_SCRTOP	<Func: 519>
BB_SCRWIDTH	<Func: 520>	BB_SCRHEIGHT	<Func: 521>
BB_PURGELIN	<Func: 522>	BB_PURGELINESTART	<Func: 523>
BB_PURGELINEEND	<Func: 524>	BB_NONSTOPTEXT	<Func: 525>
BB_LINECOUNT	<Func: 526>	DT_LANGUAGE	<Func: 527>
DT_QUICKFLAG	<Func: 528>	DT_GOODFILE	<Func: 529>

Below are functions that are System-X specific and DO NOT work under AmiExpress or any AmiExpress clones.

SX_PS	<Func: 1500>	SX_USERPO	<Func: 1501>
SX_USERPO2	<Func: 1502>	SX_USERPO3	<Func: 1503>
SX_VER	<Func: 1504>	SX_NODES	<Func: 1505>
SX_FUNCTION	<Func: 1506>		

1.5 JH_LI <Func: 000>

JH_LI Requests a string of information from the user with a default string.

```

msg->Command = 0
msg->String  = default result string
msg->Data    = Maximum length of response.

msg->Data will be set to a -1 if a loss carrier or console
TIMEOUT occurs, otherwise MSG->Data will be 1

msg->String will be the response string from the user.
(FUNCTION #: 0 )

```

1.6 JH_REGISTER <Func: 001>

JH_REGISTER Registers a door or XIM with the current node.

```

msg->Command = 1

This must be the first command issued to the express node.
This increments the number of doors active for the current
node.
(FUNCTION #: 1 )

```

1.7 JH_SHUTDOWN <Func: 002>

JH_SHUTDOWN Tells the node that a door is shutting down, this decreases the number of active doors indicator , which once at 0, the AEDoorPort will close.

```
msg->Command = 2
msg->Data     = N/A
```

(FUNCTION #: 2)

1.8 JH_WRITE <Func: 003>

JH_WRITE Allows you to send a text string to the user.

```
msg->Command = 3
msg->String   = text
msg->Data     = N/A
```

(FUNCTION #: 3)

1.9 JH_SM <Func: 004>

JH_SM Allows you to send a text string to the user.

```
msg->Command = 4
msg->String   = text
msg->Data     = 1 or 0
```

if msg->Data = 1, then a CR/LF combination will be sent.

(FUNCTION #: 4)

1.10 JH_PM <Func: 005>

JH_PM Allows you to prompt the user for a specified number of characters.

```
msg->Command = 5
msg->String   = prompt string
msg->Data     = maximum response length.
```

if msg->Data = -1, then a loss carrier has occurred or a TIMEOUT condition has occurred, otherwise msg->Data = 1.

msg->String will be the user response.

(FUNCTION #: 5)

1.11 JH_HK <Func: 006>

JH_HK Allows you to get a 1 character response from the user.

```
msg->Command = 6
msg->String   = text
msg->Data     = N/A
```

if msg->Data = -1, then a loss carrier has occurred or a
TIMEOUT condition has occurred, otherwise msg->Data = 1.

msg->String will be the result string.

(FUNCTION #: 6)

1.12 JH_SG <Func: 007>

JH_SG Allows you to display a text file to the user.

```
msg->Command = 7
msg->String   = part file name.
msg->Data     = N/A
```

ie:

```
msg->String = "BBS:Node1/Bull
```

This would try to display BBS:Node1/BULL.TXT
also takes into account language specifications.

This also searches for the access level patterns, ie:

```
Bull10.TXT, Bull100.TXT
```

(FUNCTION #: 7)

1.13 JH_SF <Func: 008>

JH_SF Allows you to display a text file to the user.

```
msg->Command = 8
msg->String   = Complete pathname
msg->Data     = N/A
```

ie:

```
msg->String = "BBS:Node1/BULL.TXT"
```

This would show the file if it exists.

(FUNCTION #: 8)

1.14 JH_EF <Func: 009>

JH_EF Allows you to use the internal msgbase editor to edit your own files.

 msg->Command = 9
 msg->String = Complete pathname
 msg->Data = 0

 if msg->Data = -1, then a loss carrier has occurred or a TIMEOUT has occurred, otherwise msg->Data will be 1.

(FUNCTION #: 9)

1.15 JH_CO <Func: 010>

JH_CO Allows you to send text string to the console only.

 msg->Command = 10
 msg->String = text
 msg->Data = 1 or 0

 if msg->Data = 1, then a CR/LF combination will be sent in addition to the text.

(FUNCTION #: 10)

1.16 JH_BBSNAME <Func: 011>

JH_BBSNAME Allows you to retrieve the BBS Name.

 msg->Command = 11
 msg->Data = N/A

 msg->String will be the BBS name.

(FUNCTION #: 11)

1.17 JH_SYSOP <Func: 012>

JH_SYSOP Allows you to retrieve the Sysop's Name.

 msg->Command = 12
 msg->Data = N/A

 msg->String will be the Sysop name.

(FUNCTION #: 12)

1.18 JH_FLAGFILE <Func: 013>

JH_FLAGFILE Allows you to add files to the list of flagged files.

msg->Command = 13
msg->String = FileName
msg->Data = N/A

Adds the msg->String to the list of flagged files for
downloading purposes,

NOTE: The files must be in the download path for this to
work.

(FUNCTION #: 13)

1.19 DT_NAME <Func: 100>

DT_NAME Allows you to retrieve or change users name/handle

msg->Command = 100
msg->Data = 1 or 0

if msg->Data = 1, then msg->String will be the name.
if msg->Data = 0, then name will be msg->String

(FUNCTION #: 100)

1.20 DT_PASSWORD <Func: 101>

DT_PASSWORD Allows you to retrieve or change users password

msg->Command = 101
msg->Data = 1 or 0

if msg->Data = 1, then msg->String will be the password.
if msg->Data = 0, then the password will be msg->String.

(FUNCTION #: 101)

1.21 DT_LOCATION <Func: 102>

DT_LOCATION Allows you to retrieve or change users location

msg->Command = 102
msg->Data = 1 or 0

if msg->Data = 1, then msg->String will be the location
if msg->Data = 0, then the location will be msg->String

(FUNCTION #: 102)

1.22 DT_PHONENUMBER <Func: 103>

DT_PHONENUMBER Allows you to retrieve or change users phone number.

```
msg->Command = 103
msg->Data     = 1 or 0
```

```
if msg->Data = 1, then msg->String will be the phonenumbr
if msg->Data = 0, then phonenumbr will be msg->String
```

(FUNCTION #: 103)

1.23 DT_SLOTNUMBER <Func: 104>

DT_SLOTNUMBER Allows you to retrieve users slot number

```
msg->Command = 104
msg->Data     = 1
```

```
if msg->Data = 1, then msg->String will be the SLOTNUMBER.
```

(FUNCTION #: 104)

1.24 DT_ACCESSLEVEL <Func: 105>

DT_ACCESSLEVEL Allows you to retrieve or change users access level.

```
msg->Command = 105
msg->Data     = 1 or 0
```

```
if msg->Data = 1, then msg->String will be ACCESSLEVEL.
if msg->Data = 0, then ACCESSLEVEL will be msg->String.
```

(FUNCTION #: 105)

1.25 DT_RATATYPE <Func: 106>

DT_RATATYPE Allows you to retrieve or change users RatioType

```
msg->Command = 106
msg->Data     = 1 or 0
```

```
if msg->Data = 1, then msg->String will be RatioType.
if msg->Data = 0, then RatioType will be msg->String.
```

(FUNCTION #: 106)

1.26 DT_RATIO <Func: 107>

DT_RATIO Allows you to retrieve or change users ratio

```
msg->Command = 107
msg->Data     = 1 or 0

if msg->Data = 1, then msg->String will be ratio.
if msg->Data = 0, then ratio will be msg->String.
(FUNCTION #: 107 )
```

1.27 DT_COMPTYPE <Func: 108>

DT_COMPTYPE Allows you to retrieve or change users ComputerTypes code

```
msg->Command = 108
msg->Data     = 1 or 0

if msg->Data = 1, then msg->String will be ComputerTypes.
if msg->Data = 0, then ComputerTypes will be msg->String.
(FUNCTION #: 108 )
```

1.28 DT_MESSAGESPOSTED <Func: 109>

DT_MESSAGESPOSTED Allows you to retrieve or change users MESSAGESPOSTED

```
msg->Command = 109
msg->Data     = 1 or 0

if msg->Data = 1, then msg->String will be MESSAGESPOSTED.
if msg->Data = 0, then MESSAGESPOSTED will be msg->String.
(FUNCTION #: 109 )
```

1.29 DT_UPLOADS <Func: 110>

DT_UPLOADS Allows you to retrieve or change number of UserUploads.

```
msg->Command = 110
msg->Data     = 1 or 0

if msg->Data = 1, then msg->String will be uploads.
if msg->Data = 0, then uploads will be msg->String.
(FUNCTION #: 110 )
```

1.30 DT_DOWNLOADS <Func: 111>

DT_DOWNLOADS Allows you to retrieve or change number of UserDownloads.

```
msg->Command = 111
msg->Data     = 1 or 0
```

```
if msg->Data = 1, then msg->String will be downloads.
if msg->Data = 0, then downloads will be msg->String.
```

(FUNCTION #: 111)

1.31 DT_TIMESCALED <Func: 112>

DT_TIMESCALED Allows you to retrieve or change number of UserCalls.

```
msg->Command = 112
msg->Data     = 1 or 0
```

```
if msg->Data = 1, then msg->String will be TIMESCALED.
if msg->Data = 0, then TIMESCALED will be msg->String.
```

(FUNCTION #: 112)

1.32 DT_TIMELASTON <Func: 113>

DT_TIMELASTON Allows you to retrieve or change time user last called.

```
msg->Command = 113
msg->Data     = 1 or 0
```

```
if msg->Data = 1, then msg->String will be TIMESCALED.
if msg->Data = 0, then TIMESCALED will be msg->String.
```

NOTE: This is not a date stamp, this is the number of seconds since January 19something.

(FUNCTION #: 113)

1.33 DT_TIMEUSED <Func: 114>

DT_TIMEUSED Allows you to retrieve or change TIMEUSED today.

```
msg->Command = 114
msg->Data     = 1 or 0
```

```
if msg->Data = 1, then msg->String will be TIMEUSED.
if msg->Data = 0, then TIMEUSED will be msg->String.
```

NOTE: This is in seconds.

(FUNCTION #: 114)

1.34 DT_TIMELIMIT <Func: 115>

DT_TIMELIMIT Allows you to retrieve or change TimeAllowed for a user.

```
msg->Command = 115
msg->Data     = 1 or 0
```

```
if msg->Data = 1, then msg->String will be TIMELIMIT.
if msg->Data = 0, then TIMELIMIT will be msg->String.
```

NOTE: Time in seconds.

(FUNCTION #: 115)

1.35 DT_TIMETOTAL <Func: 116>

DT_TIMETOTAL Allows you to retrieve or change total time remaining.
for a user today.

```
msg->Command = 116
msg->Data     = 1 or 0
```

```
if msg->Data = 1, then msg->String will be time remaining.
if msg->Data = 0, then time remaining will be msg->String.
```

NOTE: Time in seconds.

(FUNCTION #: 116)

1.36 DT_BYTESUPLOAD <Func: 117>

DT_BYTESUPLOAD Allows you to retrieve or change bytes uploads per user.

```
msg->Command = 117
msg->Data     = 1 or 0
```

```
if msg->Data = 1, then msg->String will be BYTESUPLOADED.
if msg->Data = 0, then BYTESUPLOADED will be msg->String.
```

(FUNCTION #: 117)

1.37 DT_BYTEDOWNLOAD <Func: 118>

DT_BYTEDOWNLOAD Allows you to retrieve or change bytes downloaded per
user.

```
msg->Command = 118
msg->Data     = 1 or 0
```

```
if msg->Data = 1, then msg->String will be BYTESDOWNLOADED.
if msg->Data = 0, then BYTESDOWNLOADED will be msg->String.
```

(FUNCTION #: 118)

1.38 DT_DAILYBYTELIMIT <Func: 119>

DT_DAILYBYTELIMIT Allows you to retrieve or change a users daily byte download limit.

```
msg->Command = 119
msg->Data     = 1 or 0
```

```
if msg->Data = 1, then msg->String will be bytelimit.
if msg->Data = 0, then bytelimit will be msg->String.
```

(FUNCTION #: 119)

1.39 DT_DAILYBYTEDLD <Func: 120>

DT_DAILYBYTEDLD Allows you to retrieve or change daily bytes downloaded.

```
msg->Command = 120
msg->Data     = 1 or 0
```

```
if msg->Data = 1, then msg->String will be dailybytes.
if msg->Data = 0, then dailybytes will be msg->String.
```

(FUNCTION #: 120)

1.40 DT_EXPERT <Func: 121>

DT_EXPERT Allows you to retrieve or change expert mode.

```
msg->Command = 121
msg->Data     = 1 or 0
```

(FUNCTION #: 121)

1.41 DT_LINELENGTH <Func: 122>

DT_LINELENGTH Allows you to retrieve or change user LINELENGTH specs.

```
msg->Command = 122
msg->Data     = 1 or 0
```

```
if msg->Data = 1, then msg->String will be LINELENGTH.
if msg->Data = 0, then LINELENGTH will be msg->String.
```

(FUNCTION #: 122)

1.42 ACTIVE_NODES <Func: 123>

ACTIVE_NODES Allows you to retrieve a string of active&inactive nodes.

```
msg->Command = 123
msg->Data     = N/A
```

msg->String will be a string 10 bytes in length, with
'X's marking the active nodes.

NOTE: This command will surely be changing, the current
limit is 9 nodes.

(FUNCTION #: 123)

1.43 DT_DUMP <Func: 124>

DT_DUMP Allows you to dump the user's data structure to a
specified file.

```
msg->Command = 124
msg->String   = FileName
```

(FUNCTION #: 124)

1.44 DT_TIMEOUT <Func: 125>

DT_TIMEOUT Allows you to retrieve or change the door TIMEOUT limit.

```
msg->Command = 125
msg->Data     = 1 or 0
```

if msg->Data = 1, then msg->String will equal TIMEOUT.
if msg->Data = 0, then TIMEOUT will equal msg->String.

NOTE: This time is in seconds.

(FUNCTION #: 125)

1.45 BB_CONFNAME <Func: 126>

BB_CONFNAME Allows you to retrieve or change the conference name.

```
msg->Command = 126
msg->Data     = 1 or 0
```

if msg->Data = 1, then msg->String will be name.
if msg->Data = 0, then name will be msg->String.

(FUNCTION #: 126)

1.46 BB_CONFLOCAL <Func: 127>

BB_CONFLOCAL Allows you to retrieve or change the conference location.

```
msg->Command = 127
msg->Data     = 1 or 0
```

```
if msg->Data = 1, then msg->String will be location.
if msg->Data = 0, then location will be msg->String.
```

(FUNCTION #: 127)

1.47 BB_LOCAL <Func: 128>

BB_LOCAL Allows you to retrieve the current BBS location.

```
msg->Command = 128
msg->Data     = N/A
```

(FUNCTION #: 128)

1.48 BB_STATUS <Func: 129>

BB_STATUS Allows you to retrieve the current status of the node.

```
msg->Command = 129
msg->Data     = N/A
```

```
msg->String will be 'OFFLINE' or 'ONLINE' depending on
whether a user is logged onto the node.
```

(FUNCTION #: 129)

1.49 BB_MAINLINE <Func: 131>

BB_MAINLINE Allows you to retrieve the menu prompt arguments prior to the door being entered.

```
msg->Command = 131
msg->Data     = N/A
```

```
msg->String will be the menu prompt arguments.
```

(FUNCTION #: 131)

1.50 RETURNCOMMAND <Func: 136>

RETURNCOMMAND Allows you to specify an internal command to be executed when the door is finished.

```
msg->Command = 136
msg->Data     = N/A
```

command to be executed will be msg->String.
(FUNCTION #: 136)

1.51 ZMODEMSEND <Func: 137>

ZMODEMSEND Allows you to send files to the user via Zmodem protocol.

```
msg->Command = 137
msg->String   = filename (complete pathname)
msg->Data     = N/A
```

result of transfer will be in msg->Data, where

```
if msg->Data = 1 , then transfer successful.
if msg->Data = -2, then user lost carrier.
if msg->Data = 0 , then transfer unsuccessful.
```

(FUNCTION #: 137)

1.52 ZMODEMRECEIVE <Func: 138>

ZMODEMRECEIVE Allows you to receive batch uploads via Zmodem protocol.

```
msg->Command = 138
msg->String   = receive directory path
msg->Data     = N/A
```

result of transfer will be in msg->Data, where

```
if msg->Data = 1 , then transfer successful.
if msg->Data = -2, then user lost carrier.
if msg->Data = 0, then transfer unsuccessful.
```

(FUNCTION #: 138)

1.53 SCREEN_ADDRESS <Func: 139>

SCREEN_ADDRESS Allows you to retrieve the screen address.

```
msg->Command = 139
msg->Data     = N/A
```

msg->String will be a string containing the hexadecimal address of the Node screen.

(FUNCTION #: 139)

1.54 BB_TASKPRI <Func: 140>

BB_TASKPRI Allows you to retrieve the priority the node is running at.

```
msg->Command = 140
msg->Data     = N/A
```

msg->String will contain the priority of the node.

(FUNCTION #: 140)

1.55 RAWSCREEN_ADDRESS <Func: 141>

RAWSCREEN_ADDRESS Allows you to retrieve the screen address of the node.

```
msg->Command = 141
msg->Data     = N/A
```

msg->String will be a string containing the decimal address of the express node.

(FUNCTION #: 141)

1.56 BB_CHATFLAG <Func: 142>

BB_CHATFLAG Allows you to retrieve the current chat setting.

```
msg->Command = 142
msg->Data     = N/A
```

msg->String will be "ON" or "OFF".

(FUNCTION #: 142)

1.57 DT_STAMP_LASTON <Func: 143>

DT_STAMP_LASTON Allows you to retrieve a date string containing the date of when the user last logged on.

```
msg->Command = 143
msg->Data     = N/A
```

msg->String will be the date string.

(FUNCTION #: 143)

1.58 DT_STAMP_CTIME <Func: 144>

DT_STAMP_CTIME Allows you to retrieve a current time string.

```
msg->Command = 144
msg->Data     = N/A
```

msg->String will be a current time string.
(FUNCTION #: 144)

1.59 DT_CURR_TIME <Func: 145>

DT_CURR_TIME Allows you to retrieve the current time in seconds since January something.

msg->Command = 145
msg->Data = N/A

msg->String will be the current time.
(FUNCTION #: 145)

1.60 DT_CONFACCESS <Func: 146>

DT_CONFACCESS Allows you to retrieve the users conference access.

msg->Command = 146
msg->Data = 1 or 0

if msg->Data = 1, then msg->String will be AREANAME.
if msg->Data = 0, then AREANAME will be msg->String.

(FUNCTION #: 146)

1.61 BB_NODEID <Func: 149>

BB_NODEID Allows you to retrieve the Node number for the current node

msg->Command = 149
msg->Data = N/A

msg->String will be the node number.
(FUNCTION #: 149)

1.62 BB_CALLERSLOG <Func: 150>

BB_CALLERSLOG Allows you to add a line of text to the CALLERSLOG.

msg->Command = 150
msg->String = text
msg->Data = N/A

(FUNCTION #: 150)

1.63 BB_UDLOG <Func: 151>

BB_UDLOG Allows you to add a line of text to the UDLOG.

```
msg->Command = 151
msg->String   = text
msg->Data     = N/A
```

(FUNCTION #: 151)

1.64 EXPRESS_VERSION <Func: 152>

EXPRESS_VERSION Allows you to retrieve the current version string of express.

```
msg->Command = 152
msg->Data     = N/A
```

(FUNCTION #: 152)

1.65 BB_CHATSET <Func: 162>

BB_CHATSET Allows you to retrieve or change the chat status.

```
msg->Command = 162
msg->Data     = 1 or 0
```

```
if msg->Data = 1, then msg->String will be current status.
if msg->Data = 0, then status will be msg->String.
```

(FUNCTION #: 162)

1.66 ENVSTAT <Func: 163>

ENVSTAT Allows you to retrieve or change the current environment stat variable code.

```
msg->Command = 163
msg->Data     = 1 or 0
```

```
if msg->Data = 1, then msg->String will be status.
if msg->Data = 0, then status will be msg->String.
```

(FUNCTION #: 163)

1.67 NODE_DEVICE <Func: 503>

NODE_DEVICE Allows you to retrieve the node device name.

```
msg->Command = 503
msg->Data     = N/A
```

```
msg->String will be the device string.  
(FUNCTION #: 503 )
```

1.68 NODE_UNIT <Func: 504>

```
NODE_UNIT      Allows you to retrieve the node unit number.  
  
msg->Command = 504  
msg->Data     = N/A  
  
msg->String will be the current node number.  
(FUNCTION #: 504 )
```

1.69 NODE_BAUD <Func: 505>

```
NODE_BAUD      Allows you to retrieve the initialized baud rate of the node.  
  
msg->Command = 505  
msg->Data     = N/A  
  
msg->String will be the INIT baud rate.  
(FUNCTION #: 505 )
```

1.70 JH_MCI <Func: 507>

```
JH_MCI         Allows you to send MCI text to express.  
  
msg->Command = 507  
msg->String   = text  
msg->Data     = N/A  
(FUNCTION #: 507 )
```

1.71 PRV_COMMAND <Func: 508>

```
PRV_COMMAND    Allows you to immediately execute an internal express  
menu command.  
  
msg->Command = 508  
msg->String   = commandstring  
msg->Data     = N/A  
(FUNCTION #: 508 )
```

1.72 BB_CONFNUM <Func: 510>

BB_CONFNUM Allows you to retrieve the current conference number.

```
msg->Command = 510
msg->Data     = N/A
```

```
msg->String will be conference number ranging from 0 to 8.
(FUNCTION #: 510 )
```

1.73 BB_DROPDTR <Func: 511>

BB_DROPDTR Allows you to drop carrier on a user.

```
msg->Command = 511
msg->Data     = N/A
```

```
(FUNCTION #: 511 )
```

1.74 BB_GETTASK <Func: 512>

BB_GETTASK Finds the current nodes task address.

```
msg->Command = 512
msg->Data     = N/A
```

```
msg->task will be the express task address.
(FUNCTION #: 512 )
```

1.75 NODE_BAUDRATE <Func: 516>

NODE_BAUDRATE Allows you to retrieve the current users connect rate

```
msg->Command = 516
msg->Data     = N/A
```

```
msg->String will be the connect rate
(FUNCTION #: 516 )
```

1.76 BB_LOGONTYPE <Func: 517>

BB_LOGONTYPE Allows you to retrieve the LOGONTYPE.

```
msg->Command = 517
msg->Data     = N/A
```

```
msg->Data will be:
```

```
0 = AWAIT_LOGON
1 = SYSOP_LOGON
2 = LOCAL_LOGON
3 = REMOTE_LOGON
```

(FUNCTION #: 517)

1.77 BB_SCRLEFT <Func: 518>

BB_SCRLEFT Allows you to retrieve the screen coordinates.

```
msg->Command = 518
msg->Data     = N/A
```

msg->Data will be the Node's Initial LEFTEDGE coordinate.

(FUNCTION #: 518)

1.78 BB_SCRTOP <Func: 519>

BB_SCRTOP Allows you to retrieve the screen coordinates.

```
msg->Command = 519
msg->Data     = N/A
```

msg->Data will be the Node's Initial TOPEDGE coordinate.

(FUNCTION #: 519)

1.79 BB_SCRWIDTH <Func: 520>

BB_SCRWIDTH Allows you to retrieve the screen coordinates.

```
msg->Command = 520
msg->Data     = N/A
```

msg->Data will be the Node's Initial screen height.

(FUNCTION #: 520)

1.80 BB_SCRHEIGHT <Func: 521>

BB_SCRHEIGHT Allows you to retrieve the screen coordinates.

```
msg->Command = 521
msg->Data     = N/A
```

msg->Data will be the Node's Initial screen width.

(FUNCTION #: 521)

1.81 BB_PURGELIN <Func: 522>

BB_PURGELIN Allows you to abort serial input.

```
msg->Command = 522
msg->Data     = N/A
```

aborts serial input and flushes the serial buffer and sends a request for more input.

(FUNCTION #: 522)

1.82 BB_PURGELINESTART <Func: 523>

BB_PURGELINESTART Allows you to CLEAR the serial buffer and request more serial input.

```
msg->Command = 523
msg->Data     = N/A
```

(FUNCTION #: 523)

1.83 BB_PURGELINEEND <Func: 524>

BB_PURGELINEEND Allows you to CLEAR the serial buffer.

```
msg->Command = 524
msg->Data     = N/A
```

(FUNCTION #: 524)

1.84 BB_NONSTOPTEXT <Func: 525>

BB_NONSTOPTEXT Allows you to change the NONSTOP text scrolling flag.

```
msg->Command = 525
msg->Data     = 1 or 0
```

if msg->Data = 1, then display text will not pause.

if msg->Data = 0, then display text will pause.

(FUNCTION #: 525)

1.85 BB_LINECOUNT <Func: 526>

BB_LINECOUNT Allows you to retrieve or change the user's current number of lines viewed.

```
msg->Command = 526
msg->Data     = 1 or 0
```

```
        if msg->Data = 1, then msg->String will be current line.
        if msg->Data = 0, then current line will be msg->String.
(FUNCTION #: 526 )
```

1.86 DT_LANGUAGE <Func: 527>

DT_LANGUAGE Allows you to retrieve or change the current language specifications.

```
msg->Command = 527
msg->Data     = 1 or 0
```

if msg->Data = 1, then msg->String will be language.
if msg->Data = 0, then language will be msg->String.

NOTE: Languages need to be standardized, please what for guidelines.

ie: Default language .TXT
 English .ENG
 German .GER

note that this only effects the menus, bulletins &
other screen text files.

(FUNCTION #: 527)

1.87 DT_QUICKFLAG <Func: 528>

DT_QUICKFLAG Allows you to change the QUICKTEXT flag.

```
msg->Command = 528
msg->Data     = 1 or 0
```

if msg->Data = 1, then the QuickFlag will be set.
if msg->Data = 0, then the QuickFlag will not be set.

(FUNCTION #: 528)

1.88 DT_GOODFILE <Func: 529>

DT_GOODFILE Allows you to set the results of a tested file after upload.

```
msg->Command = 529
msg->Data     = 1,0 or -1
```

if msg->Data is 1, then the file was not tested.
if msg->Data is 0, then the file passed the filetest.
if msg->Data is -1, then the file failed the filetest.

NOTE: This command is only useful with the SYSCmd door

called FILECHECK.
(FUNCTION #: 529)

1.89 System-X Only Print String

SX_PS Prints a string to the console/serial port.

msg->Command = 1500
msg->Data = pointer to a string

Note: no copying required and no length limit!

(FUNCTION #: 1500)

1.90 Get UserStruct Pointer (UserData)

SX_USERPO Gives you the pointer to the User structure

msg->Command = 1501

returns the pointer in msg->Data

(FUNCTION #: 1501)

1.91 Get UserStruct Pointer (SXUser)

SX_USERPO2 Gives you the pointer to the SXUser structure

msg->Command = 1502

returns the pointer in msg->Data

(FUNCTION #: 1502)

1.92 Get UserStruct Pointer (Index)

SX_USERPO3 Gives you the pointer to the User Index structure

msg->Command = 1503

returns the pointer in msg->Data

(FUNCTION #: 1503)

1.93 Get the System-X version

SX_VER Gives you the version of SX

```
msg->Command = 1504
```

```
returns it in msg->String, eg: "1.00"
```

```
(FUNCTION #: 1504 )
```

1.94 Get a pointer to the node structures

SX_NODES Get a pointer to the node structures

```
msg->Command = 1505
```

```
returns it in msg->Data
```

For more information, see MCP Communication/Command 14.

```
(FUNCTION #: 1505 )
```

1.95 MCP

Communication with MCP is available through standard EXEC-MESSAGES.

The structure of the message is (you must allocate it)

```
struct MCPMessage
{
    struct Message Msg;
    UWORD command;
    UWORD nodenum;
    long data1;
    long data2;
    long data3;
    long data4;
    long data5;
    long data6;
};
```

UWORD's are 16-bit

long's are 32-bit

If anything but an actual node wants to use this port, you must make nodenum equal 0 (zero) (that includes doors!).

```
.------.
| MCP Commands Available |
\-----/
```

Note: Commands 1 to 5 are PRIVATE!!

6 - MENU POINTER

Ask MCP to give you a pointer to a menu structure

```
themsg.command = 6
themsg.data1    = pointer to the menu you want
                eg: "Main_Menu"
```

returns

```
themsg.data2    = the pointer you wanted
themsg.data3    = size of the struct in bytes
```

8 - MD5 ENCRYPTION

This will ask MCP to encrypt a string for you, using MD5.

```
themsg.command = 8
themsg.data1    = pointer to the string to encrypt
themsg.data2    = pointer to 16 bytes to put the result in
```

10 - QUERY SYSOP

Find out if the sysop is available

```
themsg.command = 10
```

returns

```
themsg.data1    = pointer to the reason for being away
                (if NULL, the sysop is available)
```

12 - APPEND TO A FILE

```
themsg.command = 12
themsg.data1    = pointer to the filename string
themsg.data2    = pointer to the buffer to append
themsg.data3    = size of the append buffer
```

14 - GET A POINTER TO THE NODE INFORMATION CHAIN

```
themsg.command = 14
```

returns

```
themsg.data1    = the pointer to node 1's structure
                (use the NEXT field to browse through)
```

the struct is as follows

```
struct node_struct
{
    /* offset DEC */
    APTR next;      /* 0 */
    struct UserData *User;    /* 4 */
    struct UserIndexStruct *UserIndex; /* 8 */
    struct SXUserStruct *SXUser; /* 12 */
    char *action;    /* 16 */
    char *filename;  /* 20 */
    long baud;       /* 24 */
    long loginsecs;  /* 28 */
    UWORD number;    /* 32 */
    UBYTE actionnumber; /* 34 */
    UBYTE active;    /* 35 */
    UBYTE useron;    /* 36 */
    UBYTE misc;      /* 37 */
};
```

```
-----
| Example 'C' program using MCP's port |
|-----|
```

```
struct MCPMessage
{
    struct Message Msg;
    UWORD command;
    UWORD nodenum;
    long data1;
    long data2;
    long data3;
    long data4;
    long data5;
    long data6;
};

struct MCPMessage themsg;
struct MsgPort *MyPort, *MCPPort;

MCPPort = FindPort("SX-MCP");
if(MCPPort)
{
    MyPort = CreateMsgPort();
    if(MyPort)
    {
        themsg.Msg.mn_Length = sizeof(struct MCPMessage);
        themsg.Msg.mn_ReplyPort = MyPort;
        themsg.nodenum = 0;
        themsg.command = 10; /* 10 = QUERY SYSOP */
        PutMsg(MCPPort, (struct Message *)&themsg);
        WaitPort(MyPort);
        DeleteMsgPort(MyPort);
        if(themsg.data1)
        {
            PutStr("The Sysop is away because:\n\n");
            PutStr((char *)themsg.data1);
            PutStr("\n");
        } else {
```

```
        PutStr("The Sysop is available!\n");
    }
}
}
```

1.96 Execute an internal SX Function

SX_FUNCTION Execute an internal SX Function (sxfunctions.doc)

msg->data = pointer to this structure:

```
struct SXFuncStruct
{
    UWORD id;
    char *string;
    UWORD extra;
    UWORD low;
    UWORD high;
    char *mainarg;
    char *execarg;
} *SXFunc;
```

Must must allocate this structure and fill in all the fields, then pass it through msg->data.

(FUNCTION #: 1506)