

# GBPPR 'Zine



Issue #14 / The Monthly Journal of the American Hacker / May 2005

*"... There must be some control of what's said on the airwaves, and who says it. In the absence of control, there is only static. Random broadcasts would squash each other. Radios would become useless. Besides, there are plenty of legal stations playing by the rules broadcasting enough programming to suit virtually every taste..."*

*"Outlaw broadcasters operate without concern for decency, public safety and for the thousands who play by the rules and depend on the airwaves to make a living. The pirate thing has become one of our biggest issues. It drives everybody crazy." – C. Patrick "Pat" Roberts, President of the Florida Broadcaster's Association*

## Table of Contents

- ◆ **Page 2 / Nortel DMS-100 Table AMAOPTS**
  - ◆ DMS-100 Automatic Message Accounting options. (Refer to LOD/H Technical Journal #3)
- ◆ **Page 24 / Nortel DMS-100 AMA Log Reports**
  - ◆ DMS-100 Automatic Message Accounting log reports, a phreaks worst nightmare.
- ◆ **Page 34 / Nortel DMS-100 SOS Script Overview**
  - ◆ Found this on the Interweb, seriously.
- ◆ **Page 129 / Vehicle Tracking Beacon**
  - ◆ Build a simple "bumper beeper" using a FRS radio.
- ◆ **Page 135 / Outside Plant Symbols**
  - ◆ Overview of Bell System map and location symbols.
- ◆ **Page 153 / Outside Plant Codes & Symbols**
  - ◆ Overview of Bell System map and location codes.
- ◆ **Page 174 / TV-B-Gone Jammer**
  - ◆ TV remote control jammer.
- ◆ **Page 178 / Bonus**
  - ◆ Hack the Planet!
- ◆ **Page 179 / The End**
  - ◆ Editorial and Rants.

---

# Nortel DMS-100 Table AMAOPTS

---

## Table Name

Automatic Message Accounting Options

## Functional Description of Table AMAOPTS

Table AMAOPTS is used to control the activation and scheduling of the recording options for Automatic Message Accounting (AMA). Table AMAOPTS contains one tuple for every option, and initially contains the default values for each of these options. The default values of these options are dependent on the entry in field `FORMAT` in table `CRSFMT` (Call Record Stream Format) where entry in field `KEY` is `AMA`. The entry in field `FORMAT` can be `BCFMT` (Bellcore Format) for Bellcore offices, `NTFMT` (Nortel Networks Format), or another format type for non-Bellcore offices. Some of the options replace office parameters that were used in earlier software releases.

These options cannot be deleted. Adds are performed to force a display of the contents of the table and to make the tuple known to the table editor to allow for changes to the tuple. By changing the scheduling information for the options, the operating company can activate, deactivate, and schedule the options at specified dates and times, and so control the output produced by the AMA system.

The initial contents of table AMAOPTS are defined at Initial Program Load (IPL) time, but are not displayed in the table until the tuples have been added. To add entries without affecting the default scheduling values, use the selector `DEFAULT` in field `AMASEL`, and table control replaces the selector `DEFAULT` with the default value.

For example, to add the default entry `DA411` to table AMAOPTS in field `AMAOPT`, enter the tuple `DA411_DEFAULT`. The entry `DA411 OFF` is displayed, indicating that the option is disabled as a default.

If the options that control unanswered call recording (`UNANS_TOLL`, `UNANS_LOCAL`, and `UNANS_TOPS`) are set to `Y` (yes), table `BCCODES` (Bellcore Codes, formerly `ATTCODES`) must also be datafilled to route unanswered calls that are not equal access calls to the call recording. Call records are produced for all equal access calls, both answered and unanswered, regardless of the datafill in these tables. See table `BCCODES` for further explanation.

See the first table below for a description of available options. See the second and third tables below for the default schedule values for each option, and see the fifth table for the allowable values of field `AMASEL` for each option.

---

### *Description of Available AMA Options*

<b>Option</b>	<b>Explanation</b>
<code>ACBAR_MOD_CO</code>	This option provides a method of identifying any billable calls that are set up with Automatic Call Back (ACB) or Automatic Recall (AR) features, and includes a way of distinguishing calls to private numbers using ACB or AR. The setting of option <code>ACBAR_MOD_CO</code> determines whether features ACB and AR and the privacy status of a call are provided. The option <code>ACBAR_MOD_CO</code> is an ON or OFF parameter. If <code>ACBAR_MOD_CO</code> is set ON, features ACB and AR and the privacy status of a call are found in the

appended module code 068. The information added to the billing record is interpreted by the downstream billing center. If it is a private number, the Directory Number (DN) is not printed on the subscriber's invoice. If option ACBAR\_MOD\_CO is set OFF, module code 068 is not appended; therefore, features ACB and AR and the privacy status information of a call are not provided. If option ACBAR\_MOD\_CO is set OFF, the DN of any private number is printed on the subscriber's invoice. The default value is OFF. Values other than ON, OFF, or DEFAULT are not valid.

---

ACBAR_STY_IN	This option provides a method of identifying any billable calls that are set up with ACB or AR features, and includes a way of distinguishing calls to private numbers using ACB or AR. The setting of option ACBAR_STY_IN determines whether features ACB and AR and the privacy status of a call are provided. The option ACBAR_STY_IN is an ON or OFF parameter. If option ACBAR_STY_IN is set ON, features ACB and AR and the privacy status of a call are found in character 5 of the study indicator. The information added to the billing record is interpreted by the downstream billing center. If it is a private number, the DN is not printed on the subscriber's invoice. If option ACBAR_STY_IN is set OFF, features ACB and AR and the privacy status of a call are not found in character 5 of the study indicator. If option ACBAR_STY_IN is set OFF, the DN of any private number is printed on the subscriber's invoice. The default value is OFF. Values other than ON, OFF, or DEFAULT are not valid.
--------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

---

AMATRKTG_ANS	This option controls the generation of the AMA record triggered by the option AMATRKTG in table AMATKOPT for all calls or for answered calls only. If this option is ON, the AMA records are generated only for answered calls.
--------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

---

APPEND_ISDN_CKT_ID	This option controls the production of the ISDN channel identifier (module 180) and trunk identification (module 181) module codes. This option gives office-wide control of these module codes. The module codes are produced for originating and terminating BRI and PRI ISDN calls. To turn on the recording of modules 180 and 181, set APPEND_ISDN_CKT_ID to ON. To stop the recording of modules 180 and 181, set APPEND_ISDN_CKT_ID to OFF. The default value for this option is OFF.
--------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

---

APPEND_PRI_MODULE	This option controls the addition of AMA module 070/071 to billing records for PRI originating calls. This option can be set to ON or OFF. The default value is OFF. To turn on the recording of module 070 or 071, set APPEND_PRI_MODULE to ON. To turn off the recording of module 070 or 071, set APPEND_PRI_MODULE to OFF.
-------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

---

AR_BILLING	This option provides the possibility to append a module code 611 to the AMA record, to indicate that a call has been made using the Automatic Recall/Automatic Recall with Dialable Directory Number (AR/ARDDN) feature. If the option is set to ON, module code 611 with a generic context ID of SUSP (80024) is appended to the AMA record when such a call is made. If the option is set to OFF, a normal AMA record is generated.
------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

---

AUDIT	This option controls the resetting of the internal AMA counts for the Operational Measurements (OM) tracer record and sets up the accumulated time change for the new day. This option cannot be changed, and appears for information only.
-------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

---

BACK_CHARGE	This option controls the addition of module 611 for backwards charging information on French Telephony User Part (FTUP). This activity sends charging information during the call to the (calling subscriber's) switch that performs the billing. This option allows the service provider to control the billing of the call.
-------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

---

BCLID_USPAUD	This option controls the generation of AMA records by the Bulk Calling Line Identification Usage-Sensitive Pricing Audit (BCLID_USPAUD) that is scheduled in table AMAOPTS. Option SUSP (Subscriber Usage-Sensitive
--------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Pricing) must be turned on to obtain BCLID\_USP records. If a DN used as a BCLID group billing DN is also used as a Custom Local Area Signaling Services (CLASS) line with at least one CLASS display feature, then two AMA records are generated for the DN.

Field USP in table BCLIDGRP controls the collection of AMA information for each group of BCLID subscribers. If field USP is datafilled Y (yes), AMA counts are collected for the group. The peg counts store the number of full calling DNs delivered and the combined number of PRIVATE and OUT-OF-AREA indications delivered. An AMA record is generated for each BCLID group that is datafilled for Usage-Sensitive Pricing (USP).

BCLID groups that are datafilled with N (no) in field USP in table BCLIDGRP are not billed on a usage-sensitive basis. No AMA records are generated for these BCLID groups.

---

BCLONGCALL This option allows setting of the record generation time for long-duration Bellcore format records.

---

CALL\_FWD This option controls the usage recording of Call Forwarding (CFW/CFX).

---

CALL\_TIMECHG This option governs the triggering of the new time-change module on the AMA record. There are three possible values:

- \* ON: Enables this functionality if a time change (CI commands SETTIME or SETDATE) occurs during a call, the time-change module is appended to the resultant AMA record.

- \* OFF: Disables this functionality. This is the initial value.

- \* DEFAULT: Equates to OFF in all cases. When datafilling this tuple, option UNIVERSAL\_AMA\_BILLING in table OFCENG must be set to Y and option TIMECHANGE in table AMAOPTS must be set to OFF.

---

CAPTURE\_ This option controls the generation of MCI 098 for unanswered calls.  
CKTSZ\_UNANS When set to ON, the terminating circuit seizure date and time will be captured in MCI 098.

---

CAPTURE\_ This option controls the capture of the Class of Service (COS) Index.  
CLASS\_SERV When set to ON, the most recent Class of Service Index assigned to the originating trunk group will be captured in BCD char 6-9 of MCI 611. The MCI 611 together with the new context ID (80058) will be attached to the AMA record. If table control is set to OFF a dummy value of hex F will be captured in BCD char 6-9.

By default the COS captured in MCI 611 will be the one entered in the table TRKOPTS. If the COS is overridden during the translations then the COS captured in the MCI 611 will be the COS entered in the table CLISRVPF. The COS value assigned to the call can range from 0 - 1023.

If all the three table controls (CAPTURE\_CLASS\_SERV, CAPTURE\_COMPL\_CODE and CAPTURE\_SAT\_IND) are set to OFF then the MCI 611 with the new context ID (80058) will not be attached to the AMA record.

---

CAPTURE\_ This option controls the capture of the completion code, representing the  
COMPL\_CODE reason for the call termination. The completion code will be captured for either the: release cause, treatment or called/calling party disconnect.

If the release cause is available, then the completion code corresponding to the release cause will be captured. If the release cause is not available, then the treatment code, if available, will be captured as the completion code. If neither are available, the calling party/called party disconnect information will be captured as the completion code.

Completion code information is captured for the following protocols:

- \* ETSI ISUP V1
- \* ETSI ISUP V2
- \* IBN7
- \* FST R1
- \* FST R2
- \* RBTUP
- \* BTUP
- \* DPNSS

When set to ON, the reason for the call termination will be added to MCI 611 (BCD char 2-4) with the new context ID (80058), and will be attached to the AMA record. The BCD char 1 will indicate the event in the call for which the completion code is captured. If table control is set to OFF a dummy value of hex F will be captured in BCD char 1-4.

If all the three table controls (CAPTURE\_CLASS\_SERV, CAPTURE\_COMPL\_CODE and CAPTURE\_SAT\_IND) are set to OFF then the MCI 611 with the new context ID (80058) will not be attached to the AMA record.

---

CAPTURE\_  
INAP\_CPC            This option controls the capture of INAP CPC. If the option is turned ON and if the call involves INAP, the INAP CPC, if it exists, is captured in the AMA module code 611 (context ID: 80027).

---

CAPTURE\_  
SAT\_IND            This option controls the capture of satellite indicator information. If the call involved a satellite circuit then this one bit flag is set to 1 (ON). Information in this field is based on the SAT field value that is datafilled against the originating trunk in table TRKSGRP. For ISUP (ETSI ISUP V1, ETSI ISUP V2, IBN7) and RBTUP protocols this information is based on the satellite information carried by the incoming IAM and the SAT field value that is datafilled against the originating trunk in the table TRKSGRP. The satellite circuit information is captured in the BCD char 5 of the MCI 611. MCI 611 with the new context ID (80058) will be attached to the AMA record. If table control is set to 0 (OFF) a dummy value of hex F will be captured.

If all the three table controls (CAPTURE\_CLASS\_SERV, CAPTURE\_COMPL\_CODE and CAPTURE\_SAT\_IND) are set to OFF then the MCI 611 with the new context ID (80058) will not be attached to the AMA record.

---

CCBS\_BILLING        This option controls Call Completion to Busy Subscriber (CCBS) billing, adding usage billing for CCBS calls. It is possible to mark the billing record to indicate CCBS usage thereby providing the ability to charge for the successful usage of CCBS. The option can be set to either ON or OFF.

If the option is set to ON, it indicates that CCBS usage billing indication will be provided in AMA records for CCBS originated calls. If the option is set to OFF, no CCBS usage billing will be provided.

Indication of CCBS usage is provided for the following scenarios where CCBS has been initiated and the called party has become free:

- \* Originator ignores the ring back.
- \* Originator answers the ring back call and disconnects immediately (before or after the called party has rung).
- \* Ringing is applied to the called party and he doesn't answer.
- \* A complete call setup occurs between the originator and the called party. In this scenario if an answer message is received with no charge indication then the record will be marked as unanswered with CCBS usage marked.

**Note:** This option applies to both IBN and EBS lines.

The service feature field (field 012) is used to indicate that CCBS usage has been successful. CCBS usage is marked with a service feature code of '029' in the billing record.

**Note:** The service feature code value of '029' is also used to indicate BTUP CBWF usage.

---

CCSADATA	<p>This option controls the precedence of call code 021 over call code 072 for Common Control Switching Arrangement (CCSA) data calls. It has no effect on non-data calls.</p> <p>A call code 072 record is produced for a CCSA data call if the CCSADATA option is set to the default value OFF.</p> <p>A call code 072 record is replaced by call code 021 if the call is a CCSA data call and the CCSADATA option is set to ON.</p>
CDAR	<p>This option controls the recording of customer-dialed account codes.</p>
CDAR_EXTENDED	<p>This option controls the generation of the AMA module code 850 for the capture of account codes.</p>
CDRDUMP	<p>This option controls the generation of an audit that provides a billing record for all calls in progress on the DMS-100 at a specified time. The default value for this option is OFF, which requires that the audit be explicitly set. Once the specified time for the audit run is reached, a copy of the Call Detail Recording (CDR) for all calls in progress at that time is generated and sent to the respective AMA process.</p>
CDRLONGCALL	<p>This option controls the scheduling of a new audit to produce long-duration call CDRs (46-byte CDR format only). The first check occurs at the time specified by field START, followed by periodic checks with a period specified by field PERIOD. The option cannot be set with a period less than eight hours. ON enables the functionality. OFF disables the functionality. DEFAULT is PERIODIC with a period of eight hours, starting at the next occurrence of either 05:15, 13:15, or 21:15.</p>
CDRSYNC	<p>This option allows synchronization records (code 1B) to be added to the CDR data stream at intervals specified by the user. The option can be datafilled as OFF or PERIODIC. The CDR stream must be datafilled in field KEY of table CRSFMT before setting option CDRSYNC to PERIODIC. The default for this option is OFF and the format is CDR300FMT.</p>
CHG411	<p>This option controls charging for 411 Directory Assistance (DA) calls. Option DA411 must be set ON along with option CHG411 before local DA calls are written as charged. Local DA records are written as study records if option DA411 is set ON and option CHG411 is OFF.</p>
CHG555	<p>This option controls charging for seven-digit (555-1212) DA calls. Option DA555 must be set ON along with option CHG555 before 555 DA calls are written as charged. DA calls are written as study records if option DA555 is set ON and option CHG555 is OFF.</p>
CIDSUSPAUD	<p>This option controls the Calling Information Delivery (CID) Subscription Usage-Sensitive Pricing (SUSP) audit that generates a billing record (for each Calling Number Delivery (CND), Dialable Number Delivery (DDN) and/or Calling Name Delivery (CNAMD) feature subscriber) that contains peg counts for each of the CLASS SUSP display options.</p> <p>If option SUSP in table AMAOPTS is ON or has been ON at any point during the audit period, then at the datafilled start time and interval, the audit</p>

---

produces a record for each subscription CNAMD or CND/DDN subscriber.  
A subscription option is one that has its billing option field set to AMA.

If option SUSP in table AMAOPTS is OFF and has not been ON during the audit period, then the audit is not run and no records are produced.

If a subscriber has not received any calls that increment the delivery counts for a CLASS SUSP display, then a record that contains 0 (zero) in the available and unavailable count fields for that option is produced if:

- \* The CLASS SUSP display option is currently active.
- \* The CID options are enabled in table RESOFC.
- \* Option SUSP was ON in table AMAOPTS during the last audit period.

After the audit is complete, the AMA registers that are associated with each subscriber are set to 0 (zero) in preparation for the next audit period.

Since this audit can produce many AMA records, we recommend that the operating company schedule the audit during low traffic hours and on a 24 hour basis.

The audit is not the only source of billing records. If a CLASS SUSP display option is changed to flat-rate or removed from the line, the delivery counts stored for the CLASS SUSP display option are reported in an AMA record appropriately.

Value DEFAULT in NTFMT (Northern Telecom Format) AMA switching units is OFF and cannot be changed. Value DEFAULT in BCFMT (Bellcore Format) AMA switching units is PERIODIC and set to run once a day at midnight. Although option CIDSUSPAUD in table AMAOPTS cannot be turned off in a BCFMT switching unit, the audit does not run if option SUSP has been turned off since the last time the audit ran.

---

CITYWIDE	This option allows generation of billing records for intra-citywide Virtual Private Network (VPN) calls that are normally non-billable. Value ON turns billing on. The default value is OFF. Values other than ON, OFF, or DEFAULT are not valid.
CLI_DELV	This option controls the capture of the CLI delivery indicator. If the option is turned ON, the CLI delivery indicator is captured in the AMA module code 611 (context ID: 80058).
CMCICWK	For a terminating Central Message Control (CMC) call, the carrier connect time is the time the incoming CMC trunk is seized. If option CMCICWK is set to ON, the carrier connect time on a CMC to Feature Group D (FGD) carrier call is the time of billing wink from the FGD carrier. A CMC access record is not created unless a billing wink is received from the FGD carrier.
CMCORIG	This option controls the generation of originating CMC billing records (call codes 63 and 64). If this option is set OFF, no call code 63 or 64 records are created.
CMCTERM	This option controls the generation of terminating CMC billing records (call codes 65 and 66). If this option is set to OFF, no call code 65 or 66 records are made.
COIN	This option controls the recording of all local coin calls.
COLL_SVC_BILL_INFO	This option controls the use of Japan billing enhancements. These enhancements have an effect on module codes 611 and 612. To enable Japan billing enhancements, set option COLL_SVC_BILL_INFO to ON. If this option is OFF, the switch ignores AMAOPTS options RECORD DIALED_DIGS and MC611_80005.

CRSEQNUM	This option controls the generation of a call record sequence number for each call record.
CRT_BILLING	This parameter controls billing records for the call redirect feature on an office-wide basis. CRT_BILLING set to ON creates a billing record for the redirected call. With CRT_BILLING set to OFF, a billing record does not generate for the redirected call.
CSMI	This option can be set to ON or OFF in order to enable or disable billing for CSMI.
DA411	This option controls the recording of calls to a local DA operator. Local DA calls are recorded as study records if option DA411 is set ON. DA calls are written as charged if options DA411 and the CHG411 are both set to ON.
DA555	This option controls the recording of seven-digit (555-1212) directory assistance calls. Ten-digit (NPA-555-1212) DA calls are always recorded as station-paid calls. DA 555 calls are recorded as study records if option DA555 is set ON. DA 555 calls are written as charged if options DA555 and CHG555 are both set ON.
DISABLE_MCD_ INFO	This option activates the Minimum Call Duration parameter used on DMS-100 switches in Israel. DISABLE_MCD_AMA supports special functionality for short calls that can be used by other customers. If the option is set ON, the minimum call duration is activated. The default value is OFF.
ENABLE_SCI500	This option sets the AMA environment for DMS-100 switches in Israel. ENABLE_SCI500 provides structure code 500 in Bellcore AMA records. If the option is set ON, the AMA environment is set up for the Israeli market. Option ENABLE_TLR_BILLING must be set ON before option ENABLE_SCI500 is activated. The default value is OFF. Any change in this option requires a reload restart.
ENABLE_TLR_ BILLING	This option sets the AMA environment for DMS-100 switches in Israel. If the option is set ON, the AMA environment is set up for the Israeli market. The default value is OFF. This option controls TIV logs, generation of AMA record as a result of successful wake up request activation, generation of RBP log, implementation of OGINFNO option in AMATKOPT, special Telrad implementation of CAMA, special service features in records of BOOMERANG and VML calls, sending of AMA records via MPC card, supporting of a special structure for BEARER_CAPABILITIES and SERVICE_FEATURE fields in the AMA record, implementation of FLEXCLI option in TRKOPTS table. If there is FLEXCLI option for the trunk group in TRKOPTS table, the modification of the CLI is done as written in FLEXDIGS table and the new DN is written in the record unit for AMA. Any change in this option requires a reload restart.
ENFIA_B_C	This option controls the AMA recording of ENFIA B and ENFIA C calls (for example, 950-10xx).
FREECALL	This option controls the recording of all local calls that terminate on a free number.
FTRCODE	This option indicates whether the originating and terminating feature codes in Bellcore AMA are required for international billing. Setting the tuple FTRCODE to ON results in module code 509 being appended to the 510 structure code if one of the recorded features is activated by either the calling or called party. TIMED and PERIODIC are not valid values, and are blocked when entered. The value DEFAULT sets the tuple to OFF.
GFTBILL	This option counts transport activities that have a generic function. This option appends the activities to the AMA billing record. Value ON

turns billing on. The default value is OFF.

---

HIGHREV	This option suppresses the generation of all AMA data except for call types that are listed as high revenue in table BCCODES.
---------	-------------------------------------------------------------------------------------------------------------------------------

---

IC_CDPN_INFO_REQD	This option controls the capture of Incoming Called Party Number NPI and NOA or TON information. The FLEXCPNI option (subfield IC_CDPN_INFO) in tables AMAXLAID and FLEXAMA activates this capture.
-------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

---

IC_CGPN_INFO_REQD	This option controls the capture of Incoming Calling Party Number (CGPN) Numbering Plan Indicator (NPI) and Nature Of Address (NOA) or Type Of Number (TON) information. The FLEXCPNI option (subfield IC_CGPN_INFO) in tables AMAXLAID and FLEXAMA activates this capture.
-------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

---

IC_CGPN_PI_REQD	This option provides a means to capture the incoming calling party's Presentation Indicator (PI) in AMA records and conveys the information about the Calling Line Identity (CLI) presentation or restriction.
-----------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

---

INFO_DIGIT	This option controls whether an extra digit in the calling_dr field of the translation block in the CCB is copied into the calling_dr of the AMA PRU. When the option INFO_DIGIT is ON, the extra digit is not copied. When the option INFO_DIGIT is OFF, the extra digit is copied.
------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

---

INTL_ICR_REQD	This option provides office-wide ICR activation for all IBN and BRI lines.
---------------	----------------------------------------------------------------------------

---

INTRASITE	This option allows generation of billing records for normally non-billable intra-site VPN calls. Value ON turns billing on. The default value is OFF. Values other than ON, OFF, or DEFAULT are not valid.
-----------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

---

INWATS	This option controls the recording of all Inward Wide-Area Telephone Service (INWATS) calls.
--------	----------------------------------------------------------------------------------------------

---

ISDN_ACCIND	When set to ON, this option provides the correct ISDN access indicator value in the module code 611 AMA billing record for the supported interworkings, providing the options COLL_SVC_BILL_INFO and MC611_80005 are also ON. When the ISDN_ACCIND option is set to OFF, the terminating ISDN indicator is recorded as 'unknown' for some of the supported interworkings in the Japan market.
-------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

---

ISDN_ETSI_BS	When set to ON, this option specifies that all ISDN PRI and BRI calls have a module code 030 appended to their AMA billing record.
--------------	------------------------------------------------------------------------------------------------------------------------------------

---

ISDNBBGBILL	<p>This option allows the production of billing records for Basic Business Group (BBG) facilities and services on ISDN BRI lines. If ISDNBBGBILL is ON, the switch appends module code 074 to all ISDN BRI AMA records.</p> <p>Module code 074 identifies:</p> <ul style="list-style-type: none"><li>* The BBG call type.</li><li>* The billing number of the BBG customer.</li><li>* The associated Virtual Facility Group (VFG) or Trunk Group Number (TGN) used in the call.</li></ul> <p>This option applies only to ISDN BRI lines, and it controls the production of BBG billing records office-wide.</p>
-------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

---

ISDNCIRCUIT	This option controls the production of the ISDN circuit structures and philosophies office-wide. The structures consist of ISDN core module 070/071 and ISDN terminating user service module 073. The philosophies consist of call types and allow the operating company to specify signaling capabilities that are considered a basis for originating or terminating detailed billing.
-------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

---

LNID	This option can only be OFF. This tuple is forced to OFF during a dump
------	------------------------------------------------------------------------

and restore or when operating company personnel attempt to change its value. This option controls the inclusion of a coded representation of the line ID in the AMA record. This option has no effect if entry in field FORMAT, in table CRSFMT, where field KEY = AMA, is NTFMT.

---

LNP_721	This option creates an LNP record for Bellcore call code 721. The option has values of ON, OFF, and DEFAULT. The default is OFF.
LNP_721_Use_ SC0500	This option creates an LNP record for Bellcore call type code 721 with structure code 500. The option has values of ON, OFF, and DEFAULT. The default is OFF. With this option OFF, LNP721 records generate with structure code 0001.
LNP_721_ PortedDNOnly	This option creates an LNP record for Bellcore call code 721 only when the dialed DN is ported. The option has values of ON, OFF, and DEFAULT. The default is OFF.
LNP_722	This option creates a last resort LNP record, Bellcore call code 722. Last resort AMA is generated when last resort routing to a recipient switch occurs at a donor switch and an AMA record is not already being generated at the donor switch. The option has values of ON, OFF, and DEFAULT. The default is OFF.
LNP_BILL_ DONOR	This option is used to trigger AMA billing records at the donor exchange in LNP QoR calls.
LNP_MODULE_ 719	This option controls the kind of module that LNP appends to an AMA record. The default is OFF, which means that LNP stores portability information in module 720.
LOG117_ CALLING_DN	This option controls whether the calling DN is copied into the AMAB117 log. ON copies the DN to AMAB117. OFF does not record the DN.
LOGAMA	This option controls the generation of AMAB117 log reports. If this option is set ON, AMAB117 log reports are generated for each record that is put on the AMA tape. (This is used instead of office parameter SPECIAL_AMA_REPORT in table OFCVAR)
LOGOPT	This option controls the generation of a log outlining the status of the AMA recording options such as active or inactive.
LOGTEST	This option controls the generation of AMAB200 log reports. If it is set ON, AMAB200 log reports are generated for AMA billable calls to or from a line with the line option AMATEST enabled in table LENLINES.
LONGCALL	This option controls the production of AMA records periodically during the course of long-duration calls.
LUSORIG	This option controls the recording of all calls that originate on a line with Line Usage Study (LUS) as defined in table LENFEAT.
LUSTERM	This option controls the recording of all calls that terminate on a line with LUS as defined in table LENFEAT.
MC611_80005	This option controls the addition of module code 611 (80005) to AMA billing records for the office. To enable the recording of module code 611 (80005), set option MC611_80005 to ON. To disable the recording of module code 611 (80005), set option MC611_80005 to OFF. The default value for this option is OFF.
MWIC_AUDIT	This option provides the DMS-100 switch the ability to generate daily AMA records of aggregate counts of successful MWI control activations and deactivations on an MSRID basis. The MWIC_AUDIT option works in conjunction with the BILLNUM option of table MSRTAB. MWIC_AUDIT schedules

the audit to capture the metrics of the new BILLNUM option.

---

NTAI	This option controls the generation of AMA module 611 NTAI information switch-wide depending on the subfield ON/OFF setting.
OBSERVED	This option controls the recording of all calls that originate on a line with complaint Observed Studies (OBS) as defined in table LENFEAT.
OCCOVFL	This option controls the recording of equal access overflow calls (call code 120). Option OCCOVFL is dependent on the use of fixed pseudo-code EAPEG, which must be added to table CLLI and must be datafilled in table OFRT. When EAPEG is encountered in the route list of table OFRT, an overflow count is pegged against the destination carrier in table OCCINFO.
OCCTERM	This option controls the recording of terminating equal access calls (call code 119). If option OCCTERM is ON, these records are produced. If OCCTERM is OFF, these records are not produced.  <b>Note:</b> The only option supported by GSF031 release for equal access is OCCTERM.
OUTWATS	This option controls the recording of all OUTWATS calls.
OVERFLOW	This option controls the recording of all INWATS or LUS calls that failed to terminate for any reason.
RECORD_ DIALED_DIGS	This option controls the addition of module code 040 to AMA billing records for the office. Module code 040 records the dialed digits received at call setup. To enable the recording of module code 040, set option RECORD_DIALED_DIGS to ON. To disable the recording of module code 040, set option RECORD_DIALED_DIGS to OFF. The default value for this option is OFF.
RECORD_LSPI	This option enables the generation of LSPI recording on a switch-wide basis.
RECORD_ NATIVE_LSPI	This option enables the generation of module codes 338 and 125, and LSPI call type codes 126, 127, and 128 on a switch-wide basis for native agents. Turn on the RECORD_LSPI option before using this option.
RECORD_ RESOLD_LSPI	This option enables the generation of module codes 338 and 125, and LSPI call type codes 126, 127, and 128 on a switch-wide basis for resold agents. Turn on the RECORD_LSPI option before using this option.
RECORD_ SIGNAL_LSPI	This option lets users add the MC338 (LSPI) module to the generated billing record on a switch-wide basis. The default value for this option is OFF, which deactivates the option. Note that users must also activate the RECORD_LSPI option in table AMAOPTS to activate the RECORD_SIGNAL_LSPI option. When both options are ON, the switch records the signaled LSPA0 and LSPSO information in module 338 and adds it to the billing record.
RECORD_ TERMINATION	This option controls the recording of terminating information on trunk calls routed from the VPN trunk calls encountering busy or no-circuit problems.
RECORD_ _TRUNK _LSPI	This option controls the generation of trunk-related LSPI recording for selected inter-switch public trunk types. The default value is OFF, indicating that recording of trunk-related LSPI information is inactive on the switch. Turn on the RECORD_LSPI option before using this option.
RECORD_ UNBUND_LSPI	This option enables the generation of module codes 338 and 125, and LSPI call type codes 126, 127, and 128 on a switch-wide basis for unbundled agents. Turn on the RECORD_LSPI option before using this option.
SAID_MOD_	This option controls the production of module 047 on an office-wide

---

SUPPR basis. Module 047 is appended for Speech Activated Intelligent Dialing (SAID) calls. To stop the generation of module 047, set SAID\_MOD\_SUPPR to ON. To enable the generation of module 047, set SAID\_MOD\_SUPPR to OFF.

---

STORE\_ CALLREF This option enables the capture of call identity and point code in the AMA billing record. These values are obtained from the Call Reference Parameter in the IAM message. The default value is OFF, which deactivates this option. The STORE\_CALLREF option works in conjunction with the TRKOPTS option CALLREF for ETSI ISUP V2 trunks.

---

STORE\_ LRN This option is used to trigger additional AMA billing module 612 for ported-in number originated calls.

---

SUSP This option controls SUSP.

If option SUSP is OFF (the default):

- \* No SUSP billing occurs.
- \* Service order prompting for SUSP is suppressed.

If option SUSP is ON:

- \* Service order prompting for BILLING\_OPTION is enabled.
- \* SUSP billing is enabled for the office (and SUSP billing takes place on lines that have the AMA BILLING\_OPTION).

SUSP records are not recorded under any conditions if parameter AMA\_FORMAT is NT.

SUSP is also provided for the screening list editing (SLE) features: Selective Call Forwarding (SCF), Selective Call Rejection (SCRJ), Selective Call Acceptance (SCA), and Distinctive Ringing/Call Waiting (DRCW). The usage-sensitive context here means generating billing records each time the subscriber accesses an SLE USP feature screening list or activates or deactivates a SLE USP feature. Billing records are not generated each time a terminating call is screened by a subscriber's SLE USP feature.

To enable usage-sensitive pricing (USP) for SLE features:

- \* The feature or features must be enabled in table RESOFC.
- \* The feature or features must be assigned to a line.
- \* The SUSP entry in table AMAOPTS must be set ON.
- \* The BILLING\_OPTION prompt, which is displayed when adding or changing an SLE feature, must be set to AMA.

---

TIMECHANGE This option controls the generation of time-change records if the time or date on the switch is set by the Command Interpreter (CI) commands SETTIME or SETDATE at a Maintenance and Administration Position (MAP) terminal.

---

TRACER This option controls the production of an AMA tracer record containing peg counts of several AMA events such as originations or records output.

---

TRKID This option can only be OFF. This tuple is forced to OFF during a dump and restore or when operating company personnel attempt to change its value. This option controls the inclusion of a coded representation of the terminating trunk ID in the AMA record. This option has no effect when parameter AMA\_FORMAT is NT.

---

TRMTID\_CAPTURE This option controls the capture of the treatment ID applied to calls that have failed to be routed out of the DMS-100 switch. With this option ON, any billable calls that fail on 2-way or incoming ISUP, BTUP, FST R1, FST R2, RBTUP, ETSI PRI and DPNSS trunks result in the generation of an AMA module code 130 with call characteristic value 10.

TWC	This option controls the usage recording of Three-Way Calling (3WC).
U3WC	This option controls the usage recording of Three-Way Calling - Usage Sensitive (U3WC).
UNANS_AIN	This option controls Advanced Intelligent Network (AIN) specific unanswered call recording. If option UNANS_AIN is turned on, an AMA record is produced for every unanswered billable AIN call whose call type is datafilled in tuple AIN in table BCCODES. If switch-based unanswered call recording is in effect for a certain call type, then unanswered AIN calls of that call type are recorded, regardless of whether or not UNANS_AIN is turned ON. The default value is OFF.
UNANS_LOCAL	This option controls the recording of unanswered local calls. Only those local calls that generate AMA records are recorded. The call is defined as toll, local, high revenue, or Traffic Operator Position System (TOPS) in table BCCODES. The unanswered calls must have the associated call code datafilled in table BCCODES. If option UNANS_LOCAL is set ON, answered and unanswered local calls are recorded. (Equal access calls are unaffected; see note) If the unanswered call is a billable Capability Set 1 Revised (CS-1R) call, an AMA record is generated regardless of what UNANS_LOCAL is set to. A CS-1R call is billable if a Furnish Charging Information (FCI) operation is received from the SCP for that call.
UNANS_TOLL	This option controls the recording of unanswered toll calls. Only those toll calls that generate AMA records are recorded. The call is defined as toll, local, high revenue, or TOPS in table BCCODES. The unanswered calls must have the associated call code datafilled in table BCCODES. If option UNANS_TOLL is set to ON, answered and unanswered toll calls are recorded. Option UNANS_TOLL is used instead of office parameter, NO_ANS_CALLS_ONTAPE in table OFCENG. (Equal access calls are unaffected; see note) If the unanswered call is a billable CS-1R call, an AMA record is generated regardless of what UNANS_TOLL is set to. A CS-1R call is billable if a Furnish Charging Information (FCI) operation is received from the SCP for that call.
UNANS_TOPS	This option controls the recording of unanswered TOPS calls. Only those TOPS calls that generate AMA records are recorded. The call is defined as toll, local, high revenue, or TOPS in table BCCODES. The unanswered calls must have the associated call code datafilled in table BCCODES. If the option UNANS_TOPS is set ON, answered and unanswered TOPS calls are recorded. (Equal access calls are unaffected; see note)

**Note:** Originating equal access calls, both answered and unanswered, generate AMA records regardless of the datafill in tables AMAOPTS and BCCODES. For offices that require billing for all E800 calls, ensure that call codes 141 and 142 are datafilled against UNANS\_TOLL.

-End-

In table CRSFMT, if the entry in field KEY is AMA, and the entry in field FORMAT is BCFMT (Bellcore Format), the default values for the options in table AMAOPTS are as listed in the following table:

*Default Schedule Values for Bellcore Format Options*

Option	Default Schedule
ACBAR_MOD_CO	OFF
ACBAR_STY_IN	OFF
AMATKRTG_ANS	OFF
APPEND_ISDN_CKT_ID	OFF

APPEND_PRI_MODULE	OFF
AR_BILLING	OFF
AUDIT	PERIODIC yymmdd 0000 24 HRS
BCLID_USPAUD	PERIODIC yymmdd 0000 24 HRS
BCLONGCALL	PERIODIC yymmdd 0000 24 HRS
CALL_FWD	ON
CALL_TIMECHG	OFF
CAPTURE_CKTSZ_UNANS	OFF
CAPTURE_CLASS_SERV	OFF
CAPTURE_COMPL_CODE	OFF
CAPTURE_INAP_CPC	OFF
CAPTURE_SAT_IND	OFF
CCBS_BILLING	OFF
CCSADATA	OFF
CDAR	OFF
CDAR_EXTENDED	OFF
CDRDUMP	OFF
CDRLONGCALL	PERIODIC
CHG411	OFF
CHG555	OFF
CIDSUSPAUD	PERIODIC yymmdd 0000 24 HRS
CITYWIDE	OFF
CLI_DELV	OFF
CMCICWK	OFF
CMCORIG	OFF
CMCTERM	OFF
COIN	OFF
COLL_SVC_BILL_INFO	OFF
CRSEQNUM	OFF
CRT_BILLING	OFF
CSMI	ON
DA411	OFF
DA555	OFF
DSCWID_CONF_AUDIT	OFF
ENFIA_B_C	ON
FRECALL	OFF
FTRCODE	OFF
GFTBILL	OFF
HIGHREV	OFF
INTL_ICR_REQD	OFF
INTRASITE	OFF
INWATS	OFF
ISDN_ACCIND	OFF
ISDNBBGBILL	OFF
ISDNCIRCUIT	OFF
LNID	OFF
LNP_BILL_DONOR	OFF
LOGAMA	OFF
LOGOPT	PERIODIC yymmdd 1200 24 HRS
LOGTEST	OFF
LONGCALL	PERIODIC yymmdd 0000 24 HRS
LUSORIG	OFF
LUSTERM	OFF
MC611_80005	OFF
MWIC_AUDIT	PERIODIC yymmdd 0000 24 HRS
NTAI	OFF
OBSERVED	OFF
OCCOVFL	PERIODIC yymmdd 2300 1 HRS
OCCTERM	ON
OUTWATS	OFF
OVERFLOW	PERIODIC yymmdd 2300 24 HRS
RECORD_DIALED_DIGS	OFF
RECORD_LPSI	OFF
RECORD_NATIVE_LSPI	OFF

RECORD_RESOLD_LSPI	OFF
RECORD_SIGNAL_LSPI	OFF
RECORD_TERMINATION	OFF
RECORD_TRUNK_LSPI	OFF
RECORD_UNBUND_LSPI	OFF
SAID_MOD_SUPPR	OFF
STORE_CALLREF	OFF
STORE_LRN	OFF
SUSP	OFF
TIMECHANGE	OFF
TRACER	PERIODIC yymmdd hh00 1 HRS
TRKID	OFF
TWC	ON
U3WC	ON
UNANS_AIN	OFF
UNANS_LOCAL	OFF
UNANS_TOLL	OFF
UNANS_TOPS	OFF

-----

**Note:** 'yymmdd' is the date at Initial Program Load (IPL) time, and 'hh00' is the time at IPL.

-----

In table CRSFMT, if the entry in field KEY is *AMA*, and the entry in field FORMAT is *NTFMT* (Nortel Networks Format) or another non-Bellcore format, the default values for the options in table *AMAOPTS* are listed in the following table:

-----

*Default Schedule Values for non-Bellcore Format Options*

<b>Option</b>	<b>Default Schedule</b>
AR_BILLING	OFF
AUDIT	PERIODIC yymmdd 0000 24 HRS
CALL_FWD	OFF
CCSADATA	OFF
CDAR	OFF
CDRDUMP	OFF
CDRLONGCALL	PERIODIC
CDRSYNC	OFF
CHG411	OFF
CHG555	OFF
CIDSUSPAUD	OFF
COIN	OFF
DA411	OFF
DA555	OFF
ENFIA_B_C	ON
FREECALL	OFF
HIGHREV	OFF
INWATS	OFF
LNID	OFF
LOGAMA	OFF
LOGOPT	OFF
LONGCALL	PERIODIC yymmdd hhmm 1 HRS
LUSORIG	OFF
LUSTERM	OFF
OBSERVED	OFF
OCCOVFL	PERIODIC yymmdd 2300 1 HRS
OCCTERM	ON
OUTWATS	OFF
OVERFLOW	OFF
STORE_CALLREF	OFF
TIMECHANGE	OFF

```

TRACER                OFF
TRKID                 OFF
TWC                   OFF
UNANS_LOCAL           OFF
UNANS_TOLL            OFF
UNANS_TOPS            OFF

```

**Note:** 'yyymmdd' is the date at Initial Program Load (IPL) time, and 'hh00' is the time at IPL.

In table CRSFMT, if the entry in field KEY is AMA, and the entry in field FORMAT is BCFMT (Bellcore Format), the allowable values for field AMASEL in table AMAOPTS are as listed in the following table:

*Allowable AMASEL Values for Each Bellcore Format Option*

<b>Option</b>	<b>ON</b>	<b>OFF</b>	<b>TIMED</b>	<b>PERIODIC</b>	<b>DEFAULT</b>
ACBAR_MOD_CO	X	X			X
ACBAR_STY_IN	X	X			X
AMATRKTG_ANS	X	X			X
APPEND_ISDN_CKT_ID	X	X			X
APPEND_PRI_MODULE	X	X			X
AUDIT					X
BCLID_USPAUD	X	X	X		X
BCLONGCALL				X	X
CALL_FWD	X	X	X		X
CALL_TIMECHG	X	X			X
CAPTURE_CKTSZ_UNANS	X	X			X
CAPTURE_CLASS_SERV	X	X			X
CAPTURE_COMPL_CODE	X	X			X
CAPTURE_INAP_CPC	X	X			X
CAPTURE_SAT_IND	X	X			X
CCBS_BILLING	X	X			X
CCSADATA	X	X			X
CDAR	X	X	X		X
CDAR_EXTENDED	X	X			X
CDRDUMP		X		X	X
CDRLONGCALL	X	X		X	X
CHG411	X	X	X		X
CHG555	X	X	X		X
CIDSUSPAUD				X	X
CITYWIDE	X	X			X
CLI_DELV	X	X			X
CMCICWK	X	X	X		X
CMCORIG	X	X	X		X
CMCTERM	X	X	X		X
COIN	X	X	X		X
COLL_SVC_BILL_INFO	X	X			X
CRSEQNUM	X	X			X
CRT_BILLING	X	X			X
CSMI	X	X			X
DA411	X	X	X		X
DA555	X	X	X		X
DSCWID_CONF_AUDIT		X		X	X
ENFIA_B_C	X	X	X		X
FRECALL	X	X	X		X
FTRCODE	X	X			X
HIGHREV		X	X		X
INTL_ICR_REQD	X	X			X
INTRASITE	X	X			X

INWATS	X	X	X		X
ISDN_ACCIND	X	X			X
ISDNBBGBILL	X	X			X
ISDNCIRCUIT	X	X	X		X
LNID		X			X
LNP_BILL_DONOR	X	X			X
LOGAMA	X	X	X		X
LOGOPT		X		X	X
LOGTEST	X	X			X
LONGCALL				X	X
LUSORIG	X	X	X		X
LUSTERM	X	X	X		X
MC611_80005	X	X			X
MWIC_AUDIT		X		X	X
OBSERVED	X	X	X		X
OCCOVFL	X	X	X	X	X
OCCTERM	X	X	X		X
OUTWATS	X	X	X		X
OVERFLOW		X		X	X
RECORD_DIALED_DIGS	X	X			X
RECORD_LPSI	X	X			X
RECORD_NATIVE_LSPI	X	X			X
RECORD_RESOLD_LSPI	X	X			X
RECORD_SIGNAL_LSPI	X	X			X
RECORD_TRUNK_LSPI	X	X			X
RECORD_UNBUND_LSPI	X	X			X
SAID_MOD_SUPPR	X	X			X
STORE_CALLREF	X	X			X
STORE_LRN	X	X			X
SUSP	X	X	X		X
TIMECHANGE	X	X			X
TRACER		X		X	X
TRKID		X			X
TWC	X	X	X		X
U3WC	X	X	X		X
UNANS_AIN	X	X	X		X
UNANS_LOCAL	X	X	X		X
UNANS_TOLL	X	X	X		X
UNANS_TOPS	X	X	X	X	X

-End-

In table CRSFMT, if the entry in field KEY is *AMA*, and in field FORMAT is *NTFMT* (Nortel Format) or other non-Bellcore format, the allowable values for field AMASEL in table AMAOPTS are as listed in the following table:

*Allowable AMASEL Values for Each non-Bellcore Format Option*

<b>Option</b>	<b>ON</b>	<b>OFF</b>	<b>TIMED</b>	<b>PERIODIC</b>	<b>DEFAULT</b>
AUDIT					X
APPEND_ISDN_CKT_ID	X	X			X
CALL_FWD		X			X
CCSADATA	X	X			X
CDAR		X			X
CDRDUMP		X		X	X
CDRLONGCALL	X	X		X	X
CDRSYNC		X		X	X
CHG411		X			X
CHG555		X			X
CIDSUSPAUD		X			X
COIN		X			X

DA411					X
DA555					X
ENFIA_B_C					X
FRECALL					X
HIGHREV					X
INWATS					X
LNID					X
LOGAMA					X
LOGOPT					X
LONGCALL			X		X
LUSORIG		X			X
LUSTERM		X			X
OBSERVED		X			X
OCCOVFL	X	X	X	X	X
OCCTERM	X	X	X		X
OUTWATS		X	X		X
OVERFLOW		X		X	X
SAID_MOD_SUPPR	X	X			X
STORE_CALLREF	X	X			X
TIMECHANGE		X			X
TRACER		X			X
TRKID		X			X
TWC		X			X
UNANS_LOCAL		X			X
UNANS_TOLL		X			X
UNANS_TOPS		X			X

-----  
-End-

### **Datafill Sequence**

Table CRSFMT must be datafilled before table AMAOPTS.

### **Table Size**

Table size is determined by the number of options currently supported. This number is static.

### **Datafill**

The following table lists datafill for table AMAOPTS:

-----  
*Field Descriptions for Table AMAOPTS*

<b>Field</b>	<b>Subfield</b>	<b>Entry</b>	<b>Explanation</b>
OPTION		See subfield	<i>Option</i> This field consists of subfield AMAOPT.
	AMAOPT	Alphanumeric	<i>AMA Option</i> Enter one of the option values listed in the first table in this chapter.
SCHEDULE		See subfield	<i>Schedule</i> This field consists of subfields AMASEL, ONDATE, OFFDATE, SCHED, ONTIME, and OFFTIME.
	AMASEL	ON, OFF, DEFAULT, PERIODIC, TIMED	<i>AMA Selector</i> Enter one of the values listed in the fourth and fifth tables in this chapter.

- \* ON: Activate the option immediately.
- \* OFF: Deactivate the option immediately.
- \* DEFAULT: Use the default schedule for the option. The value DEFAULT never appears in table AMAOPTS, since table control replaces it with the actual default value; details are shown in the cross-reference tables in this chapter. The DEFAULT selector can be used at any time and the switch recalculates the default value if the default AMASEL value is PERIODIC.
- \* PERIODIC: Activate the option at the specified date and time and perform the activity periodically at the interval specified. Datafill subfields ONDATE and ONTIME to specify the date and time for activation, and datafill SCHED for the time intervals at which to perform the activity. For PERIODIC refinements, refer to AMASEL=PERIODIC conditional datafill table.
- \* TIMED: Activate the option between the specified dates and times. Datafill refinements ONDATE and ONTIME to activate the option, and refinements OFFDATE and OFFTIME to deactivate the option. For TIMED refinements, refer to AMASEL=TIMED conditional datafill table.

-----  
 -End-

## AMASEL = PERIODIC

If the entry in subfield AMASEL is PERIODIC, datafill refinements ONDATE, ONTIME, SCHED, TV, and TU as described in the following table:

-----  
 Field Descriptions for Table AMAOPTS

Field	Subfield	Entry	Explanation
	ONDATE	0 to 9 (6 digits)	<i>Activation on Date</i> Enter the year, followed by the month, followed by the day (yyymmdd) on which the activation of the option is set to ON. For example, 821105.
	ONTIME	0 to 9 (4 digits)	<i>Activation on Time</i> Enter the hour, followed by the minute (hhmm) on which the activation of the option is set to ON. For example, an ON time of 1:45 P.M. is entered as 1345.

SCHED	See subfields	<i>Periodic Schedule</i> This field consists of subfields TV and TU.
TV	0 to 255	<i>Time Value</i> Enter the time value for periodic scheduling. For example, an entry of 9, activates the option for the period of time units selected in subfield TU.
TU	HRS, MINS, or SECS	<i>Time Unit</i> Enter the time unit for the time value selected in subfield TV.

-----  
**Note:** Years 82 to 99 mean 1982 to 1999, while years 00 to 81 mean 2000 to 2081.  
 Dates that are past cannot be entered in this field by the commands CHANGE or ADD.  
 -----

-End-

## AMASEL = TIMED

If the entry in subfield AMASEL is `TIMED`, datafill refinements `ONDATE`, `ONTIME`, `OFFDATE`, and `OFFTIME`, as described in the following table:

-----  
*Field Descriptions for Table AMAOPTS*

<b>Field</b>	<b>Subfield</b>	<b>Entry</b>	<b>Explanation</b>
	ONDATE	0 to 9 (max of 6 digits)	<i>Activation on Date</i> Enter the year, followed by the month, followed by the day (yyymmdd) on which the activation of the option is set to ON. For example, 821105.
	ONTIME	0 to 9 (max of 4 digits)	<i>Activation on Time</i> Enter the hour, followed by the minute (hhmm) on which the activation of the option is set to ON. For example, an ON time of 1:45 P.M. is entered as 1345.
	OFFDATE	0 to 9 (max of 6 digits)	<i>Activation off Date</i> Enter the year, followed by the month, followed by the day (yyymmdd) on which the activation of the option is set to OFF. For example, 821106.
	OFFTIME	0 to 9 (max of 4 digits)	<i>Activation off Time</i> Enter the hour, followed by the minute (hhmm) on which the activation of the option is set to OFF. For example, an OFF time of 11:00 P.M., is entered as 2300.

-----  
**Note:** Years 82 to 99 mean 1982 to 1999, while years 00 to 81 mean 2000 to 2081.  
 Dates that are past cannot be entered in this field by the commands CHANGE or ADD.  
 -----

-End-

## **Datafill Example**

The following example shows sample datafill for table AMAOPTS.

---

### **Datafill Example for Option CCBS\_BILLING in Table AMAOPTS**

---

*Example of a MAP display:*

OPTION	SCHEDULE
CCBS_BILLING	ON

---

---

### **Datafill Example for Table AMAOPTS**

---

*Example of a MAP display:*

OPTION	SCHEDULE
ACBAR_MOD_CO	OFF
RECORD_SIGNAL_LSPI	OFF
INTL_ICR_REQD	ON

---

---

### **Datafill Example for Option LNP\_BILL\_DONOR in Table AMAOPTS**

---

*Example of a MAP display:*

OPTION	SCHEDULE
LNP_BILL_DONOR	ON

---

---

### **Datafill Example for Option RECORD\_TRUNK\_LSPI in Table AMAOPTS**

---

*Example of a MAP display:*

OPTION	SCHEDULE
RECORD_TRUNK_LSPI	ON

---

---

### **Datafill Example for Option NTAI in Table AMAOPTS**

---

*Example of a MAP display:*

OPTION	SCHEDULE
NTAI	OFF

---

---

**Datafill Example for Option STORE\_CALLREF in Table AMAOPTS**

---

*Example of a MAP display:*

OPTION	SCHEDULE
STORE_CALLREF	ON

---

---

**Datafill Example for Option STORE\_LRN in Table AMAOPTS**

---

*Example of a MAP display:*

OPTION	SCHEDULE
STORE_LRN	ON

---

---

**Datafill Example for Option CAPTURE\_CKTSZ\_UNANS in Table AMAOPTS**

---

*Example of a MAP display:*

OPTION	SCHEDULE
CAPTURE_CKTSZ_UNANS	ON

---

---

**Datafill Example for Option TRMTID\_CAPTURE in Table AMAOPTS**

---

*Example of a MAP display:*

OPTION	SCHEDULE
TRMTID_CAPTURE	ON

---

---

**Datafill Example for Options CAPTURE\_CLASS\_SERV, CAPTURE\_COMPL\_CODE, and CAPTURE\_SAT\_IND in Table AMAOPTS**

---

*Example of a MAP display:*

OPTION	SCHEDULE
CAPTURE_COMPL_CODE	ON
CAPTURE_CLASS_SERV	ON
CAPTURE_SAT_IND	ON

---

## **Error Messages**

The following explains error messages that can occur when you attempt to datafill table AMAOPTS.

### **Message:**

ERROR: LNP Billing options may not be activated unless Software Optionality Control option LNP00200 is ON.

**Explanation:** SOC option LNP00200 is in the IDLE state.

**User Action:** Activate SOC option LNP00200. Activate the LNP billing option.  
(again)

---

# Nortel DMS-100 AMA Log Reports

---

## AMA100

### Explanation

The Automatic Message Accounting (AMA) subsystem generates AMA100 when an AMA process changes status to TRAP or DEATH. When an AMA process like AMAPROC, AMAUDITP, or AMAEI changes status, the AMA log shows the changes. Do not suppress the AMA100 log. The AMA100 log indicates the status of active billing functions in the switch.

### Format

The log report format for AMA100 is as follows:

```
AMA100 mmmdd hh:mm:ss ssdd INFO AMA PROCESS STATUS CHANGE
PROCESS NAME = aaaaa
STATUS = <status>
RECREATE COUNT= nnnnnn
```

### Example

An example of log report AMA100 follows:

```
AMA100 JAN02 00:22:53 6401 INFO AMA PROCESS STATUS CHANGE
PROCESS NAME= SMDR
STATUS= PROCESS
RECREATE COUNT= 1
```

### Field Descriptions

The following table describes each field in the log report:

Field	Value	Description
INFO AMA PROCESS STATUS CHANGE	<i>constant</i>	Indicates a process status change.
PROCESS NAME	<i>constant</i>	Identifies the status of the processor.
STATUS	PROCESS, TRAP, DEATH	
RECREATE COUNT	PROCESS, TRAP, DEATH	Identifies the recreated number.

-End-

## Action

The status of each process determines the next step. Follow the directions of the last AMA100 log received for a stream.

<u>Status</u>	<u>Action</u>
PREPROCESS IS DEAD	<p>The process listed has trapped more than five times in five minutes. If the stream is not the AMA stream, any mechanism will automatically start the process. You can use the AMARESTART command to restart the stream. If it is the AMA AUDIT process then you must pass the NIL stream to the AMARESTART command.</p> <p><b>Note:</b> If the stream is AMA, the AMARESTART command works. If the stream is not AMA, the stream restarts automatically within 10 minutes.</p>
AMA: RECORDING SHUTDOWN	<p>This only occurs with the AMA DADDY process name. If any call recording is expected in the switch, this log indicates these expectations are not fulfilled. This should only occur after a RESTART. If this condition occurs in an active office, contact the next level of maintenance.</p>
CREATE ATTEMPT FAILED	<p>A recording stream trapped and was recreated successfully. If additional AMA100 logs are not received for that process, monitor the condition.</p>
SBA BUFFERING RECORDS	<p>If recording to an SDM this log shows the recording stream has died and the stream has been restarted. This log will be produced with a recreate count of 1 showing that the SBA application is accepting records to be recorded to the SDM. There will be a total of six AMA100 logs incrementing the recreate count. The first will appear once the stream has been started successfully after dieing. Two more logs will appear within the next four minutes and three more every ten minutes for a total of six logs in thirty-four minutes.</p>
All Others	<p>Another log that reports what you did to correct the event must follow any other status. Any other status is an intermediate status.</p>

-End-

**Associated OM Registers:** There are no associated Operational Measurement (OM) registers. (*all AMA reports*)

**Additional Information:** There is no additional information. (*all AMA reports*)

---

# Nortel DMS-100 AMA Log Reports

---

## AMA112

### Explanation

The Automatic Message Accounting (AMA) subsystem generates this report at the intervals defined in table AMAOPTS. The AMA112 log lists the calls in progress for longer than the period defined in OFCENG. This report is not generated if there are no long duration calls.

### Format

The log report format for AMA112 is as follows:

```
UKC21071CG AMA112 mmmdd hh:mm:ss ssdd INFO LONG DURATION CALL
  ORIG = <clli>
  TERM = <clli>
  CLG NO = <dn>
  CLD NO = <dn>
  CONNECT_TIME = day/hh:mm:ss
  REPORT COUNT = nnn
  ANSWERED = Y/N
```

### Example

An example of log report AMA112 follows:

```
UKC21071CG AMA112 DEC19 12:00:03 2112 INFO LONG DURATION CALL
  ORIG = CKTKNGA201BTWBE3
  TERM = LEN HOST 50 1 01 00 DN 8114005
  CLG NO = 1628798004
  CLD NO = 123456789123456789123456789123
  CONNECT TIME = 185/11:16:47
  REPORT COUNT = 324
  ANSWERED = Y
```

### Field Descriptions

The following table describes each field in the log report:

Field	Value	Description
INFO LONG DURATION CALL	<i>constant</i>	Indicates that a call has been in progress for more than the period defined in OFCENG.
ORIG	<i>character string</i>	The Common Language Location Identifier (CLLI) for the trunk group of the calling party.
TERM	<i>character string</i>	Gives the Common Language Location Identifier (CLLI) for the trunk group of the called party.
CLG NO	<i>integers</i>	Gives the Directory Number (DN) of the calling party. Left justified.

-continued-

<b>Field</b>	<b>Value</b>	<b>Description</b> (continued)
CLD NO	<i>integers</i>	Gives the DN of the called party. Right justified.
CONNECT TIME	<i>symbolic text</i>	Gives the day and time when the connection for this call was set up (as day in year/hours/minutes/seconds/milliseconds).
REPORT COUNT	0-999	Gives a sequential count for each series of reports.
ANSWERED	Y (yes), N (no)	Indicates whether the call was answered or not.

-End-

### **Action**

Maintenance personnel must make sure that the specified trunks work. If the call is correct and in talking state, notify the downstream processing personnel and give details of the call.

---

# Nortel DMS-100 AMA Log Reports

---

## AMA114

### Explanation

The Automatic Message Accounting (AMA) subsystem generates report AMA114 when an AMA rotate entry forms. The rotate entry goes into the AMA buffer, which goes into the AMA file. The count in AMA114 should match the count in the DIRP101 file rotation logs.

### Format

The log report format for AMA114 is as follows:

```
AMA114 mmmdd hh:mm:ss ssdd INFO FILE ROTATION ENTRY
  ROTATION TYPE = <rottype>
  STREAM NAME = <stream>
  RECORD COUNT = <reccount>
  CALL COUNT = <recnumber>
```

### Example

An example of log report AMA114 follows:

```
*** AMA114 DEC19 12:00:00 5383 INFO FILE ROTATION ENTRY
  ROTATION TYPE = OUTGOING-EMERGENCY
  STREAM NAME = AMA
  RECORD COUNT = 40
  CALL COUNT = 40
```

### Field Descriptions

The following table describes each field in the log report:

Field	Value	Description
INFO FILE ROTATION ENTRY	<i>constant</i>	Indicates an AMA transfer.
ROTATION TYPE	INCOMING	Indicates that the file rotation entry is an incoming transfer record.
	OUTGOING	Indicates that the file rotation entry is an outgoing transfer record.
	OUTGOING-EMERGENCY	Indicates that the file rotation entry is an outgoing/emergency transfer record.
STREAM NAME	<i>symbolic text</i>	Indicates the record stream set in table CRSFMT. Examples are AMA, SMDR, and CDR.

-continued-

<b>Field</b>	<b>Value</b>	<b>Description</b> (continued)
RECORD COUNT	0-99999999	Indicates the count of the eight-digit numeric record that enters a just-closed AMA file. This count should be identical to the count in the DIRP101, File Rotation Logs.
CALL COUNT	0-99999999	Indicates the call count is an eight-digit numeric field. The field shows the number of records produced from call processing activities that generate records. Does not include transfer records, time change records, or restart indication records. AMA114 generates when file restoration occurs for recording streams found in table CRSFMT.

-End-

### **Action**

Order up another standby AMA device.

---

# Nortel DMS-100 AMA Log Reports

---

## AMA117

### Explanation

The Automatic Message Accounting (AMA) subsystem generates AMA117 according to the schedule that appears in table AMAOPTS. Log report AMA117 provides the current state of the AMA options. Table AMAOPTS controls the AMA options.

### Format

The log report format for AMA117 is as follows:

```
AMA117 mmmdd hh:mm:ss ssdd INFO AMA_OPTIONS
AUDIT:          <statxt>
LOGAMA:         <statxt>
LOGOPT:         <statxt>
LONGCALL:      <statxt>
TRACER:        <statxt>
SST:           <statxt>
DA411:         <statxt>
CHG411:        <statxt>
DA555:         <statxt>
CHG555:        <statxt>
UNANS:         <statxt>
TRKID:         <statxt>
```

### Example

An example of log report AMA117 follows:

```
AMA117 JUL14 23:56:00 4721 INFO AMA_OPTIONS
AUDIT:          PENDING
LOGAMA:         ACTIVE
LOGOPT:         ACTIVE
LONGCALL:      PENDING
TRACER:        ACTIVE
SST:           ACTIVE
DA411:         INACTIVE
CHG411:        INACTIVE
DA555:         INACTIVE
CHG555:        INACTIVE
UNANS:         ACTIVE
TRKID:         INACTIVE
```

## **Field Descriptions**

The following table describes each field in the log report:

<b>Field</b>	<b>Value</b>	<b>Description</b>
INFO AMA_OPTIONS	<i>constant</i>	Indicates that the AMA option and current state of the AMA option follow.
AUDIT	ACTIVE	Indicates the option is active.
	PENDING	Indicates the option is not active. Table AMAOPTS specifies the date and time the subsystem activates the option.
	INACTIVE	Indicates the option is not active.
LOGAMA		Refer to preceding AUDIT.
LOGOPT		Refer to preceding AUDIT.
LONGCALL		Refer to preceding AUDIT.
TRACER		Refer to preceding AUDIT.
SST		Refer to preceding AUDIT.
DA411		Refer to preceding AUDIT.
CHG411		Refer to preceding AUDIT.
DA555		Refer to preceding AUDIT.
CHG555		Refer to preceding AUDIT.
UNANS		Refer to preceding AUDIT.
TRKID		Refer to preceding AUDIT.

-End-

## **Action**

There is no action required. The operating company can use this report to make sure the AMA recording options are in the required state. To change the states of options, the operating company can adjust the tuples in table AMAOPTS.

---

# Nortel DMS-100 AMA Log Reports

---

## AMA118

### Explanation

The Automatic Message Accounting (AMA) subsystem generates AMA118 to indicate the status of AMA options. Set option LOGOPT in table AMAOPTS to control the generation the AMA118 log. The user can set LOGOPT so that the system does not generate the AMA118 log. The user can set the LOGOPT so that the system generates the AMA118 log every 24 hours, or at chosen intervals. When LOGOPT is in the default configuration, the system does not generate this log report.

### Format

The log report format for AMA118 is as follows:

```
AMA118 mmmdd hh:mm:ss ssdd INFO AMA_OPTIONS
  <AMA option> : <status>
```

### Example

An example of log report AMA118 follows:

```
AMA118 APR13 12:00:00 0615 INFO AMA_OPTIONS
  ACBAR_MOD_CO:      INACTIVE
  ACBAR_STY_IN:      INACTIVE
  APPEND_ISDN_CKT_ID: ACTIVE
  AUDIT:             ACTIVE
  BCLID_USPAUD:      ACTIVE
  BCLONGCALL:        ACTIVE
  CALL_FWD:          ACTIVE
  CALL_TIMECHG:      INACTIVE
  CCSADATA:          INACTIVE
  CDAR:              INACTIVE
  CDRDUMP:           INACTIVE
  CDRLONGCALL:       INACTIVE
  CDARSYNC:          INACTIVE
  CHG411:            INACTIVE
  CHG555:            ACTIVE
  CIDSUSPAUD:        ACTIVE
  CITYWIDE:          INACTIVE
  CMCICWK:           INACTIVE
  CMCORIG:           INACTIVE
  CMCTERM:           INACTIVE
  CMCTERM:           INACTIVE
  COIN:              ACTIVE
  CRSEQNUM:          INACTIVE
  CSMI:              ACTIVE
  DA411:             ACTIVE
  DA555:             ACTIVE
  DSCWID_CONF_AUDIT: ACTIVE
  ENFIA_B_C:         ACTIVE
  FRECALL:           INACTIVE
  FTRCODE:           INACTIVE
  HIGHREV:           INACTIVE
  INTRASITE:         INACTIVE
```

```

INTRASITE:          INACTIVE
INWATS:            ACTIVE
ISDNCIRCUIT:      INACTIVE
LOGAMA:           ACTIVE
LOGOPT:           ACTIVE
LOGTEST:          INACTIVE
LONGCALL:         ACTIVE
LUSORIG:          ACTIVE
LUSTERM:          ACTIVE
OBSERVED:         ACTIVE
OCCTERM:          ACTIVE
OUTWATS:          ACTIVE
OCCOVFL:          ACTIVE
OVERFLOW:         INACTIVE
SAID_MOD_SUPPR:   ACTIVE
SUSP:             INACTIVE
TIMECHANGE:       ACTIVE
TRACER:           INACTIVE
TWC:              ACTIVE
U3WC:             ACTIVE
UNANS_AIN:        INACTIVE
UNANS_LOCAL:      ACTIVE
UNANS_TOLL:       ACTIVE
UNANS_TOPS:       ACTIVE

```

## **Field Descriptions**

The following table describes each field in the log report:

<b>Field</b>	<b>Value</b>	<b>Description</b>
INFO AMA_OPTIONS	<i>constant</i>	Indicates a report of status of options in table AMAOPTS.
<AMA option>		Identifies the option in table AMAOPTS.
<status>	ACTIVE	Indicates activation of the option in table AMAOPTS.
	INACTIVE	Indicates deactivation of the option in table AMAOPTS.
	PENDING	Indicates the option scheduled in table AMAOPTS not activated.

-End-

---

# Nortel DMS-100 SOS Script Overview

---

## Abstract

This document is designed to give the reader a quick-and-dirt view of the SOS Command Interpreter (CI for short) and execs on the DMS-100 switch. It is not designed to replace reading of the pertinent DIS documents. The document covers the basic CI commands as well as numerous examples of using SOS execs, and many of the "secret" features and Murphitic surprises awaiting the unwary.

The following conventions are used in this manual:

Commands as you would type them in are always shown in CAPITALS, preceded with a greater-than: ">". If there is an optional parameter, it will be shown in braces: "{blah}". Comments are indicated out to the right of a line and will start with a double-dash: "--".

You should not enter the comments. Thus, the line:

```
>READ OPERPROF      -- start up the profile
```

Means to enter the command "READ OPERPROF". You do not enter the ">" or the comment "-- start up the profile" at any time.

Parameters which are variable values are usually indicated with the greater-than and less-than and in lower case letters: "<blah-blah>".

For example:

```
>QFLAG <user-sys>
```

The possible values for user-sys are given by the use of braces. For example:

Where, <user-sys> = {ATV, AUTOVON, D250, XYZ, etc.}

Here, the term "etc" stands for "et cetera" or "and so forth" and is *not* an option. Usually, a help command will be available to explain what the legal options are.

In this document, I always use <BREAK> to indicate the break key.

I will also use the "^" in place of the caret.

## Overview

To many the mere word (acronym) SOS evokes terror, disgust, yea even a tight feeling in the left shoe. However, we must be of strong will for (though nasty the subject be) many must use SOS.

The perspective of this paper is not so much the operation of SOS itself (of which much has been written and said), but on how to use the SOS features to implement "programming" execs. Thus, if you want to know how the scheduler works, or how to interface your CIBINCOMs to SOS or CIP, look elsewhere. There are several good DIS documents and a couple of video tapes on that subject.

Thus, we are only concerned with the rather nasty "computer" type operations -- not the operation of SOS as an operating system. We are like BASIC programmers trying to print the first ten integers:

(BASIC)	(SOS)
10 FOR I = 1 TO 10	1 -> II
20 PRINT I	REPEAT 10 (PRINT II; (II+1)->II)
30 NEXT I	

### Logging In and Getting Started

When you log in, the SOS creates a user area of memory within which your execs and files will reside. Also, when you log out, that area of memory continues to exist. Only if you UNPERMIT a user (i.e., erase the user log on) will you wipe out files, etc. (Of course, if the switch is re-booted during your absence, then those files are lost also.)

To log on:

- Find a terminal, turn on the power, and...
- Hit the <BREAK> key. On some terminals, this is marked "BREAK" (easy), on others it is marked "INT" (for interrupt), on others, you hold down a function key (<FUNC>) and then hit <INT>, and on still others, it is <F5>. All designed to confuse the novice.
- You should get a question mark (also known as "hook"): ?
- Type in the word: LOGIN
- When prompted for the name and password, enter them on one line (unless you have "enhanced password security"):

```
--- Enter user name and password:
>ALPHA ALPHA          -- already permitted

--- Enter user name:    -- security version
>ALPHA

--- Enter password:
>ALPHA
```

Of course, if the user is not permitted, you may need to go to the operator console and enter:

```
>PERMIT ALPHA ALPHA 2 5000 ALL
```

or something like that.

### A Few Useful Keys

You may have noticed by now that CI doesn't like the arrow keys. With that in mind, here are some

substitute key sequences, as well as some other handy keystrokes. Like in any other ASCII environment, "<CTRL> H" means "hold down the control key and hit H."

- <CTRL> U -- Erase contents of line
- <CTRL> I -- Go into insert mode
- <CTRL> X -- Exit insert mode
- <CTRL> E -- Erase to end of line
- <DELETE> -- Delete character under cursor (destructive backspace key on VT220s, etc.)
- <CTRL> F -- Skip to next character
- <CTRL> H -- Backspace (also <BS> on VT100s, or <F12> on VT220s, VT320s, NT220s, etc.)
- <CTRL> S -- Stop scroll
- <CTRL> Q -- Resume scroll
- <CTRL> G -- Bell

**Note:** If you're getting a lot of "Stack Overflow" messages while messing around on the switch, you probably need to increase your stack size. To do that, you will have to re-PERMIT yourself, with a larger number for the stacksize field (5,000 in the example above). 10,000 is the limit. That's what I always use.

### The SOS Editor (Introduction)

Well, since there are files on the system, we need some way of creating and changing them -- enter the SOS editor. It is a little primitive (not full-screen) -- however, we can do a lot with it by using SOS macros within the editor. How?? Read on.

To proc it up:

```
>EDIT myfile                (or whatever)

.... edit commands ....
.....
... and to exit:

>QUIT                       -- discards input --- NO failsafe!
>FILE SFDEV                 -- files the changed file in RAM
```

### Directory Structure While in EDIT

When a user name is permitted, a permanent area of memory is allocated. This same SFDEV (RAM) is always accessed when the user logs in. The user always has read-write access to the memory. Upon entering the EDIT command, the SOS Command Interpreter (CI) creates a temporary linkage to a *read only* directory. It will be layered on top of your existing directory and you can then access the EDIT commands (INPUT, TYPE, UP, etc.).

Thus, you might try this:

```
>EDIT JUNK
NEW FILE                -- SOS tells you that the file doesn't exist
>LISTST                -- to show the directory structure...
EDITDIR      CE01,8071   RO
ALPHA        CE01,C047   RW
PROGDIR      CE01,C004   RO
SYSDIR       CE01,2002   RO
ROOTDIR      CE01,4001   RO
STDIRS       CE01,A01E   RO
```

SOS puts the copy of the `EDIT` directory on top. And even though it is Read Only (RO), you can still create macro commands in your Read/Write (RW) directory "`USERS.ALPHA`" that use the edit commands.

## **EDIT <filename>**

Let us say that we want to create a junkfile called "`MYJUNK`" to play with. We enter:

```
>EDIT MYJUNK      -- enter the editor, filename is MYJUNK
NEW FILE         -- the editor says the file is a new one
EDIT:           -- prompt to remind us we are in EDIT
>               -- prompt for the next edit command
```

We want some stuff in the file, so we use the `INPUT` command.

```
>INPUT
INPUT MODE:
>               -- Now we can enter our lines.  To exit
                -- we enter an extra carriage return.

>Mary had a little lamb.
>Sally lives here.
>>             -- i'll use ">>" for double carriage return
EDIT:
>
```

Now we want to see what we entered, so use the `TYPE`, `UP`, and `DOWN` commands:

```
>UP 1
>TYPE 33
>DOWN 11
>DOWN END      -- special variable will put us at the bottom of the file
```

Also we can change text on the *current* line with the `CHANGE` command:

```
>CHANGE 'Mary' 'Sammy'
Sammy had a little lamb.

      or

>CHANGE 'Marry' 'Sammy'
STRING NOT FOUND      -- EDIT couldn't find the string
>
```

We must always use the apostrophe ( `'` ), so things sometimes get a bit messy:

```
>INPUT
INPUT MODE:
>Don't you know?
```

```
>>
```

```
>CHANGE 'Don't' 'Didn't'  
Didn't you know?
```

```
>CHANGE ''t' '-t'  
Didn-t you know?
```

The general rule is to use an extra apostrophe for the one in the text.

Or we can delete the whole mess with the `DELETE` command. The `DELETE` command doesn't output anything except the new current line (if it changes).

Both `DELETE` and `CHANGE` always refer to the current line, which can be displayed by the `TYPE 1` command. `UP` and `DOWN` move you through the text files.

`CHANGE` and `DELETE` have "until" features:

```
>DELETE 'zotto' -- Delete all lines from here until  
-- the search string "zotto" is found.  
-- If it doesn't exist then nothing is zapped.
```

```
>CHANGE GLOBAL 'a' 'A' -- Goes from current line to end of file.
```

or

```
>CHANGE 3 'a' 'A' -- change 3 times on the same line:  
Mary hAd A little lamb.  
| | |
```

The line number of the current line in the editor is stored in a special variable called `LINE`:

```
>TYPE 1  
Mary had a little lamb.  
>PRINT LINE  
1  
>DOWN 1  
Sally lives here.  
>PRINT LINE  
2
```

The current line's contents are always copied to a special variable called `LINESTR`:

```
>PRINT LINESTR  
Sally lives here.
```

`TOP` and `END` move you quickly through the file. To search for a string, we can use `DOWN` with a string:

```

>TOP      -- The editor will always reserve a dummy
          -- blank line at the top of your file.

>DOWN 'lives'
Sally lives here.  -- Or you might get an error message
                  -- STRING NOT FOUND.

>UP 'lamb'
Mary had a little lamb.  -- UP with a search string

```

Commands can be strung out on the same line with the ";" (just like SOS commands):

```

>TOP;DOWN 'here';TYPE 333  -- Go back to the top of file,
                          -- go down until the string 'here'
                          -- is found, and type the next 333
                          -- lines (or until End-of-File).

or

>TOP
>DOWN 'here'
>TYPE 333

```

By the way, there is another form of INPUT... INPUT B. This will input the data *before* the current line. Sort of handy from time to time.

```

>FILE SFDEV      -- save our file in RAM (SFDEV)
>FILE D010TEST1  -- save our file on disk volume D010TEST1

```

## EDIT Macros

Since the EDIT directory is on top of ours, which is on top of the system and program directories, we can still use SOS to create commands — and while in edit, these can be *very* powerful:

```

>EDIT MYJUNK      -- If the file already exists, we won't
                  -- get the "NEW FILE" message here.

>COMMAND CM COMMAND  -- Define a shorter version of the
                    -- command declaration command.

>CM D DOWN        -- short form for DOWN
>CM U UP          -- short form for UP
>CM T TYPE        -- short form for TYPE
>CM BOT END       -- XEDIT users like "BOT" not "END"
>CM CH CHANGE     -- short form for CHANGE
>CM CHA CHANGE    -- three-letter version
>CM DEL DELETE    -- short form for DELETE

>CM PUT1 (LINESTR->XX1)  -- store the current line in a temp variable "XX1"
>CM GET1 (INPUT XX1)    -- and input it after the current position.

```

Example:

```
>EDIT LAMBS
NEW FILE:
>INPUT
INPUT MODE:
>Mary had a little lamb.
>Sally lives here.
>>          -- extra carriage return, exits input mode

>TOP;DOWN 1
Mary had a little lamb.
>PUT1          -- XX1 now equals 'Mary had a little lamb.'
>END
>GET1          -- store it as the new last line, and...

>TOP;T 5555    -- show the file:
Mary had a little lamb.
Sally lives here.
Mary had a little lamb.
```

Neato, eh? Of course, we could write a macro to put an unlimited number of lines, but let's wait until later.

## **SOS Fundamentals**

### **What's the Difference Between NT40 and SuperNode Switches?**

This is a really broad question, but I'm going to try to sum it up in a page or so.

First, let's talk about how a switch works. A captive office is basically two parts: the network and the switch. The network is a collection of peripherals (each of which is driven by its own microprocessor). The switch is the master computer which, among other things, sets up connections between peripherals so they can talk to one another.

With this in mind, I'll try to explain the life of a simple phone call. The call comes into the switch on a trunk/line. The peripheral says, "Hey, I just got a call. I better tell the switch." So it sends a message to the switch. A little volley goes on and eventually the peripheral tells the switch what digits were dialed. The switch figures out where to send the call next and sets up the connection between the incoming and outgoing peripherals. This connection is set up in the network. From here on out, the peripherals talk to each other without any help from the switch until the call ends.

This is why, once you get a call up, you can do anything to the switch you want to and the call stays up. You can restart warm, restart cold, whatever. The peripherals never even know it.

Now that we have that out of the way, we can get on with the real question.

The NT40 is a proprietary processor developed by Northern Telecom specifically for digital telephony applications. This processor is, of course, the heart of what we call NT40 switches.

A few years ago, someone decided it would be a really neat idea to base a switch around a 68020. That's where SuperNode came from. But the differences don't end there. SuperNode is actually two computers instead of just one. (Two... two... two computers in one!) It's made up of a CM (Computing Module) and an MS (Message Switch).

One of the results of this arrangement is that images are about twice as large for SuperNodes as for NT40s, and consequently take about twice as long to boot. Enter the SLM.

SLM stands for System Load Module. An SLM consists of a Winchester disk and a magnetic-tape cartridge. The SLM is used to contain all images for a SuperNode, as opposed to the NT40 which stores images on the regular disk drives. The overall result is that the SLM tapes boot many times faster than the normal tapes, and the SLM drives leave that much more room on the regular disk drives.

The only other major difference from a user standpoint is the way you boot the two machines. NT40 switches are booted using the thumbwheel on the front of the switch. SuperNodes are booted from the Reset Terminal. The procedures are described in detail in the "Booting from Disk or Tape" section.

That about covers it. Any additional questions you have about SuperNode versus NT40 are probably covered in the *DMS-100 SuperNode System Description* (SYSDESC). If you're curious about SLMs, look a little further down in the "SLM Stuff" section.

## **A Few Things to Know Before Writing Execs**

### **SOS Data Types**

Basically, there are several unusual data types that cause all sorts of problems:

- Integers
- Strings
- Symbols
- Files

And, of course, since Protel is heavily typed, it is hard to get the computer to convert them back and forth. For some conversions there exist functions, for others not:

```
12->TWELVE          -- TWELVE is a decimal value of 12
PRINT (HEXTOSTR TWELVE) -- print the form-feed
                        or
PRINT (HEXTOSTR #0C)  -- actual hex value "#"
```

So we have a function "HEXTOSTR" to do the conversion. We'll see others later.

### **Variables & Values**

There are several types of variables in SOS and these include:

- Numeric & string variables (the usual)
- Symbols (which are strange beasts)
- Files, devices, commands, execs, etc.

You see, all entities have a partye associated with them and we can find out what that is with the `PARTYPE` built-in function:

```
>1->II
>PRINT (PARTYPE II)
1
```

We can, of course, assign values. For example:

```
>'abc' ->VARX
>32767 ->MAXINT
```

Now we come to a very important aspect. The so-called "goes-into".

### The Goes-Into "->"

To assign a variable, we use two characters instead of just 1:

```
BASIC: BBB = 33
SOS: 33->BBB
```

Now a very interesting thing occurs, and that is that the number of spaces on each side of the goes-into must be the same:

```
123-> XYZ      -- does nothing at all!!
123 ->XYZ      -- does nothing at all!!
123 -> XYZ      -- ok
123->XYZ      -- ok
123 ->  XYZ      -- ok
```

This can be tricky, so either use no spaces or at least one space on each side.

### Comparison and Relational Operators

Of course, SOS has IFs, UNTILs, and WHILEs, and what would an IF, UNTIL, or WHILE be without comparison and relational operators? Here are the comparison operators:

<u>Comparator</u>	<u>Symbol</u>
Less Than	<
Greater Than	>
Less Than or Equal	<=
Greater Than or Equal	>=
Equal	=
Not Equal	^=

These are the relational operators:

<u>Relator</u>	<u>Symbol</u>
AND	&
OR	
EXCLUSIVE OR	!

As with the goes-into, SOS is kinda picky about how many spaces are on each side of a comparator or a relational operator. The rule for comparators is the same as that for the goes-into. At least one space on both sides or no spaces on either side.

The relational operators are different, though. With them, the number of spaces on the left don't matter, but there must be at least one space on the right. Don't ask me. I just work here.

Let's look at some examples:

```
IF ((x>y) & (w=1)) THEN (...          -- do it fine.
IF ((x > y) & (w = 1)) THEN (...       -- ok by me.
IF ((x > y)& (w = 1)) THEN (...        -- no prob.
IF ((x > y)& (w = 1)) THEN (...        -- ok fine.

IF ((x > y)&(w = 1)) THEN (...  -- won't work!!!
                                (need a space after the &)
IF ((x>y) &(w=1)) THEN (...  -- won't work!!!
                                (need a space after the &)
IF ((x> y)&(w=1)) THEN (...  -- won't work!!!
                                (uneven spacing around > )
IF ((x>y)&(w =1)) THEN (...  -- won't work!!!
                                (uneven spacing around = )
```

## Numeric Variables

Numeric variables are always signed 16-bit integers. This means that the limit of an integer is between -32,768 and 32,767.

Thus, the following are examples:

```
123 -> AVAR
32767->BVAR
-32760 -> JVAR
```

All arithmetic is integer and must conform to these range restrictions.

## String Variables

String variables are always enclosed in the single-quotes mark (apostrophe):

```
'here is a string' -> MY_STRING
'Joe's pizza'->$PARLOR_MESG
```

The rule in general is that you use one extra ' in the middle of a string. Thus, the following do the following:

```
>PRINT $PARLOR_MESG
Joe's pizza

>' 'x' '->QUOTO
>PRINT QUOTO
'x'
```

The extra extra ' is required due to the fact that the first one is the delimiter. Anyway, you can fool around with it until you get the hang of it. We saw some examples of that back in the editor discussion.

## **NUMTODECSTR, NUMTOHEXSTR, HEXTOSTR, & DECSTRTONUM**

SOS is pretty flaky about consistently providing type-conversion, so here we go:

```
>123->NUMBER
>(NUMTODECSTR NUMBER) -> STRING
>(DECSTRTONUM STRING) -> SAMENUMBER
>(NUMTOHEXSTR NUMBER) -> STRING2
>PRINT NUMBER STRING STRING2 SAMENUMBER
123 123 007B 123

>PRINT (HEXTOSTR 12)
```

Would produce a CONTROL-L (form-feed in ASCII)

### **Simple SOS Commands**

The keyword for a command declaration is `COMMAND`. (Very catchy, eh?)

```
>COMMAND FF (PRINT (HEXTOSTR 12))
```

This would give us a short form of the form-feed command.

Now let us look at a fancy version of form-feed:

```
>COMMAND FF (PRINT (HEXTOSTR 12))
>PRINT '***** DEBUGGING RLT '
>DATE;PRINT ' ')
```

Notice two or three things:

A command starts with a " (" and must end with a ")".

Continuations on the next line can go on for quite a while (as long as you don't over-flow the token parm area -- 255 max when input interactively. Otherwise, 2,048 in a file).

The ";" can be used within a command just as it can be on a line for separating commands.

Now, lets look at a neat command:

```
>COMMAND NEATO ((SYMTOSTR @1)->$PARM1;
>IF ($PARM1='OK') THEN(
>PRINT '-- It's ok with me.'
>)ELSE(
>PRINT '-- Not ok.' )
```

Things get complicated fast in SOS, so let's look at it blow by blow.

Any string parm passed into a command is automatically converted to a symbol. Numeric values are not. They are left as numerics.

The IF statement can have an ELSE, but it stops scanning after the THEN is completed. That is why I put the closing ")" for the THEN on the next line. The following has *no* ELSE clause:

```
>COMMAND NOT_NEATO ((SYMTOSTR @1)->$PARM1;  
>IF ($PARM1='OK') THEN(  
>PRINT '-- It's ok with me.')>ELSE(  
>PRINT '-- Not ok. )
```

Yes, SOS strikes again! Since you closed out the THEN clause at the end of the "ok with me" line, the scanner just ignores the ELSE and its clause on the next two lines. It only makes sure that you have a final closing ")" to balance the very first one in the very first line.

Since commands give us more power, they also require more support. We will now look at some of the functions built in to help with commands.

## SYMTOSTR & STRTOSYM

As we have seen, string parms that are passed to a command are converted to a SYMBOL type of variable. So we must convert them to strings to compare or print them.

```
>COMMAND NEATO ((SYMTOSTR @1)->$PARM1;... etc.)
```

The first parm is stored in the variable @1, #2 in @2, etc. Thus, after execution of NEATO, we could print the variables where we have transferred the values:

```
>NEATO HOWDY  
>PRINT $PARM1  
HOWDY
```

We can use this to "parse" a line:

```
>COMMAND ALTER ((SYMTOSTR @1)->$IX_TABLE;  
>(SYMTOSTR @2)->$IX_CLLI;  
>(SYMTOSTR @3)->$IX_FIELD; (SYMTOSTR @4)->$IX_VAL)  
  
>COMMAND SHOW_ALT (PRINT '-- ALTER-PARMS: ' $IX_TABLE;  
>PRINT $IX_CLLI $IX_FIELD $IX_VAL)
```

And, I could use them thusly:

```
>ALTER trkgrp mda12wdtgs01 authfld $3334444  
>SHOW_ALT
```

```
-- ALTER-PARMS: trkgrp mda12wdtgs01 authfld $3334444
```

An interesting thing to note is the need for a non-numeric as the first character of field four. This is because the numeric value would not be transformed into a symbol, while a string value would!! This means that the (SYMTOSTR @4) would fail to get a proper input variable -- and give a "parameter of wrong type" message. Ah, SOS -- how wonderful. There is a way out of this by using the PARTYPE function. Since PARTYPE returns the parameter type (as an integer), we could use this in an IF statement to do the proper conversion.

Sometimes we want to concatenate strings, so we may need to do several conversions into strings, and then do the concatenations:

```
>1->II
>'TM8 '->$CKT
>$CKT+(NUMTODECSTR II) -> $TM8_SHELF
>PRINT $TM8_SHELF
TM8 1
```

Usually, I will put an extra set of parentheses around stuff to be concatenated. Also, note that string concatenation is one of the slowest things SOS does!

Well, since we can do a symbol to string, why not a string to a symbol? No sooner said than done. Study this:

```
>1->II
>33 -> VAR33
>('VAR'+(NUMTODECSTR II)) -> VAR_NAME
>22 -> (STRTOSYM VAR_NAME)
>PRINT VAR33 VAR1
33 22
```

Here's what happens:

- II gets set to 1.
- The value 33 is put into a variable called VAR33.
- A new string variable called VAR\_NAME is built up with a value of 'VAR1'.
- The value of 22 is stored in a variable called VAR1.

So we can create variables out of thin air. Sort of like an array. Consider the following:

```
>1->II
>REPEAT 10 (II -> (STRTOSYM ('NUM'+(NUMTODECSTR II)));
(II+1) -> II)
```

This will do the same as:

```
>1 -> NUM1
>2 -> NUM2
>3 -> NUM3
...
>10 -> NUM10
```

This is indeed an array. Not only that, but the thing can be used to execute using variables passed into a command!

```
>COMMAND PUT_LINE (  
>LINESTR -> (STRTOSYM ('XLINE'+(NUMTODECSTR II)))  
>(II+1)->II)  
  
>COMMAND GET_LINES (  
>REPEAT II (  
>INPUT (STRTOSYM ('XLINE'+(NUMTODECSTR II))  
>);0->II) )
```

We need only initialize the `II` value, and then we can use it in `EDIT`. So the ability to build symbols from strings is one of the most important aspects. Still...

## Symbols & Devices

Since we can build the symbol of what something is in a variable, we can pass to a variable something other than a plain old string or number. Consider:

```
>'D010TEST' -> $VOL_NAME    -- Create a string with the  
                           -- name of a disk volume in it.  
>(STRTOSYM $VOL_NAME) -> $VOL -- A symbol is created!  
>DSKUT  
>LISTVOL $VOL ALL          -- List the files on the given volume!
```

Notice, however, that were we to try and do the following...

```
>PRINT $VOL  
device                -- the same as >PRINT D010TEST  
  
>PRINT $VOL_NAME  
D010TEST              -- just what we put in the var  
  
>LISTVOL $VOL_NAME ALL  
Wrong type: <volume name> DEVICE name
```

Wherein SOS is telling us that "`$VOL_NAME`" is not in the root directory as a symbol representing a disk volume. This is one of the differences between a string and a symbol.

## RINDEX & LINDEX -- Parm Pointers

`LINDEX` and `RINDEX` tell you how many parameters appear to the left and right of your command, respectively. For example:

```
>COMMAND DOIT (PRINT (LINDEX);PRINT (RINDEX))  
  
>DOIT A B C  
0  
3
```

```
>A DOIT B C
1
2
```

You can use this to make sure that the user enters the correct number of parameters to your command, and maybe even give them a little more descriptive error message than SOS's helpful: "@ -- Parameter 1 does not exist".

Many people also use `LINDEX` to see if they should print help information. If the user typed "HELP cmd", `LINDEX` would be 1.

This brings up another interesting point about SOS. Notice that, in our second example above, the command `DOIT` was executed even though it was not the first thing in the command line (ooh, aah!). What SOS actually does is scan the line from left to right and execute the first command it finds.

Like many things in SOS, this is potentially disastrous. Say, for example, you had a command `DOIT`. Say, also for example, that you were in table `IOC` (a rather important table), positioned on a particularly important tuple (like the one that contains all the terminals). Now, say you were to type:

```
>DIOT DELETE          -- Note the misspelling :)
```

I think you get the point.

## **The SOS Commands -- Attack of the CI Commands**

Now children, are you all nice and cozy? Well, good. Once upon a time, there was a `DIS` document that described the commands that SOS allowed you to use. It was a document, and it did exist (long after the fires of Mordor had been quelled). Then, one day it up and disappeared! But, then it wasn't really very useful, so no one really noticed that it was gone (until some poor novice was scanning the `DIS` disk looking for it). So, here is a tale of SOS CI commands. (With examples, yet!)

By the way, the only existing document on SOS that is in `DIS` is the Northern Telecom Publication (NTP) 297-1001-509, *DMS-100 Family Command Reference Manual*. Use the following to get a copy of it:

```
>DIS INDEX
>PRINT N1001509 LISTING I
>CP SP PRT CLO FORM XHDP
```

(or however you print stuff at your site)

It is a bit dated, and somewhat more like a dictionary. However, I will quote a bit from it.

## **Survey of the Commands**

I couldn't decide if I should put the commands in here in logical order or in alphabetical order, so I finally decided on both! This section is the one that is logical. The others are alphabetical. By

necessity, there are some commands that I talk about quite a bit, and others that I don't really get into very much at all.

## Logging On, and Stuff Like That

Commands discussed:

- FORCEOUT
- LOGIN
- LOGOUT
- MSG
- PERMIT
- PRINT USERS
- PROFILE
- SETLOGMSG
- SHOW USERS
- UNPERMIT
- <BREAK> HT
- <BREAK> HX & HXX

## TERMDEV, LOGIN, LOGOUT, FORCEOUT

You must always make sure the terminal is configured correctly in table TERMDEV. Its baud rate, etc. should match the setting on your terminal:

```
MAP 0 8 CYB B1200 CL 1X67BC NONE N NONE ALL
```

would be the entry for a Cybernex terminal. The device name is "MAP", which is the default for the operator position. For a printer, the entry would be:

```
PRT0 0 9 KSR B1200 CL 1X67BC NONE N NONE ALL
```

For a VTxxx (VT100, VT220, etc.), the entry would be:

```
MAP 0 8 VT100 B1200 CL 1X67BC NONE N NONE ALL
```

Then, you log in. You get the switch's attention by hitting the break key. In this document, I always use <BREAK> to indicate that key.

```
<BREAK>
?          -- This is the standard prompt.
?LOGIN     -- Then, you enter LOGIN after the "?". The switch
           -- will then prompt you for your user name and
           -- password if there is no default user for your
           -- terminal.

>LOGOUT    -- Zap. You are logged out. All commands,
           -- variables, and directories that you had
           -- defined disappear.
```

Another handy command is:

```
>FORCEOUT <user>
```

Which will forcibly logout another user. There is no warning, so use it with caution. The user does get a message something like "Logged out by user".

## Detached User LOGIN

This brings us to the topic of detached users (about which I will have a whole lot to say later). If you are already logged in, you can use the `LOGIN` command with the following format:

```
>LOGIN <user> <password> <profile> <device>
```

See the discussion under the `LOGIN` command for more details.

## MSG & SETLOGMSG

`MSG` is used to send a message to another (or all optionally) user(s):

```
>MSG MAP4 'Are you ready for lunch?'
```

When we use the `MSG` command, we have to know who is logged on. We can use the `PRINT USERS` command to find out.

`SETLOGMSG` allows you to set the message that will be displayed at log on. Also, if you hit return twice in a row, you will get the log message displayed (along with a prompt showing where you are):

```
>  
>  
CI:  
BCS-18ZO RMT with TATS  AUG-12-85  No Patches  
>
```

## PRINT USERS, SHOW USERS

`PRINT USERS` -- shows all users who are logged in.

`SHOW USERS` -- shows all users who have been permitted.

## PERMIT & UNPERMIT

`PERMIT` allows you to add a user name that the switch will recognize at login. For example:

```
>PERMIT MAP3 MAP3 2 8000 ALL      -- or something like that
```

Now, someone can go to a terminal and log in as MAP3.

UNPERMIT just does the opposite of PERMIT. For example:

```
>UNPERMIT MAP3      -- erases a user name from existence
```

These have an interesting property. The user name has nothing to do with the device. However, if it happens that a user name has been permitted that has exactly the same spelling as a device name, and then you go to log on at that device --- you won't be prompted for name or password!

## PROFILE & NOPROFILE

You can set up a user profile for each login user name with the PROFILE command. You have one chance to override the default profile at login time. After you login, the very first command you enter must be NOPROFILE. Otherwise, you will automatically execute the associated login profile the first time you hit return after logging in. For example:

```
<BREAK>
?LOGIN
--- Enter user name and password:
>BETA BETA
--- User BETA logged in on 1PRT0 on Jan 31, 1993.
>NOPROFILE  -- this should disable the profile
```

## The <BREAK> Key with HT, HX and RT

If a long listing is coming out and you want to discard the output, you can hit the break key and enter "HT" after the "?". Then, if you want to look at the output again (before the output is done), you can hit break again and enter "RT" after the new "?".

These are modeled after VM/CMS's Halt Typing (HT) and Resume Typing (RT) commands. Of course, there are problems. For example, if you have a record link going to a device (e.g., RECORD START ONTO PRT0), *then even though you have entered the HT*, the listing still continues to PRT0. (It *may* be a little bit faster since the line doesn't actually have to be output to two different devices.)

To kill the current program you are running, use the break with either "HX" or "HXX". These "Halt Execution". Sometimes it takes a few moments to clear, but you will usually get a message like "User process stopped." followed by a prompt.

## Disk Stuff

Well, of course the switch has a disk drive... it's a computer, isn't it? It's used for pretty much the same things on the switch as it is on any other computer... storing junk and retrieving it fairly quickly.

In this little section, I'll discuss some of the things you can do with the disk, along with some of the pitfalls of working with the disk.

Commands discussed:

- DSKALLOC (increment)
- DSKUT (increment)
- EDIT (increment)

- COPY
- DUMP

## DSKALLOC

DSKALLOC is a handy little increment that does all the low-level stuff that make the disk drive on the switch useful.

Just like on a PC, the disk must be formatted before it's any good to anybody. This is done with the REINIT command in DSKALLOC. Within DSKALLOC, you can also partition the disk drive into several "volumes" using the ADD and DELETE commands. You can even make the volumes accessible only to certain users using the DIRADD and DIRDEL commands.

I'm not going to go into detail about DSKALLOC here, since you probably shouldn't be using it anyway. That's a job for the maintenance guys. If you're really curious, look in NTP 297-1001-526.

## DSKUT

In general, if you want to do *anything* relating to the disk (print files, create files, copy files, etc.), you need to use DSKUT.

Probably the most valuable command in DSKUT is LISTVOL. LISTVOL allows you to list the files on a given disk, which puts pointers to them in your read-write directory, which makes the files accessible to you. If you want to READ, PRINT, COPY, or EDIT a file, you have to use LISTVOL to do it.

Like the entries in SFDEV, the files on disk know who owns them. So, you can have several different files with the same name on the same disk, but SOS is able to keep them straight because it knows who owns them. If you just do a:

```
>LISTVOL D010CNTFL
```

you would get only the files that you own. If you do a:

```
>LISTVOL D010CNTFL OPERATOR
```

you would get only the files that OPERATOR owns. If you do a:

```
>LISTVOL D010CNTFL ALL
```

you would get all the files on the volume.

This brings up an important point. If you do a LISTVOL and get more than one file listed with the same name, SOS will only remember the last one that popped up. So be careful.

This gets to be even more of a pain, because if you file a new version of a file that was already there, the older version *is not erased*. It's still sitting out there, and if you do a LISTVOL, it will show

up as an "\*\*\* older duplicate \*\*\*", and these can be a real pain to get rid of. Of course, if you do a LISTVOL and the last copy of the file that pops up is an "\*\*\* older duplicate \*\*\*", that's the only one SOS knows about.

But, enough about LISTVOL. There are several other little goodies in DSKUT. You can get information about a volume or a file (SHOWVOL and SHOWFL) and erase files (ERASEFL). If you're messing with the boot volume, you can even mess with the current image (SETBOOT, SHOWBOOT, and CLEARBOOT).

## Getting a File on the Disk

There are only three ways I know of to get a file on the disk. The first is to COPY it there. You can copy files from SFDEV, from other disk volumes, or from tape.

The second way is to EDIT a file and FILE it on disk. EDIT is the SOS file editor. You start off by EDITing a file that doesn't exist, putting some stuff in it, and issuing the FILE command:

```
>FILE D010CNTFL
```

The third way is to take an image using the DUMP command.

## Tape Stuff

Of course, if you really want your data to be safe, you'll back it up to tape every once in a while. (Maintenance has this nasty habit of reformatting disks when they develop problems.)

In this section, I'll briefly discuss how to get stuff onto and off of a tape.

Commands discussed:

- ERATAPE
- MOUNT
- DEMOUNT
- LIST
- COPY
- DUMP

## Mounting the Tape

Of course, the first thing you have to do when using a tape is mount it. To do that, you go to the tape drive, put your tape on the wheel, and manually thread the tape onto the takeup reel. (Autoloading? What's that?) There should be a diagram on the tape drive to show you how to do that.

You then hit the LOAD button. The tape will spin for a few seconds and stop. The LOAD light will come on when it stops. You then hit the ONLINE button.

You've now done all the manual work involved in mounting the tape. The switch still doesn't know it's there, though. You have to issue the MOUNT command... like this:

```
>MOUNT 0  
or
```

```
>MOUNT 1
  or
>MOUNT 10
  depending on the name of your tape drive
```

That's the simple form. We'll get to the more complicated forms in a minute.

Anyway, the switch now knows that there's a tape there. You're ready to go.

### Getting Files from Tape to Disk

To get files from tape to disk, you mount the tape just as outlined in the "Mounting the Tape" section. The next step is to get all the files on the tape into your symbol table. This requires the use of the `LIST` command:

```
>LIST T0
```

Notice that the tape drive is now prefaced with a "T". Why? Who knows. It's just one of those Walter Cronkite things: "That's the way it is."

Be forewarned: unlike `LISTVOL` or `LISTSF`, no filenames will appear on the screen when you use the `LIST` command.

Anyway, once you've done that, you use the `COPY` command to pull the files off and put them where you want them:

```
>COPY FILE1 D010CNTFL
>COPY FILE2 SFDEV
```

See? No problem.

### Getting Files from Disk to Tape

Getting files from tape to disk is just a little more complicated at first. Specifically, you have to erase the tape before you mount it:

```
>ERASTAPE 0
>MOUNT 0
```

Now, you use the `COPY` command, much the same as you would to copy from tape to disk:

```
>COPY FILE1 T0
>COPY FILE2 T0
```

### Demounting the Tape

This is pretty easy. To demount the tape, just type:

>DEMOUNT TO

The tape will rewind back to the `LOAD` position. To take the tape off, hit the `ONLINE` button, then hit the `REWIND` button. No problem.

The only trick is that the `DEMOUNT` command has to be issued at the same terminal where you issued the `MOUNT` command.

There are a couple of special cases, though. If you forget to do the `DEMOUNT` command, you can't screw anything up. When you go to the tape drive and hit the `ONLINE` button, it'll just ignore you.

The other case is if you do a restart without `DEMOUNTING` a `MOUNTED` tape. In this case, the switch no longer knows the tape is mounted, so just go over and hit the `ONLINE` and `REWIND` buttons.

### Booting from Tape and Imaging to Tape

Yes, you can boot from tape and take an image to tape. Both are discussed in "Images" section. Just thought you'd like to know.

### SLM Stuff

What's an SLM? Well, it's a combination disk drive and tape drive. Basically, someone decided it would be neat to use those new high-density cartridge tapes, and someone else decided it would be a good idea to have a disk drive used strictly for storing images, so they put them both in one box and called it an SLM. To see one, just open the cabinet doors on your nearest SuperNode. The SLMs are the things with the big slots in them toward the bottom. If you don't see it, go around to the other side.

I'm going to try to explain how you use an SLM in terms of how you use a regular disk drive. Ok, you know that switches have one regular disk drive per side and that those drives are identified as `D000` and `D010`. There can be several logical volumes on a single drive (`IMAGE`, `CNTFL`, `PMLOADS`, etc.), so to identify a particular volume on a particular drive, you do something like `D010CNTFL` or `D000IMAGE`.

Well, SLMs work kind of the same way. Each SLM contains two devices: a disk drive and a tape drive. The disk drive is `S00D` or `S01D` and the tape drive is `S00T` or `S01T`. SLM disks also have volumes. However, the only volume I've ever seen on an SLM disk in the lab is `IMAGE`. So, you have `S01DIMAGE` and `S00DIMAGE`.

Now, SLMs are supposed to be the only things you use for images and booting on a SuperNode. They're the only things you can `DUMP` to or `\BOOT` from. The disks act pretty much like regular disks, though. You can `COPY` things to or from one, `EDIT` things and `FILE` them on one, and just generally treat one like another disk drive. The exception to that is that `DSKUT` doesn't access an SLM. The corollary to `DSKUT` for an SLM is `DISKUT`. `DISKUT` will be explained a little further down.

The SLM tape drive is an interesting beast. It doesn't behave much like the reel tape drives. It's used strictly for backing up files and booting. You can't even use the `COPY` command on it. To get files onto it, you have to use the `DISKUT BACKUP` command, and to get them off, you have to use the `DISKUT RESTORE` command.

I will spend the remainder of this section talking about how to use the SLM disk and tape drives.

Commands discussed:

- `DISKADM` (increment)
- `DISKUT` (increment)
- `DUMP`

## DISKADM

`DISKADM` is the SLM version of `DSKALLOC`. From within `DISKADM`, you can create and delete volumes on an SLM disk, display disk and volume information, format the disk, and reinitialize a volume.

As you can see, `DISKADM` is a pretty hazardous little utility. If you really think you'll ever need this thing, look in NTP 297–1001–509. It's the only thing I've found that talks about `DISKADM`.

## DISKUT

`DISKUT` performs all the same basic functions on an SLM as `DSKUT` performs on a regular disk. In addition, it has a couple of commands to `MOUNT` and `DEMOUNT` those tape cartridges (`INSERTTAPE` and `EJECTTAPE`).

One of the largest differences between the operation of `DSKUT` and `DISKUT` is the way they handle images. On an NT40 switch, the `IMAGE` volume of a drive is assumed to contain only images, so when you want to see what images are available, you just do a `LISTVOL` on the `IMAGE` volume of the disk. Not so with a SuperNode.

An SLM contains all images on a SuperNode, but those images can be spread out over multiple volumes on the SLM. So, to keep everything straight, each SLM device has two Image Tables Of Contents (ITOC) associated with it. One ITOC contains an entry for every MS load, and the other contains an entry for every CM load. When you dump an image, entries are automatically generated in the appropriate ITOCs. But, if you were to put an image on an SLM from a reel tape (yuck), you would have to use the `SETBOOTFL` command to update the ITOC.

The reason for having an ITOC is so that the switch can look for other loads to boot if it can't boot the `ACTIVE` load. So, if you try to boot the switch from SLM and the image fails its reload restart, the switch will go down the ITOC list looking for a load to boot. If you want to know the full search order the switch goes through, look at `SLMUG` in `DIS`. It goes through the whole thing.

You can also control which image is booted with `SETBOOTFL` and `CLEARBOOTFL`, and see which image is current with `LISTBOOTFL`.

If you want to manipulate the cartridge tape drive, you can use `BACKUP` to back up either single files or whole volumes from the SLM disk, and then `RESTORE` to get them back off again. This brings up an interesting point. When you `BACKUP` an entire volume, everything is put in one file on the tape. If you want to see the names of the individual files, you have to use the `BVOL` option of the `LISTFL` command in `DISKUT`.

You can see what volumes are available with `LISTVOLS`, and then there are some other commands to manipulate files. (`RENAMEFL`, `DELETEFL`, and `CLEARVOL`)

I could give you more information here, but since it's already in the dictionary section, I'll just refer you there.

## DUMP

The `DUMP` command on a SuperNode switch has a few more options than the `DUMP` command on an NT40. In short, since there is more than one CPU on a SuperNode, you can dump each one individually. If you do a:

```
>Q DUMP
```

you will see all the extra options at the end, after the `VERBOSITY` option.

These are optional parameters, so if you don't specify anything the default is `TOTAL`, which will dump the active CM and MS.

"How will it differentiate between the CM and MS loads on the disk?" you ask. It will simply tag "\_CM" and "\_MS" onto the ends of the respective images.

## Images

Just what the heck is an image, anyway? Well, an image is a bit-for-bit copy of the contents of memory on the switch at a given point in time, and since all the switch's software and data is kept in memory, that can be quite a bit of stuff.

Images can reside on tape or on disk, and they're put there with the `DUMP` command. They are reloaded by use of the thumbwheel on the switch.

Basically, what happens when you use the `DUMP` command is that all the switch's used memory is copied to tape or disk in such a manner that the switch is able to reload the data in exactly the same place later on. It will then continue on about its business, starting at the point immediately following the point where the image was taken.

Usually, you will want to take an image right after you've made some major changes in the datafill on the switch (so that you don't lose it), or right before you do something that could mess the switch up so bad that you'd have to reload from disk or tape (like loading a module that could cause a restart loop).

"Ok, so now I know how to take an image. But how do I boot with it once I've taken it?"

## Booting from Disk or Tape

This is one item that's as different as night and day between NT40 and SuperNode. Take your pick.

## Booting the NT40

Booting the NT40 from disk or tape involves using that funny little thumbwheel on the front of the switch. (Scary, huh!) The general formula is:

- Flip the "ACT/DEACT" switch to "DEACT".
- Turn the thumbwheel to 7.
- Hit the `RESET` button.

- Turn the thumbwheel to 8.
- Hit the `RESET` button.
- Wait for "D2" to be displayed.
- Turn the thumbwheel to 9.
- Hit the `RESET` button.
- Wait for "D3" to be displayed.
- Turn the thumbwheel to the desired number.
- Hit the `RESET` button.
- Repeat steps 8 and 9 if necessary.
- Flip the "ACT/DEACT" switch back to "ACT".

The desired number on the thumbwheel in step 10 depends on what device you want to boot from:

```
A --> Side 0 tape drive
B --> Side 1 tape drive
C --> Side 0 disk drive
D --> Side 1 disk drive
```

In the case of most (99%) of the split switches, step 10 involves turning the thumbwheel to `E` and hitting `RESET` if you're trying to boot side 1 of the switch. This step is not necessary when booting side 0.

If you're using an unsplit switch in the lab, chances are when you look at the front of the switch, one of the sides will be switched to "DEACT" and the other will be switched to "ACT". When booting these switches, step 12 is never necessary. Just go to the "DEACT"ed side, do the thumbwheel–reset routine, and flip both "ACT/DEACT" switches at the same time. (The "ACT" one to "DEACT" and the "DEACT" one to "ACT".)

If all went well, the display on the side you booted will start at E9, go to E8, and then start "spinning". (You'll know what I mean when you see it.) If it doesn't work, try it again. If it still doesn't work, scream for help.

Note to the novice: Never *ever* boot a switch without making sure that everyone using the switch knows you're going to and has given you the go-ahead. And, for heaven's sakes, be *sure* to boot the right switch. Everyone's booted somebody else's switch at least once, and, believe me, it's not a good way to make friends with your co-workers.

## Booting SuperNode

The SuperNode, unlike the NT40, uses a terminal interface to boot the system. The functionality of this terminal is similar to that of the CC thumbwheel and toggle switch of the NT40. This interface is known as a Reset Terminal InterFace (RTIF).

Now, the thing about booting SuperNode is that there are currently no split SuperNodes. That makes booting a little trickier, because the side you boot has to be `ACTIVE`, and the other side has to be `INACTIVE`.

The way you tell the difference is to go to the RTIF (that's the terminal next to the video selector A–B box) and find a CM. Just turn the knob until you see "CM 0" or "CM 1" at the top of the screen. Right beside that, it will say either "active" or "inactive".

If both sides are active, you'll need to type in:

```
>\JAM
>YES
```

to make one side inactive. Then you can boot the other side. If both sides are inactive, you'll need to type in:

```
>\RELEASE JAM
```

to activate one side.

After you've gotten one side active and the other inactive, you're ready to boot. If you're booting from SLM tape, you should now put your tape in one of the slots located at the bottom of the front of the SuperNode cabinet. If you're booting from SLM disk, you can skip this step (obviously).

Now, you're ready to boot. Here's what you do:

```
>\OVERRIDE
>\BOOT SLM1 T      (or SLM0 if it's the one being used)
>YES
```

```
(SLM0 T or SLM1 T) for booting from cartridge tape
(SLM0 or SLM1) for booting from the disk
```

After the switch boots (takes about 15 minutes) you will need to place the splitter cards for SuperNode into their proper slots if the switch is split. The exact procedure for this differs for each lab. If you're doing this for the first time, either ask the person who told you to boot the switch what to do, or call maintenance if you're all alone.

## A Dictionary of the SOS Commands

### Dictionary Format

Each entry looks like the following. I try to give a brief summary of the essentials first.

-----  
COMMAND NAME

All Known Formats : >command <parm>  
                  >different format

Opposite Command : The "inverse" of it

Associated Commands : Other commands

E.G. : >command, as you might enter it  
      >maybe even another example  
-----

Then I go ahead and discuss the command in a more or less random manner. There may be warnings for the unwary, or even examples. And, the stuff may ramble on endlessly. The whole idea is to present as many ideas as possible as to how you actually *use* the command. This

material may duplicate (in intent) the ideas contained elsewhere. And that is good, for there are no other examples other than on the switch execs themselves. Have fun!

---

**ATTACH**

All Known Formats : >ATTACH <directory name>  
>ATTACH <directory> ABOVE <dir-name>

Opposite Command : DETACH

Associated Commands : DETACH, DIRECTORY, LISTST

E.G. : >DIRECTORY USER\_DIR1 -- create the dir  
>ATTACH USER\_DIR1 -- and attach it

---

It is almost certain that without the good old ATTACH command (and its two best pals DETACH and DIRECTORY), little serious work could be done in the world of Regression. The main idea is that you can create your own read-write directories and put them on top of other directories, you can attach to other user directories, etc. It's like the LINK or GU commands used in VM/CMS.

One example is to override the infamous TATS QUIT command. (There is no failsafe, so you can quite easily exit TATS and you have to start all over.)

```
>LOGIN
>BETA BETA          -- login as BETA user name and password

>TATS              -- This will create the two or three
                  -- topmost directories that TATS uses.

>DIRECTORY REGDIR
>ATTACH REGDIR     -- It is now the topmost directory

REGDIR      CE01.2025    RW
TATSUSERDIR CE01.600E    RO
TATSRWDIR   CE01.800D    RO
TATSDIR     CE01.8006    RO
BETA        CE01.C047    RW
PROGDIR     CE01.C004    RO
SYSDIR      CE01.2002    RO
ROOTDIR     CE01.4001    RO
STDIRS      CE01.A01E    RO

>COMMAND QUIT (PRINT '*** To quit, enter: TATS_QUIT')
>COMMAND TATS_QUIT (ERASE QUIT;QUIT ALL;CRT_QUIT)
```

The idea is that our new command will be encountered before the version in the TATSDIR directory. Thus, we can create and attach our own directories. We can also attach other user directories:

```
>LOGIN -- login as OPERATOR
>OPERATOR OPERATOR
>'hi there' -> SECRET_MSG

>LOGIN -- at another terminal, login as BETA
>BETA BETA
```

```

>ATTACH USERS.OPERATOR -- attach the operator's directory
>LISTST

OPERATOR    CE01.2025    RO
BETA        CE01.C047    RW
PROGDIR     CE01.C004    RO
SYSDIR      CE01.2002    RO
ROOTDIR     CE01.4001    RO
STDIRS      CE01.A01E    RO

>PRINT SECRET_MSG
hi there
>

```

---

**CIPROMPT**

```

All Known Formats : >CIPROMPT SETPROMPT ON
                   >CIPROMPT SETPROMPT OFF
                   >CIPROMPT SETABORT <string>
                   >CIPROMPT SETQUERY <string>

```

```

Opposite Command : No opposite command

```

```

Associated Commands : None

```

```

E.G. : >CIPROMPT SETPROMPT OFF
       >CIPROMPT SETABORT QUIT
       >CIPROMPT SETQUERY EXPLAIN

```

---

There are three separate and distinct uses for this command. I'll save the best for last.

First, there's the `SETABORT` use. You know when you put the wrong thing in and CI prompts you for a parameter to the command you typed in and you type "ABORT" to make it stop asking you? Well, with `SETABORT`, you can change "ABORT" to something else. For instance, you could say:

```
>CIPROMPT SETABORT NEVERMIND
```

Then, when you were prompted for a parameter, you could type "NEVERMIND" and CI would leave you alone.

Second, there's the `SETQUERY` flavor. You know when you want to know the format of a command, like `LIST`, and you type "Q LIST"? Well, with `SETQUERY`, you can change "Q" to something else. For instance, you could say:

```
>CIPROMPT SETQUERY TELLMEABOUT
```

Then, when you want to know about the `LIST` command, you type "TELLMEABOUT LIST".

Finally, and most importantly, there's the `SETPROMPT` flavor. This is most useful in SOS execs. Ever written a SOS exec and used a command wrong inside a loop? Me, too. You look at

your terminal and you see is "invalid bla bla bla" scrolling up the screen. Well, with the CIPROMPT SETPROMPT command, you can stop that mess.

CIPROMPT SETPROMPT OFF sets things up so that CI only asks you to correct your input once. So, if you're writing an exec that contains a command that just might blow up, SETPROMPT to OFF and just put a blank line after the command in your exec. That way, if the command blows up, the blank line shuts CI up and your exec can continue on the next line.

---

#### CLEARST

All Known Formats : >CLEARST

Opposite Command : No opposite command

Associated Commands : ERASE (erases one symbol)  
ATTACH, DETACH

E.G. : >CLEARST

---

This handy little command clears your symbol table. Your symbol table is the table in your own private CI process that keeps track of variables you've defined, commands you've defined, and files you know about through LISTSF or LISTVOL.

CLEARST detaches all directories except STDIR, ROOTDIR, SYSDIR, PROGDIR, and the user directory.

---

#### COMMAND

All Known Formats : >COMMAND <user-command> ()  
>COMMAND <user-command> (text...)  
>COMMAND <user-command> <file name>

Opposite Command : (very complicated, so watch closely)

Associated Commands : ERASE (erases the command)

E.G. : >COMMAND REM () -- dummy command like %  
>COMMAND FF (PRINT (HEXTOSTR 12)) -- gen a form-feed  
>COMMAND ZOT XZOT\_EXEC\_FILE -- references an exec

---

COMMAND is the most powerful of the SOS commands. In a sense, all of this document is about the COMMAND command. Here are a few examples:

>COMMAND REM ()

Then, in a file, you can have lots of comments without ever generating a single prompt message (log message):

```
>REM % any comments (Just use the "%" as the first
>REM % character, and leave a blank after the REM.)
```

The SELECT (CASE) command:

```
>COMMAND SEL (@(@1+2))
```

(one of the great mysteries of nature!)

```
>COMMAND KVALS (IF ((RINDEX) < (2)) THEN (
PRINT '*** ERR - Missing parms: <filter-time> <retries>'
PRINT '*** Format is: >KVALS ### ##')
)ELSE(
@1->$KV_FTIME;@2->$KV_RETRIES
PRINT '--- K-Values are now: ' $KV_FTIME $KV_RETRIES) )
```

Here the command has a built-in help facility!

In the following example, we want to create a high-speed tokenized exec:

```
>EDIT XTI_FUNC
>INPUT
  IF ((RINDEX) < (2)) THEN (
    PRINT '*** TI-FUNC, missing parms:  ADD/DEL tuple'
    READ PREVIOUS)
%
  (SYMTOSTR @1) -> $OPT
%
  IF ($OPT='ADD') THEN (
    TYPEIN 'ADD TUPLE '+ (SYMTOSTR @2)
    READ PREVIOUS)
%
  IF ($OPT='DEL') THEN (
    TYPEIN 'DELETE TUPLE '+ (SYMTOSTR @2)
    READ PREVIOUS)
%
  PRINT '*** ERR -- Bad option: ' $OPT
  PRINT '*** Must either be ADD or DEL'
>>
>FILE D010CNTFL  -- put the file out on disk
>COMMAND TIFUNC XTI_FUNC
```

After we enter the `COMMAND` command, if there are any errors, SOS may not in fact tell us!! If we have an extra opening parenthesis, we may get the "TOKENIZING AREA FULL" and the CI system will ignore all the rest of the stuff in the file, until we either get to EOF or get an extra ")".

In order to display the character string that makes up a command you need to be very desperate, because it is more complicated than on the surface it seems it should be. But according to all sources that I know there is no other way, yet, to do this. Anyway, first you need to have a command that you want to display, lets call it `NEATO`. Then you need to invoke `LISTST` to determine the name of your Read Write (R/W) directory. Assuming that you logged onto the switch using `ADMIN` as your userid, `LISTST` might generate the following:

```

>COMMAND NEATO (PRINT 'This is a simple command')
>LISTST
TABDIR          2B01, 60B9    RO
ADMIN           2B01, 20BB    RW
PROGDIR        2B01, A0C5    RO
SYSDIR         2B01, 60C7    RO
ROOTDIR        2B01, 40C8    RO
STDIRS         2B01, 40BA    RO

```

Notice that the `ADMIN` directory has RW to the left of it. Next you should print the R/W directory. On the SuperNode, it might look like this:

```

>PRINT ADMIN
INPUT           Device      Copy    1802
OUTPUT         Device      Copy    1802
NEATO          List        Orig    005890D8

```

and on the NT40 it might look like this:

```

>PRINT ADMIN
INPUT           Device      Copy    1804
OUTPUT         Device      Copy    1804
NEATO          List        Orig    900618

```

Notice that the number on the far right of the `NEATO` command is an address, either an NT40 three byte address, or an S/DMS four byte address. This address is actually a pointer to a structure in CI called `TOKENS`. The first element of the structure is a descriptor of token information, and the second element is a descriptor of the characters making up the command itself. So if you haven't guessed what your about to do, then I will go ahead and tell you: Go into debug and display the silly thing. On the NT40, given that a descriptor is three words long, the follow debug statements will do it for you:

```

>DEBUG
debug mode
>DI 900618.3 (&0) N=80 CHAR
8F7FE3:  ..COMMAND..NEATO..PRINT..This is a simple co
8F7FF9:  mmand.....

```

Notice that this is only the characters making up the command, not tokens. This means that quotes and parentheses will not show up, but @ signs and other characters will show up. On the S/DMS it will look like this:

```

>DEBUG
debug mode
>DI 005890D8.8@ N=50 CHAR
0058B9F0:  ..COMMAND..NEATO..PRINT..This is a simple co
0058BA1C:  mmand.

```

On the S/DMS, the descriptor is eight bytes long, so offset the address by eight, then dereference it, and display the result in CHARACTERS. The N=80 or N=50 is completely arbitrary, and depends on the length of the command. It is possible that you will miss it the first time and need to increment some more.

---

#### COPY

All Known Formats : >COPY <filename> <device>  
>COPY <filename> <new-fname> <device>

Opposite Command : No opposite command

Associated Commands : COPYFILE (I think has disappeared)  
LIST/TLIST, LISTSF, LISTVOL

E.G. : >COPY my\_sf\_file D010TEST2 -- copy to disk

---

See the discussion under MOUNT, as that is where I talk about copying files to tape.

Whenever you use one of the LIST commands (LIST <tape>, LISTSF, or LISTVOL <vol-name>), you create a current pointer to each file. This means that if you had three separate files, each with the same name, and each one on a different device (e.g., SFDEV, tape, and D010TEST2), only the last one you reference is actually "known". So when you do a COPY, it always refers to the current file and that's that. Well, almost.

Let's say we have a file in SFDEV called PIZZA, and we can do the following:

```
>LISTSF
PIZZA
>COPY PIZZA D010TEST2      -- there is now a copy of our file on disk

>COPY PIZZA NEW_PIZZA D010TEST2 -- and now another
                                -- copy out there

>MOUNT 0 FORMAT 'MYTAPE'
ok
>COPY PIZZA T0             -- Can't change the name when
>DEMOUNT T0               -- we copy out to tape.

>COPY PIZZA OLD_PIZZA D010TEST2
*** Error -- device off-line
```

Oops, we forgot that whenever we access a tape, that becomes our new source from the file. To recover, we must do a LISTSF and then copy.

By the way, you *can* create duplicate files on the disk volumes, which are terribly inconvenient.

---

#### DATE

All Known Formats : >DATE

Opposite Command : No opposite command

Associated Commands : SETDATE, TIME, SETTIME

E.G. : >DATE  
Date is TUE. 19/AUG/1986 18:02:40

---

Actually, DATE gives you the time as well, which suggests the following little exec:

```
>EDIT XGET_DATE
>INPUT
SEND SINK
  ERASESF XXTEMP_FILE -- Erase any old copy of the temp
  ABORT -- file we will use for output.
%
SEND SFDEV XXTEMP_FILE -- re-direct the output to file
  DATE -- issue the date command
SEND PREVIOUS -- back to sink (will close file)
EDIT XXTEMP_FILE -- edit the file and extract the
  DOWN 1 -- string with the date info
  LINESTR -> $DATESTR
QUIT
(SUBSTR $DATESTR 15 12) -> $DATE -- use the substring
(SUBSTR $DATESTR 28 8) -> $TIME -- function to get them
%
ERASESF XXTEMP_FILE
SEND PREVIOUS -- continue output to disk, exit and..
>>
>FILE D010CNTFL -- put the file out on disk

>READ XGET_DATE
>PRINT $TIME $DATE
18:02:30 19/AUG/1986
```

Neat-o, eh?

---

DEBUG

All Known Formats : >DEBUG

Opposite Command : No opposite command

Associated Commands : Beats me

E.G. : >DEBUG

---

DEBUG is an obnoxious little module that has no help and refuses to tell you what commands it recognizes. Even worse, it is not an increment, so once you're in DEBUG you're stuck with only DEBUG's commands (whatever they are) until you QUIT out of it.

Actually, I've never used DEBUG, so I don't have much to say about it. There is a document in DIS about it (DEBUG LISTING). Basically, it is a program to debug Protel modules on the switch. It lets you set breakpoints, trace through your program, display variables... all that nifty poop. If you think you might need or want it, print out the manual. Then you'll know more than me.

---

**DECSTRTONUM**

**All Known Formats :** >DECSTRTONUM <string>

**Opposite Command :** NUMTODECSTR

**Associated Commands :** HEXTOSTR, NUMTODECSTR, NUMTOHEXSTR

**E.G. :** >PRINT (DECSTRTONUM '123')  
123

---

DECSTRTONUM takes a string containing an integer and converts it to an integer. Thus, in the example above, the string '123' was converted to the integer 123.

NUMTODECSTR goes the other way. That is, it takes an integer and converts it to a string. Like converting the integer 123 to the string '123'.

---

**DEMOUNT**

**All Known Formats :** >DEMOUNT T##

**Opposite Command :** MOUNT

**Associated Commands :** LIST, TLIST, MOUNT, LOGUTIL

**E.G. :** >DEMOUNT T0  
>DEMOUNT T10

---

Used to demount a tape when you are done with it. For more info, see the discussion under MOUNT.

By the way, the DEMOUNT must be performed from the exact same user terminal where the MOUNT command was issued.

---

**DETACH**

**All Known Formats :** >DETACH <directory-name>

**Opposite Command :** No opposite command

**Associated Commands :** ATTACH, DIRECTORY, LISTST

**E.G. :** >DETACH MY\_DIR

---

See also the discussions under the associated commands.

The DETACH command disconnects the indicated directory from your current symbol table. Note that the directory and all its contents still exists. It is just no longer visible when you do a LISTST. Also note that the entry still exists in the R/W directory wherein you created the

directory; i.e., you still have a pointer to it, even though it is not attached. You can then later re-attach it. However, if you do a `DETACH` and then an `ERASE` on the directory name... guess what? Yep, all of the contents of the directory are *destroyed* forever. You can then re-attach it -- but it will of course be empty.

```
>DIRECTORY MY_DIR
>ATTACH MY_DIR
>COMMAND DOOZY (PRINT '** doozy **') -- created in MY_DIR
>DETACH MY_DIR
```

The command `DOOZY` is temporarily unknown. However, if we then re-attach the directory, all of its contents can again be accessed.

---

#### DIRECTORY

```
All Known Formats : >DIRECTORY <dir-name>

Opposite Command : No opposite command

Associated Commands : ATTACH, DETACH, LISTST, PRINT

E.G. : >DIRECTORY MY_DIR
      >ATTACH MY_DIR
```

---

```
>DIRECTORY MY_DIR
>ATTACH MY_DIR
>3 -> XYZ
>'GOFORIT' -> ABC
>PRINT MY_DIR
XYZ          Int          Copy 3
ABC          String       Copy 'GOFORIT'
```

Thus the entries are created in the topmost read/write directory, and you can always access them with the `(USERS.dir.dir.variable)` format.

---

#### DISKADM

```
All Known Formats : >DISKADM

Opposite Command : No opposite command

Associated Commands : FORMATDISK, DISPLAYDISK, CREATEVOL,
                    DELETEVOL, REINITVOL, DISPLAYVOLS,
See Also : DISKUT

E.G. : >DISKADM
```

---

`DISKADM` is an increment which is mainly used to format and configure the SLM disk drives in a

SuperNode switch. This is an extremely dangerous increment to be in if you're inexperienced.

Since this is so touchy, I'm not even going to explain it any further here. If you really need to know, look in NTP 297–1001–509. That's the only place I've seen it explained.

---

**DISKUT**

**All Known Formats :** >DISKUT -- special disk utility

**Opposite Command :** No opposite command

**Associated Commands :** BACKUP, RESTORE, CLEARBOOTFL,  
SETBOOTFL, LISTBOOTFL, CLEARVOL,  
LISTVOLS, DELETEFL, LISTFL, RENAMEFL,  
INSERTTAPE, EJECTTAPE

**See Also :** DISKADM

**E.G. :** >DISKUT;LISTVOLS S01D;QUIT  
>DISKUT;LISTFL S01DIMAGE;QUIT

---

DISKUT is similar to DSKUT, but it works only on SLM devices.

Like DSKUT, DISKUT is an increment. You enter the increment like this:

```
>DISKUT
```

and get out of it like this:

```
>QUIT
```

I don't want to spend a lot of time in this top–level section. The real fun is in the subcommands. So let's get to it. You'll see something new in there: abbreviations. This is the only place I know of on the switch where the commands have abbreviations that the switch recognizes. They're listed with each command.

Before you go in there, though, I want to give you a little glossary so you'll know what the parameters mean:

- <dev–vol> means the fully qualified volume name, like S01DIMAGE.
- <device> means either S00D or S01D.
- <volume> means the up to eight character volume name which usually follows a device name, like IMAGE.
- <boot table> is either CM (for Computing Module) or MS (for Message Switch).
- <filename> is the name of a file.

Oh, yeah... there's one more thing that makes this increment unique among all the commands on the DMS–100. It actually has good help. In fact, the help on the switch is actually better than any other documentation I've found for the increment. Make use of it. In fact, you might even want to print it out if you're going to be using DISKUT a lot. Use `HELP <command> FULL`. It gives you even more stuff.

Well, that's about it. Let the fun begin!!

---

## BACKUP

Abbreviation : BA

Format : BACKUP FILE <dev-vol> <filename> (<tape filename>)  
or : BACKUP VOLUME <dev-vol> <backup volume name>

---

This command allows you to back things up onto those cute little tape cartridges. You can back up individual files or entire volumes. The only trick is that both the disk volume and the tape have to be in the same SLM. So you can't back up volume `S01DIMAGE` onto tape `S00T`.

Also, notice that the tape filename in the first version of the command is optional. If you don't specify, the filename on the tape will be the same as the filename on the disk.

One final note: when you back up an entire volume, `BACKUP` puts all the files together into a single file. When you use restore, everything is unpacked back into individual files.

For example:

```
>BACKUP FILE S00DVOL1 MYFILE % copies myfile onto tape S00T
>BACKUP VOLUME S00DVOL1 MYVOL % copies entire volume onto tape
```

In the SuperNode environment, backing things up is very important. People tend to erase things from the SLM disks pretty often, so once you get an image from loadbuild configured properly, be sure and back it up onto tape if you intend to use it again.

---

## RESTORE

Abbreviation : RE

Format : RESTORE FILE <dev-vol> <filename>  
or : RESTORE VOLUME <dev-vol> <backup volume name>

---

This command is the opposite of the `BACKUP` command. It allows you to pull files from tape onto the SLM. Again, the disk and the tape have to be on the same SLM.

Examples:

```
>RESTORE FILE S00DVOL1 MYFILE % copies myfile from tape S00T
% onto S00DVOL1

>RESTORE VOLUME S00DVOL1 MYVOL % copies entire volume MYVOL
% from tape onto the disk volume
% S00DVOL1
```

The only catch is that when you restore volumes, the destination volume has to be empty. Otherwise, RESTORE will refuse to restore.

---

CLEARBOOTFL

Abbreviation : CBF

Format : CLEARBOOTFL <device> <boot table> ALL  
ACTIVE  
FILE <volume> <filename>

---

Look familiar? Look a lot like CLEARBOOT? Well, there are some very important differences.

The fundamental thing to remember is that, since SuperNode is actually two computers, there are two active loads: the CM load and the MS load. CLEARBOOTFL clears these loads separately. This, of course, aids you greatly in screwing yourself on those late nights when it's crucial that you don't screw yourself.

Here are some examples:

```
>CLEARBOOTFL S00D CM ALL      % clears all CM loads on S00D
>CLEARBOOTFL S00D MS ACTIVE  % clears the active MS load
                             % on S00D
>CBF S01D CM FILE IMAGE SMC_29_RTM_CM % clears only the
                                     % SMC_29_RTM_CM CM
                                     % load on S01DIMAGE
```

See? No problem. Remember, though, that CLEARBOOTFL does *not* erase the file. It only removes the entry from the ITOC for that device. You have to use the DELETEFL command to erase it.

---

SETBOOTFL

Abbreviation : SBF

Format : SETBOOTFL <dev-vol> <filename> <boot table>  
<entry number> ACTIVE  
REGULAR <-- default

Where <entry number> is the entry in the ITOC where the file name and volume name is to be placed (1 to 15).

Where ACTIVE specifies that the file is to be the active boot file.

Where REGULAR adds the file to the ITOC without making it the active boot file.

---

Not much to explain on this one. It just puts an entry in the ITOC for the file you specify. If you specify the `ACTIVE` option, the file you enter becomes the active boot file. `REGULAR` is the default, though.

Examples:

```
>SBF S01DIMAGE SMC_29_RTM_CM CM 1 ACTIVE % adds SMC_29_RTM_CM
                                         % to the CM boot table
                                         % for S01D and makes it
                                         % the active boot file

>SBF S00DIMAGE MS_TEST_LOAD MS 5 % adds MS_TEST_LOAD to the MS
                                  % boot table for S00D, but
                                  % leaves the active boot file
                                  % alone
```

-----  
LISTBOOTFL

Abbreviation : LBF

Format : LISTBOOTFL <device> <boot table> ALL  
ACTIVE <-- default

Where `ACTIVE` specifies that only the `ACTIVE` boot file for the device is to be listed.

Where `ALL` specifies that the entire contents of the ITOC be listed.

-----

As the name implies, `LISTBOOTFL` lists the contents of the ITOC for the given device and boot table. It's sort of analagous to the `SHOWBOOT` command in `DSKUT`.

Examples:

```
>LISTBOOTFL S01D CM ALL
Image Table Of Contents for CM on SLM, unit 1:
```

ITOC Entry Number	Volume Name	File Name
1	IMAGE	MCI_29BN_CT_CM
2	IMAGE	MSMC29BP_E250E_CM

```
>LISTBOOTFL S01D CM ACTIVE
Image Table Of Contents for CM on SLM, unit 1:
```

```
Active Boot File Device: DISK
Active Boot File Volume: IMAGE
Active Boot File File:   MSMC29BP_E250E_CM
```

---

**CLEARVOL**

Abbreviation : CVOL

Format : CLEARVOL <dev-vol>

---

Yes, this command is just as dangerous as it sounds. It deletes every file from the volume you specify. There are two failsafes, though. It will ask for confirmation before actually doing it, and it will refuse to clear the volume if any of the files are registered in the ITOC.

Example:

```
>CVOL S01DIMAGE
*** WARNING ***
Clearing a volume will destroy all files stored on the volume.
Do you want to clear volume S01DIMAGE (Yes/No)?
>YES
Volume S01DIMAGE has been cleared.
```

Whew!! Gives me the heebie-jeebies just thinking about it.

---

Abbreviation : LV

Format : LISTVOLS <device> {FULL}

---

This will give you extensive information about all the volumes on the SLM device you specify. If all you want to see is the names of the volumes available, `PRINT ROOTDIR` will give you that. `LISTVOLS` tells you all sorts of stuff about each volume. It'll tell you even more stuff if you use the `FULL` option. If you want the details, look in NTP 297-1001-509 for the complete poop.

Example:

```
>LISTVOLS S01D
Volume information for SLM disk 1

Volume   Volume   Modify   Total   No. of   ITOC
Name     Status   Date     No. of   Open     Files
         Y/M/D   Files    Files
-----
IMAGE    OPENED   89/08/03   4        0        4
```

---

**DELETEFL**

Abbreviation : DDF

Format : DELETEFL <dev-vol> <filename>

-----  
Pretty self-explanatory, really. There are a couple of exceptions, though. If the file you're trying to delete appears in an ITOC, you have to CLEARBOOTFL it out before deleting it. The other exception is that you can't DELETEFL a file that's been marked as read-only or non-erasable.

Sorry, but I don't know anything about marking files as read-only or non-erasable. This is the first time I've seen it mentioned.

Example:

```
>DELETEFL S01DIMAGE MCI_29BP_CM
```

-----  
LISTFL

Abbreviation : LF

Format : LISTFL <dev-vol> {FULL | BVOLS}

-----

This command allows you to list the files on a specified volume. In this case, volumes also include tapes. If you want to list all files on a tape, you use S01T or S00T as the <dev-vol>. Actually, you shouldn't do a LISTFL on a tape unless you have a lot of time to kill. It's *extremely* slow.

If you do have a lot of time to kill, and the tape has entire volumes backed up on it, you might want to use the BVOLS option. That will list all the individual files backed up from each volume.

Also, if regular LISTFL doesn't give you all the stuff you wanted to know, you can always use the FULL option to get more info.

Anyway, just like LISTVOL or LIST, LISTFL takes the filenames it finds and puts them in your symbol table, so you can READ them or COPY them or whatever.

Examples:

```
>LISTFL S01DIMAGE
```

File information for volume S01DIMAGE.

File Name	File Org.	File Code	In ITOC
-----	----	----	----
MSMC29BP_E250E_MS	IMG	0	YES
MSMC29BP_E250E_CM	IMG	0	YES
MCI_29BN_CT_MS	IMG	0	YES
MCI_29BN_CT_CM	IMG	0	YES
MYFILE	OTH	0	YES

---

**RENAMEFL**

Abbreviation : RNF

Format : RENAMEFL <dev-vol> <old filename> <new filename>

---

Now here's a little command I wish `DSKUT` had. It does exactly what it sounds like it does... changes the name of an existing file. Of course, if the file is registered as a boot file in an ITOC, you can't rename it.

Example:

```
>RENAMEFL S01DWORK MYFILE YOURFILE % renames MYFILE to YOURFILE
```

---

**INSERTTAPE**

Abbreviation : IT

Format : INSERTTAPE <device> WRITELABEL <label name> RET <-- default  
CHECKLABEL <label name> NORET  
READLABEL <-- default

Where <label name> is up to six characters long.

---

This is the equivalent of doing a `MOUNT` for a tape cartridge. Each tape cartridge has a label name up to six characters.

`READLABEL` is the default. It looks for the label name and tells you what it finds. If there is no label, this is a good indication that the tape has not been written to.

If you choose the `WRITELABEL` option, all existing files on the tape are destroyed.

`CHECKLABEL` could come in handy if you forget to label the outside of the tape. If the label on the tape doesn't match the one you said to check for, the `INSERTTAPE` command is aborted. That way, you don't have to do an `EJECTTAPE` afterwards.

One more thing. When you do an `INSERTTAPE`, the tape will automatically be retensioned for you. If you don't want this to be done, you can use the `NORET` option. I'll probably get in trouble for this, but I highly recommend the `NORET` option. The tape is retensioned every time you do an `EJECTTAPE` anyway, and I think doing it on an `INSERTTAPE` is overkill. Just my opinion.

---

**EJECTTAPE**

Abbreviation : ET

Format : EJECTTAPE <device> WAIT <-- default NOWAIT

---

This command is analogous to a `DEMOUNT` for a cartridge tape. It's rather deceptively named, though, because the tape doesn't actually pop out of the slot when it's done. You have to pull it out by hand.

Do yourself a favor and use the `NOWAIT` option. Otherwise, you'll end up sitting there for about five minutes waiting for the tape to retension. If you use `NOWAIT`, it'll still retension, but you don't have to wait for it.

Example:

```
>EJECTTAPE S01T NOWAIT    % demounts the side 1 tape cartridge.
```

---

#### DSKALLOC

All Known Formats : `>DSKALLOC <unit number>`

Opposite Command : No opposite command

Associated Commands : `DISPLAY, ADD, REINIT, DELETE, DIRADD,`  
`DIRDEL, UPDATE, QUIT`

See Also : `DSKUT`

E.G. : `>DSKALLOC 1`

---

`DSKALLOC` is CI increment which is used mainly to partition and format the hard disk on the switch. As such, it is *very* dangerous in the hands of the inexperienced.

In order to use `DSKALLOC` on a drive, the drive must be manual-busy. To accomplish this, use `MAPCI` as follows:

```
>MAPCI;MTC;IOD;IOC 0;CARD 1;BSY
```

This, of course, depends on the actual `IOC` and `CARD` number where your particular disk drive resides.

The associated commands listed above are the commands available in the `DSKALLOC` increment. I'm not going to discuss `DSKALLOC` any further here. If you really think you're ever going to use it, print out NTP 297-1001-526.

---

#### DSKUT

All Known Formats : `>DSKUT -- special disk utility`

Opposite Command : No opposite command

Associated Commands : `LISTVOL, SHOWVOL, SHOWFL, RENAMEFL,`

```
ERASEFL, SETBOOT, SHOWBOOT,  
CLEARBOOT, QUIT  
And : ROOTDIR  
Also See : DSKALLOC  
  
E.G. : >DSKUT;LISTVOL D010CNTFL ALL;QUIT  
>DSKUT;SHOWVOL D010CNTFL
```

---

```
>PRINT ROOTDIR -- One of the fastest ways to see what all  
-- volumes have been allocated to your disk drive.
```

DSKUT is a CI increment (sort of like LOGUTIL) that gives you some ability to see what all files are on your disk volumes. It also allows you to create and destroy not only your own files, but files from other people. And you can also create disk files with duplicate names, etc.

Naturally, there were plans to revise and update the system. However, there just hasn't been time. So, we must learn to live with it. Now, this section is rather lengthy (especially since I kept referring you here from various other commands), so you might want to take a break before digging in.

```
>DSKUT -- this puts you in the disk utility sub-system  
>QUIT -- gets you out
```

Actually, it is difficult to find out what all is allocated with this sub-system, so we use two tricks:

```
>PRINT ROOTDIR
```

This, of course, prints out all of the devices and should give you a list of the disk volumes which have been allocated.

The second trick to getting the volumes is to do this:

```
>MAPCI NODISP;MTC;IOD;IOC 0;CARD 1;ALLOC  
or  
>MAPCI ;MTC;IOD;IOC 0;CARD 1;ALLOC
```

Not only does this give you the list of the volumes, but whether or not any user has any files open on them. In fact, this is the routine you have to use to re-allocate the volumes (see the discussion under DSKALLOC).

Next on our grand tour are the more "normal" things that we all want to do with a disk (or do to it as the case may be):

```
>DSKUT;LISTVOL D010CTS ALL  
UNI14_5LOOPS  
SBS15_5LOOPS  
STD_PROF
```

```
>QUIT
```

Now, as we know from the `LISTST` and `LISTSF` commands, the `LISTVOL` command makes the various files known to our symbol table. That is, there may well be a file called "SIDE\_0" over on the disk under the volume `D010MAINT`, but unless we do a `LISTVOL D010MAINT ALL`, we will never know it. So, `LISTVOL` makes a copy of the file's location in our directory. This means that we can access it as if it were in `SFDEV`.

```
>READ STD_PROF -- execute the file directly from disk.
```

```
>COPY STD_PROF SFDEV -- copy the file to SFDEV (temp storage)
```

The main difference between `DISK` files and `SFDEV` is this:

`SFDEV` files: Remain on the user memory as long as we do not un-permit the user or reboot.

`DISK` files: Should be fairly robust, but you should always back them up on tape if you want to keep them. Better yet, move copies up to the IBM. Better yet even, do both.

Once we are in `DSKUT`, we can do a help and use the various commands:

```
>DSKUT
DSKUT:
>HELP DSKUT
```

From here on out, I'll discuss the various commands available in `DSKUT` (listed in the header as associated commands).

```
-----
LISTVOL
```

```
Format : >LISTVOL <device> ALL | <username>
-----
```

`LISTVOL` lists all files on the given volume which belong to the given user. If no username is given, the default is yourself. `LISTVOL` also puts pointers in your directory to the files it lists, so that you can do things with those files.

Say, for instance, that you sat down at `OPERATOR` one day, edited a file, and filed it on `D010CNTFL`. The next day, you went back to `OPERATOR` and tried to `PRINT` the file:

```
>DSKUT
DSKUT:
>PRINT MYFILE
File not found
>LISTVOL D010CNTFL -- Lists all files on D010CNTFL
MYFILE -- that OPERATOR created.
>PRINT MYFILE -- it's found this time
....
```

You would have to do the LISTVOL before MYFILE could be found and printed.

Now, let's have some fun.

```
>DSKUT
>LISTVOL D010CNTFL ALL -- all files on D010CNTFL
>LISTVOL D010CNTFL BETA -- all files on D010CNTFL that
                        -- BETA created
>LISTVOL D010CNTFL      -- all files on D010CNTFL that you created
```

---

#### SHOWVOL

Format : >SHOWVOL <volume> ALL

---

SHOWVOL shows you some information on the disk. ALL is an optional parameter which will give you additional information that you probably don't need. SHOWVOL without ALL will show you the volume size (in blocks), the number of files on the volume, and the amount of free space remaining on the volume (also in blocks). For example:

```
>SHOWVOL D010IMAGE      -- basic information
  or
>SHOWVOL D010IMAGE ALL  -- more detailed information
```

---

#### SHOWFL

Format : >SHOWFL <filename> ALL

---

SHOWFL gives information about the specified file. ALL is an optional parameter to give you some more information that you probably don't need.

Entering SHOWFL <filename> will tell you the number of records in the file, the date and time it was last modified, and format (fixed or variable length) and length of the records. For example:

```
>SHOWFL MYFILE          -- for information on MYFILE
  or
>SHOWFL MYFILE ALL     -- for more information
```

---

#### RENAMEFL

Format : >RENAMEFL <volume> <old file name> <new file name>

---

*Note:* When you do `HELP DISKUT`, this command will be listed. However, it applies to the SLM and, thus, is only available on SuperNode. Personally, I have never had occasion to use this command, so if you need more information on it look at the NTP document 297-1001-509.

---

**ERASEFL**

**Format :** `>ERASEFL <file name>`

---

Erases a file. (Self-explanatory, really).

Actually, there is a quirk. If the same file name occurs on more than one volume, the volume you `LISTVOL'd` last is the one the file is deleted from.

---

**SETBOOT**

**Format :** `>SETBOOT <boot file name>`

---

On any NT40 switch, there is an image volume which contains one or more bootable software loads. `SETBOOT` allows the user to specify which image is to be loaded when the switch is booted from that volume.

For example, let's say you had a volume `D000IMAGE` on a switch which contained two loads: one for Sprint (`USS29BC`) and one for MCI (`MCI29BC`). If you wanted to boot the MCI load, you would do the following:

```
>DSKUT
DSKUT:
>LISTVOL D000IMAGE ALL
MCI29BC
USS29BC
>SETBOOT MCI29BC
```

The next time you boot from the side 0 disk, `MCI27BC` will be loaded.

---

**SHOWBOOT**

**Format :** `>SHOWBOOT <volume>`

---

`SHOWBOOT` tells you which is the active image on the specified disk volume. For example, if you did a `SHOWBOOT` after the `SETBOOT` above, `SHOWBOOT` would tell you that `MCI29BC` is the active image.

---

**CLEARBOOT**

**Format :** >CLEARBOOT <volume>

---

CLEARBOOT clears the current boot file indicator for the volume specified. For example, if you did a CLEARBOOT after the SETBOOT above, D000IMAGE would still contain both boot files, but neither would be current. Consequently, if you tried to boot the switch from the side 0 disk, you would get an error.

---

**DUMP**

**All Known Formats :**

```
>DUMP <filename> <volume> | MATE | UPDATE {SILENT}
                             | ACTIVE | RETAIN {TERSE}
                             | DEBUG | {VERBOSE}
                             | UNSAFE |
```

**Opposite Command :** No opposite command

**Associated Commands :** RESTART, SHOWBOOT, SHOWBOOT

**E.G. :** >DUMP MCI29BJ D010IMAGE ACTIVE UPDATE  
>DUMP USS29BJ D010IMAGE MATE RETAIN  
>DUMP TRES29BJ T0 ACTIVE UPDATE -- dump to tape

---

The dump command is used to "take an image". This means copying the contents of all used memory to disk or tape so that you can reload the switch at a later time and start where you left off. Actually, the above are not all the known formats of the DUMP command. As important as this command is, the only place I've found reference to it is NTP Supplement 297-1001-509.

The filename and device are pretty much self-explanatory.

Of the four options of the next parameter, I am only going to discuss the first two. If you want to know anything about DEBUG or UNSAFE, you'll have to look in NTP 297-1001-509 or find someone else who knows.

MATE and ACTIVE simply refer to which side of the switch you want to dump in an unsplit switch. Since most of the labs are split, you will probably use ACTIVE most of the time. However, if you were on an unsplit switch, you could use MATE to dump the side of the switch that is inactive.

If you choose the UPDATE option, the image you just took will be rebooted whenever anyone boots from the device you just DUMPed to. If you choose the RETAIN option, the image which was active remains active.

One more little hint: when you dump to tape, always make the dump UPDATE. Otherwise, you won't be able to boot with the tape.

## Addendum for SuperNode

SuperNode's `DUMP` has one more option tagged onto the end. The `SCOPE` option. Since SuperNode has more than one CPU, there is a `SCOPE` option so you can dump whichever image you want. The options are rather lengthy, and I don't know what a lot of them do, but if you do a:

```
>Q DUMP
```

on your nearest SuperNode, it'll give you the full story. The options I do understand are:

```
CM          -- dumps the CM load (ACTIVE or MATE from above)
MS {0|1}    -- dumps the MS load
TOTAL      -- default... dumps CM and MS
```

If you use the `TOTAL` option, the dump will use the filename you specified and tag on "`_CM`" for the CM load and "`_MS`" for the MS load, so you'll end up with two files.

When you do a `DUMP` on a SuperNode, it's automatically recorded in the `ITOC`.

```
-----
ELSE      (See the discussion under IF)
```

```
E.G. : IF (X=3) THEN (
        PRINT '--- Ah-ha, X is three.'
        ) ELSE (PRINT '--- Well, X is certainly not 3.')
```

```
-----
ERASE
```

```
All Known Formats : >ERASE <symbol>
                   >ERASE <command>
```

```
Opposite Command : No opposite command
```

```
Associated Commands : ERASESF, ERASEFL (DSKUT), EDIT,
                      COMMAND (used to create new ones), etc
```

```
E.G. : >COMMAND REM ()    -- dummy command like %
       >ERASE REM         -- Zappo... it's gone
```

```
-----
This command is mainly used to erase an existing command. That is so that we can redefine it. Note that a new definition does not override the older one. You have to ERASE a command to redefine it.
```

```
-----
ERASEFL
```

```
All Known Formats : >ERASEFL <filename>
-----
```

See discussion under DSKUT.

---

**ERASESF**

All Known Formats : >ERASESF <filename>

Opposite Command : No opposite command

Associated Commands : LISTSF, ERASE

E.G. : >ERASESF MYFILE -- deletes it from SFDEV

---

ERASESF erases a file from the user's RAM work area SFDEV. In order to erase a file, you must do a LISTSF to get an entry for that file in your user directory.

---

**EXECUTE**

All Known Formats : >EXECUTE <filename>

Opposite Command : No opposite command

Associated Commands : READ

E.G. : >EXECUTE MYFILE

---

This command is thoroughly nifty. It's basically like the read command, except with error checking. It will load your SOS exec up and execute it, but it will abort execution if one of the commands gives a bad return code.

If you want the full information, there is a short DIS document about EXECUTE named "W190" (really descriptive). It'll tell you all you need to know.

---

**FAILMESSAGE**

All Known Formats : >FAILMESSAGE <type>

Where type is one of:  
{SHORT, LONG, ALL, SAVE, PREVIOUS}

Opposite Command : No opposite command

Associated Commands : None

E.G. : >FAILMESSAGE SAVE  
>FAILMESSAGE PREVIOUS

---

FAILMESSAGE determines the amount of information to be given when a command fails. It is usually set to LONG, which gives you the error message and the command which was attempted:

```
>LISTSF ALL BLABLA
Either incorrect parameter or wrong number of parameters
LISTSF -- Wrong number of parameters
```

SHORT gives you the error text, but not the attempted command, and ALL gives you the route to the failed command in addition to all information given with LONG. SAVE saves the current setting (to be reinstated with PREVIOUS) and sets FAILMESSAGE to LONG.

I've just given you all the information in NTP 297-1001-509, so if you want to know more, play with it.

---

#### HEXTOSTR

```
All Known Formats : >HEXTOSTR #nn
                    Where 'nn' is any hexadecimal number

Opposite Command : No opposite command

Associated Commands : NUMTODECSTR, SYMTOSTR, STRTOSYM,
                    DECSTRTONUM

E.G. : >PRINT (HEXTOSTR #61)
      a
```

---

HEXTOSTR is one of many data conversion commands provided to allow the user (that's you) to convert data from one form to another. Unfortunately, there is no facility for converting string data directly to hex.

Always remember to put the "#" in front of the number. If you don't, you get some really strange results.

---

#### HT

```
All Known Formats : >HT

Opposite Command : RT

Associated Commands : RT, HXX, HX

E.G. : ><BREAK>
      ?HT
```

---

HT is used in the same way as it is used in VM/CMS. If you do something that generates a lot of output (like LISTVOL D010CNTFL ALL) and you don't want to see all the output, just do a <BREAK> HT. It's like doing a SEND SINK in the middle of the command. When the command

finishes, you will get a command prompt and everything will be back to normal.

The inverse of HT is RT. If you decide that you want to see how far along the command is after you've done an HT, you can do a <BREAK> RT and the output will resume.

---

#### HX

All Known Formats : >HX

Opposite Command : No opposite command

Associated Commands : HXX, HT, RT

E.G. : ><BREAK>  
?HX  
User process stopped.

---

HX is used in the same way as it is used in VM/CMS. If you get yourself in trouble (like an infinite loop in a SOS exec), you can do a <BREAK> HX to get yourself out of it. It stops your user process and resets it, so that you end up back at a CI prompt.

---

#### HXX

All Known Formats : >HXX

Opposite Command : No opposite command

Associated Commands : HX, HT, RT

E.G. : ><BREAK>  
?HXX  
User process stopped.

---

As far as I can tell, this does the same thing as HX.

---

#### IF

All Known Formats : >IF (expr) THEN (cmds) {ELSE (cmds)}

Opposite Command : No opposite command

Associated Commands : UNTIL, WHEN

E.G. : >IF (A=1) THEN (DATE;LOGOUT) ELSE (PRINT 'Howdy')

---

IF is discussed much more in depth in the "SOS Exec" section. Briefly, IF works much the same as in any programming language. The expression in parentheses is evaluated and, based on the outcome, either the THEN or the ELSE command lists are executed.

The SOS `IF` command supports compound expressions, expressions ANDed and ORed together. Each expression is in the form <comparand> <comparator> <comparand>.

The comparands can be variables (`A`), functions (`SYMTOSTR @1`), or literal values (`1`, `'ON'`).

Logical expressions can be combined with relational operators to form compound expressions.

Now, for some examples:

```
>1->A
>IF (A=1) THEN (PRINT 'then') -- the ELSE is optional
  then

>IF (A>1) THEN (PRINT 'then') ELSE (PRINT 'else')
  else

>COMMAND DOIT (IF ((SYMTOSTR @1)='THEN') -- using a
>                THEN (PRINT 'then'      -- function call
>                ) ELSE (PRINT 'else'))
>DOIT THEN
  then
>DOIT BLABLA
  else

>COMMAND DOIT2 (IF ((RINDEX=0) | (LINDEX>0)) -- complex
>                THEN (PRINT DOIT2_HELPF    -- expression
>                ) ELSE (READ DOIT2_EXEC))   -- using or
```

---

#### LEAVE

All Known Formats : `>LEAVE`

Opposite Command : No opposite command

Associated Commands : `QUIT`

---

`LEAVE` is the same as `QUIT`. See `QUIT`.

---

#### LINDEX

All Known Formats : `>IF (LINDEX=0) THEN ...`

Opposite Command : `RINDEX`

Associated Commands : `RINDEX`

E.G. : `>IF (LINDEX = 1) THEN (PRINT 'Help information')`

---

No, this is not a cheap substitute for the leading window cleaner. `LINDEX` stands for Left Index. It tells you how many parms there are to the left of the command. It always applies to commands. Its

main use seems to be for the help facility;

```
>COMMAND BERLIOZ (IF (LINDEX=1) THEN (PRINT BERLIOZ_HELPF
>                               )ELSE(READ XBERLIOZ_XEC) )
```

That way if the user enters `>HELP BERLIOZ`, they will see the file `BERLIOZ_HELPF`. Otherwise, the file `"XBERLIOZ_XEC"` is run.

Some fancy things can be done with it.

---

#### LIST

All Known Formats : `>LIST <tape drive designator>`

Opposite Command : No opposite command

Associated Commands : MOUNT, DEMOUNT, TLIST

E.G. : `>MOUNT 0` -- mount the tape on drive 0  
`>LIST T0` -- list all the files on the tape

---

`LIST` is an older form of `TLIST`. It is used to read record-by-record the entire tape, so as to give a list of the files on that tape. See also the discussion under the `TLIST` command.

`LIST` is very important since it does not require that the first file on the tape be the special `"TAPE$DIR"` file (as does `TLIST`). Further, `LIST` will allow you to copy the files from the tape directly to disk. Here is the recipe:

See the discussion under `MOUNT` for how to create the tape shown here.

```
>MOUNT 0
>LIST T0
MY_FILE
MY_DATA_INFO
SEND_HELPF
>COPY MY_FILE      D010TEST2  -- copy the files to disk
>COPY MY_DATA_INFO D010TEST2
>COPY SEND_HELPF  D010TEST2
>DEMOUNT T0
```

Now we notice that the first file on the tape is *not* the good old `TAPE$DIR` (which is what is created on the IBM with `XDMSTAPE exec`). When we use the `TLIST`, we can only move the files to `SFDEV` (not to a disk volume like `D010TEST2`, etc).

---

#### LISTSF

All Known Formats : `>LISTSF {ALL | username}`

Opposite Command : No opposite command

**Associated Commands :** ERASESF

**E.G. :** >LISTSF ALL  
>LISTSF OPERATOR

---

LISTSF lists all files on SFDEV owned by the given user (default is you). If ALL is specified, it lists all files on SFDEV owned by any user.

Like LISTVOL, LISTSF makes an entry for each file in your read-write directory. Also like LISTVOL, if there is more than one file listed with the same name, the current entry in your directory will be the last one listed. Et tu, LISTSF?

---

**LISTST**

**All Known Formats :** >LISTST

**Opposite Command :** No opposite command

**Associated Commands :** PRINT, DIRECTORY, ATTACH, DETACH

**E.G. :** >LISTST

---

LISTST shows you all the directories current available to you in the order they will be searched.

The PRINT command will list the contents of any of those directories. The notable exception to this is when you autologin (like when you login at MAP4 with the username MAP4). Then, when you do a LISTST, one of your directories will be MAP4. But when you try to PRINT that directory, the switch will respond "device". In order to get the contents of the directory, you have to fully qualify it:

>PRINT (USERS.MAP4)

---

**LISTVOL**

**All Known Formats :** >LISTVOL <volume> {ALL | username}

**Opposite Command :** No opposite command

**Associated Commands :** DSKUT, SHOWVOL, LISTSF

**E.G. :** >LISTVOL D010CNTFL -- all your files  
>LISTVOL D010TEST2 OPERATOR -- all the operator's files

---

See discussion under DSKUT.

---

## LOAD

All Known Formats : >many and varied

Opposite Command : UNLOAD

Associated Commands : See the DIS document OSA10 LISTING

---

LOAD is a command not for the faint of heart. If you don't know what it is, you probably don't need it. If you know what it is and need to know more about it, look at the DIS document "OSA10 LISTING".

---

## LOGIN

All Known Formats :

At a Terminal : ?LOGIN

Detached : >LOGIN <username> <password> <profile> <device>

Opposite Command : LOGOUT

Associated Commands : PERMIT, UNPERMIT

E.G. : <BREAK> -- hit the break key  
?LOGIN  
Enter user name and password:  
>BETA BETA  
User logged in on MAP4 at 3:30PM Aug-3-89  
CI:  
> -- ready for action

---

Before you can log in, user name and password must be permitted. The usual protocol is to have the password the same as the name; e.g., "BETA" user name and "BETA" password.

At a MAP that is already logged in (say the OPERATOR terminal):

>PERMIT BRAND BRAND 3 8000 ALL (See the permit command)

## Login at a Terminal

Go to an unused terminal and enter:

```
<BREAK> -- hit the break key
?LOGIN
Enter user name and password:
>BRAND BRAND
User logged in on MAP4 at 3:30PM Aug-3-89
>
```

## Detached User Login

The detached users are much like the disconnected users on the IBM. They don't have a terminal. The point of having a detached user is so you can get something done in the background while you do other things at your terminal.

The recipe for logging in a detached user is the following:

At a terminal you've logged in to, enter

```
>PERMIT DETUSER DETUSER 2 8000 ALL
```

At this point, I have created a logon user name and password called "DETUSER". There is nothing magic about the name. It can be "ZAPPA" or "BOZO" or anything you want.

Now I'm going to create a profile for the detached user to run.

```
>EDIT TPROF      -- edit a profile
EDIT:
>INPUT
MSG ALL "---- Hi!  I'm a detached user.'
LISTSF ALL
MAPCI;MTC;IOD;IOC 0;CARD 0
RTS
CARD 1
RTS
QUIT ALL
MSG ALL '---- The tape & disk drives are now inservice.'
>>
>FILE SFDEV
```

Now, I will stream the detached user "DETUSER" with an initial logon profile of TPROF (which I must be able to access... but since I just created it, I certainly can access it).

```
>LOGIN DETUSER DETUSER TPROF SFDEV
```

I have never tried using any device other than SFDEV.

Now, after a few moments (depending on how the switch is running), we would get a message on every logged on console:

```
MSG FROM DETUSER  --- hi!  i'm a detached user
MSG FROM DETUSER  --- the tape & disk drives are now inservice
```

Of course, promptly after that the user dies, since we didn't give him much to do. But you can see the potential.

---

## LOGOUT

All Known Formats : >LOGOUT

Opposite Command : LOGIN

Associated Commands : PERMIT, UNPERMIT, FORCEOUT

E.G. : >LOGOUT

User logged out on MAP4 at 3:31PM Aug-3-89

---

The LOGOUT command terminates your session. When you log out, the files that you have saved on disk or in SFDEV are still there. However, the variables, commands, and directories you have created will disappear.

---

## LOGUTIL

All Known Formats : >LOGUTIL

Opposite Command : No opposite command

Associated Commands : SUPPRESS, RESUME, THRESHOLD,  
STARTDEV, STOPDEV

---

LOGUTIL is an increment which enables several commands related to the log system. The commands listed as associated commands are just a few of the commands LOGUTIL recognizes.

If you want to do any really fancy stuff with logs, you need to look in DIS under the heading "LOG SYSTEM" to find the document most suited to your purposes.

For now, here are a few quickies:

>LOGUTIL -- go into the increment

>STARTDEV 1PRT0 -- to start logs to a device

>STOPDEV 1PRT0 -- to stop logs going to a device

To set all the logs on, you go through MAPCI:

>MAPCI;MTC;IOC;CDR;SETLOG ALL

>QUIT ALL

To suppress all logs of a particular type:

>SUPPRESS PM -- dangerous since you might lose important info

```
>SUPPRESS PM 108 -- still dangerous
```

Better to just set them at a threshold (e.g., print only every 20<sup>th</sup> report of this type):

```
>THRESHOLD 20 PM
```

```
14 reports thresholded
```

```
>RESUME PM -- to "un-threshold" reports
```

```
-----  
MOUNT
```

```
All Known Formats : >MOUNT 0          -- to read  
                   >MOUNT 0 FORMAT 'mytape' -- to write
```

```
Opposite Command : DEMOUNT
```

```
Associated Commands : LIST, TLIST, COPY
```

```
E.G. : >MOUNT 0  
       >TLIST T0  
       >READ BUILD$LOAD  
-----
```

MOUNT makes the switch recognize a tape you have mounted on a tape drive. The following is an example of how to copy files from tape to disk.

```
>MOUNT 0  
>LIST T0  
MY_FILE  
MY_DATA_INFO  
SEND_HELPF  
>COPY MY_FILE          D010TEST2 -- copy the files to disk  
>COPY MY_DATA_INFO    D010TEST2  
>COPY SEND_HELPF      D010TEST2  
>DEMOUNT T0
```

Now, let's copy files from disk to tape.

```
>ERASETAPE 0  
>MOUNT 0  
>LIST T0  
>COPY MY_FILE T0      -- copy the files to tape  
>COPY MY_DATA_INFO T0  
>COPY SEND_HELPF T0  
>DEMOUNT T0
```

By the way, the TLIST command is similar to the LIST command. However, TLIST does not allow you to copy files from tape directly to disk. (To SFDEV only.) That's why I usually use the LIST command.

---

**MSG**

All Known Formats : >MSG <user | ALL> '<message>'

Opposite Command : No opposite command

Associated Commands : None

E.G. : >MSG OPERATOR 'Please demount your tape.'  
Message sent to 1 user(s).

>MSG ALL 'Cold restart in 1 minute. Anybody object???'  
Message sent to 12 user(s).  
Message from MAP4 Yes. I am in the middle of a DMOPRO  
>MSG MAP4 'How long do you think it'll be?'  
Message from MAP4 About another 10 mins.  
>MSG MAP4 'OK. Tell me when you're done.'  
Message from MAP4 Ok.

---

Enough said? Well, not quite. If the other user is a detached user, we can use the `MSG` command to tell us that certain tasks have been completed. (See the example under `LOGIN`).

---

**NOPROFILE**

All Known Formats : >NOPROFILE -- must be the first thing  
-- typed after login

Opposite Command : No opposite command

Associated Commands : LOGIN, PROFILE

E.G. : <BREAK>  
?LOGIN  
Enter user name and password:  
>BETA BETA  
CI:  
>NOPROFILE <-- will override a login profile  
>

---

This is used to override a login profile that has been set up for the user name. It must be typed immediately after the login process is completed. Otherwise, you will execute the login profile (if there has been one assigned). Sometimes a good thing, other times dangerous. Depends on what the profile is supposed to do for you. Some profiles save your life. Others are just plain dumb.

---

**NUMTODECSTR**

All Known Formats : >NUMTODECSTR <integer>

Opposite Command : DECSTRTONUM

Associated Commands : HEXTOSTR, NUMTOHEXSTR

E.G. : >PRINT (NUMTODECSTR 25)

---

NUMTODECSTR is one of many data conversion commands provided to allow the user (that's you) to convert data from one form to another. NUMTODECSTR takes an integer and converts it to a string. Thus, in the example above, the integer 25 was converted to the string '25'.

DECSTRTONUM goes the other way. That is, it takes a string and converts it to an integer. Like converting the string '25' to the integer 25.

---

#### NUMTOHEXSTR

All Known Formats : >NUMTOHEXSTR <integer>

Opposite Command : No opposite command

Associated Commands : HEXTOSTR, NUMTODECSTR, DECSTRTONUM

E.G. : >PRINT (NUMTOHEXSTR 123)  
007B

---

Like its brothers (listed above), NUMTOHEXSTR does data conversion. It takes an integer and converts it to a 4-digit hexadecimal number in string form. Confused? What I mean is, in the above example, the integer 123 is converted to the string '007B'. The result is a string, not an integer.

Unfortunately, there is no command opposite to NUMTOHEXSTR.

---

#### PAR

All Known Formats : >PAR <integer>

Opposite Command : No opposite command

Associated Commands : COMMAND

E.G. : >PAR 1

---

The PAR command (more easily recognized as "@") is the facility for passing parameters to user-defined commands. Allow me to illustrate:

```
>COMMAND DOIT (PRINT PAR 1)
>'HOWDY' -> A
>DOIT A
HOWDY
```

The reason for the two different formats ("PAR" versus "@") is illustrated in the PAR functions section.

---

**PARNAME**

All Known Formats : >PARNAME <symbol>

Opposite Command : No opposite command

Associated Commands : PARTYTYPE, SYMTOSTR

E.G. : >PARNAME @1

---

This command is similar to the SYMTOSTR command in that it converts a symbol into a character string. The difference is that SYMTOSTR gives you the string representation of the contents of the variable. PARNAME gives you the string representation of the variable name.

Take, for example, this set of commands and responses:

```
>COMMAND DOIT (PRINT (PARNAME @1))
>DOIT HOWDY
HOWDY
>DOIT D010CNTFL
D010CNTFL
```

To illustrate the difference between PARNAME and SYMTOSTR, look at this:

```
>(3+4)->A
>PRINT (SYMTOSTR A)
7
>PRINT (PARNAME A)
A
```

As you can see, this is a pretty useful command. The strange thing is that it doesn't appear in any of the documents I've found. It's not in NTP 297-1001-509 or NTP Supplement 297-1001-509, anyway. I got it out of an eight year old document in DIS called "OSLD2". Good stuff, though.

---

**PARTYPE**

All Known Formats : >PARTYPE <symbol>

Opposite Command : No opposite command

Associated Commands : PARNAME

E.G. : >PARTYPE VAR1

---

This function returns an integer which indicates the type of the parameter passed in. The types and their corresponding values come from the module CITYYPES and are as follows:

<u>Type</u>	<u>Value Returned</u>
null	0
integer	1
unsigned integer	2
double integer	3
unsigned double integer	4
boolean	5
string	6
file	7
directory	8
program	9
increment	10
cibincom	11
interp command(?)	12
list command	13
non-res command	14
device	15
pointer	16
non-res increment	17
long word	18

This could be really useful for making sure parameters are of the correct type. For example:

```
>COMMAND DOIT (IF ((PARTYPE @1) ^= 1) THEN (
>PRINT 'Sorry. Only integers allowed.'
>) ELSE (
>PRINT (3 + @1)))
```

Important point:

If the parameter you pass to `PARTYPE` doesn't exist (i.e. should return 0), `PARTYPE` will blow up. That means that control structure you're in (`IF`, `WHILE`, etc.) will also blow up. This makes `PARTYPE` rather hazardous to the robustness of your exec. Below is an example of how to get around this problem in a simple `IF` statement:

```
FALSE->CHECK
IF ((PARTYPE SCMSTOSOS) = 13) THEN (TRUE->CHECK;QUIT)
PRINT ' '
IF (^CHECK) THEN (PRINT 'You are not in NCMS.')
```

Notice that there is no `ELSE` to the `IF` in the second line of this exec. Normally, you would expect this segment to be written like this:

```
IF ((PARTYPE SCMSTOSOS) = 0) THEN (PRINT 'You are not in NCMS.')
```

But, if the statement were written this way, `PARTYPE` would blow up the `IF` command before the `THEN` was ever reached. Thus, the problem. Just keep it in mind.

---

**PERMIT**

All Known Formats : >PERMIT <username> <password> {priority} {stksize} {language} {cmdclass}

Opposite Command : UNPERMIT

Associated Commands : LOGIN

E.G. : >PERMIT BETA BETA 4 8000 ENGLISH ALL  
>PERMIT ALPHA ALPHA ENGLISH ALL

---

PERMIT allows you to add to the list of authorized users on the switch. The custom is to make the username and password the same. (That makes it easier to guess your way in.)

Priority values range from 1 to 4 with 4 being the highest. 4 is also the default.

Stacksize values range from 1,500 to 10,000. The default is 5,000.

Language can be either ENGLISH or FRENCH.

Command Class determines which commands are available to the user. I've always used ALL.

Remember that both real users and detached users have to be permitted.

---

**PRINT**

All Known Formats : >PRINT {HEX | DEC} <item>

Where 'item' is one of:  
expression, integer, string, symbol,  
filename, directory, booleans

Opposite Command : No opposite command

Associated Commands : LISTST, LISTSF, LISTVOL, LIST

---

Oh, goody!! Now, this is a really neat command. You can do all sorts of nifty things with it. For one thing, you can PRINT just about anything you can think of, and you can even get the hexadecimal value of it.

The best way I can think of to show you the true niftiness of this command is to do some examples. So here goes!!

```
>DIRECTORY MYDIR
>ATTACH MYDIR
>1->A
>PRINT MYDIR      -- example of printing a directory
A                Int      Copy    1

>PRINT A          -- example of printing a symbol
```

1

```
>PRINT HEX A -- example of printing a hex value
#0001

>'abc'->ABC
>PRINT ABC
abc
>PRINT HEX ABC -- example of printing the hex
#616263 -- value of a string

>LISTST -- you've seen this one before
MYDIR CE01,8071 RW
MAP4 CE01,C047 RW
PROGDIR CE01,C004 RO
SYSDIR CE01,2002 RO
ROOTDIR CE01,4001 RO
STDIRS CE01,A01E RO

>PRINT MAP4 -- Oops... this is a directory, but
device -- PRINT evaluates it first as a device.

>PRINT (USERS.MAP4) -- there... we fully qualified it
B Int Copy 1

>PRINT ((4*4)+(6/3)) -- yes, CI does support mathematical
18 -- expressions, including parentheses
-- and +, -, *, /

>1->A;2->B
>PRINT (A+B) -- it even works with variables
3

>TRUE->BOOLVAR -- boolean variables, too
>PRINT BOOLVAR
TRUE
>FALSE->BOOLVAR
>PRINT BOOLVAR
FALSE

>PRINT HEX (A+B) -- and will even return hex values
#0003

>EDIT MYFILE -- input a file to print
EDIT:
>INPUT
INPUT:
>First line
>Second line
>Third line
>>
EDIT:
>FILE SFDEV
CI:
>LISTSF -- have to do this so PRINT can find MYFILE
MYFILE
>PRINT MYFILE -- PRINT that puppy
First line
Second line
Third line
```

There now... wasn't that fun?

---

## PROFILE

```
All Known Formats :      | LOGIN   | <filename> |
                        >PROFILE | AUTO    |           |
                        | RESTART | CLEAR     |
```

Opposite Command : No opposite command

Associated Commands : LOGIN, RESTART, NOPROFILE

```
E.G. : >PROFILE LOGIN CLEAR  -- clears current login profile
       >PROFILE LOGIN MYFILE -- sets up a new login profile
```

---

This is the command that makes it possible for you to set up a login profile (a SOS exec to be executed when you login). There are two other forms of it that need discussion first, though.

AUTO is used only by the DMS-100 emulator on the IBM. I have never used it, so if you need it, look somewhere else.

RESTART is used on the switch, presumably to make the switch execute the given SOS exec at every restart. I have never used this, either, but it looks pretty straight-forward.

So much for that discussion. Now, on to the stuff I have done... LOGIN profiles.

As the example above indicates, clearing out a login profile is as simple as entering >PROFILE LOGIN CLEAR. Setting up a new profile is just about as easy. You just enter PROFILE LOGIN and the file name you want to execute as the profile.

The PROFILE command has another nifty feature. If you just enter >PROFILE LOGIN, you will know whether a login profile has been set up. If there is none, you will get a message that looks something like "LOGIN PROFILE IS NIL".

Now, if you did a >PROFILE LOGIN and there was a login profile in effect, you would expect to see the filename, right? Oh, well... no such luck. All you get is a 12-digit hexadecimal number, which bears no obvious relation to the file name.

Anyway, it's a neat command. It can come in useful for various things. If you use the same username every time, and nobody else uses that username, you could customize your environment somewhat... a few special abbreviations for often used commands, etc. If the switch you're using is an unusual beast (like an EIOC or something), you might want to use the login profile to make it look more civilized. Maybe make the operator profile do some things like datafilling the rest of the terminals and bringing them up.

Food for thought.

---

## QUIT

```
All Known Formats : >QUIT
                   >QUIT ALL
                   >QUIT <nlevels>
                   >QUIT <increment>
```

Opposite Command : No opposite command

Associated Commands : LEAVE

E.G. : >QUIT 3 -- quits out 3 levels  
>QUIT MAPCI -- quits out of MAPCI

---

QUIT is used everywhere. It is the all-purpose escape hatch. Again, the best way for me to do this is to give you a bunch of examples, so here goes:

CI:

```
>TABLE TERMDEV
TABLE TERMDEV:
>TABLE TRKGRP
TABLE TRKGRP:
>MAPCI
MAPCI:
>QUIT 2 -- quit 2 of the 3 levels
TABLE TERMDEV:
```

CI:

```
>TABLE TERMDEV
TABLE TERMDEV:
>TABLE TRKGRP
TABLE TRKGRP:
>MAPCI NODISP
MAPCI:
>MTC
MTC:
>IOD
IOD:
>IOC
IOC:
>QUIT -- quit 1 level
IOD:
>QUIT MAPCI -- quit MAPCI increment
TABLE TRKGRP:
>QUIT ALL -- quit all the way back to CI
CI:
```

---

READ

All Known Formats : >READ <filename>  
>READ PREVIOUS -- clear read stack  
>READ INPUT -- reads from keyboard

Opposite Command : No opposite command

Associated Commands : EXECUTE

E.G. : >READ MYEXEC -- read the file "MYEXEC"  
>(READ PREVIOUS;READ MYSELF) -- recursive read

---

This is the command that reads an exec file. That is, if you edit and fill a file full of commands, you can file it away and READ it at any time.

The READ <filename> is the most powerful format, as it gives you a full-fledged macro processing facility. The READ PREVIOUS is like the RETURN in BASIC. It returns to the previous calling exec. READ INPUT is useful for getting interactive input from the user from inside an exec file. I'll go into that later.

Enough talk... let's see some action:

```

>EDIT ONE_FILE          >EDIT TWO_FILE
>INPUT                  >INPUT
1::  PRINT '-- Here goes'  PRINT '-- Two-file here'
2::  1 -> XLEVEL          PRINT 'Do da, do da'
3::  READ TWO_FILE        IF (XLEVEL=1) THEN (
4::  PRINT '-- And again'   READ PREVIOUS)
5::  2 -> XLEVEL          PRINT 'More of the same'
6::  READ TWO_FILE        READ PREVIOUS
7::  PRINT 'All done.'
>>
>FILE SFDEV            >FILE SFDEV

```

And now, let's execute it:

```

>READ ONE_FILE
-- Here goes           Line 1 of ONE_FILE
-- Two-file here      Line 1 of TWO_FILE
Do da, do da         Exiting at line 4 of TWO_FILE
-- And again         Back at line 4 of ONE_FILE
-- Two-file here     Line 1 of TWO_FILE again
Do da, do da        Line 2 of TWO_FILE again
More of the same     About to exit TWO_FILE, line 6
All done.           Exiting ONE_FILE

```

By the way, I didn't really need that final READ PREVIOUS in line 6 of TWO\_FILE. SOS automatically does a READ PREVIOUS after the End-of-File. Either way, it's neat!

Of course you can do recursion. Just step this way:

```

>EDIT PREAMBLE          >EDIT DO_FOREVER
>INPUT                  >INPUT
1::  PRINT '-- Here we go'  % Do this until II>LIMIT
2::  1 -> II                IF (II > LIMIT) THEN
3::  5 -> LIMIT             (READ PREVIOUS)
4::  READ DO_FOREVER       PRINT '--- II: ' II
5::  PRINT '-- Bye for now' (1+II)->II
6::                          (READ PREVIOUS;
7::                          READ DO_FOREVER)
>>
>FILE SFDEV            >FILE SFDEV

```

It's pretty easy to see *what* is going to happen, but *how*??? Well, you see, the whole key is the statement in lines 6 and 7 of the DO\_FOREVER exec. Since it is in parentheses, the SOS CI

interpreter will sort of store it aside until it gets the closing parenthesis. Then it will do the `READ PREVIOUS`, which "clears the stack" as far as the call to `DO_FOREVER` is concerned. The next `READ DO_FOREVER` is then the only pending call. So, the exit in lines 2 and 3 finally returns to the caller that is on the stack. Namely, `PREAMBLE`.

`READ INPUT` is pretty neat, but can be awkward if you're not used to it. Let's look at a portion of an example exec:

```
COMMAND CM1 (@1 -> VARIABLE1;READ PREVIOUS)
PRINT 'Enter the first variable using the CM1 command.'
PRINT 'syntax: CM1 <variable1>'
READ INPUT
```

What this does is create a command, `CM1`, which will put its first parameter into `VARIABLE1`. The exec then prompts the user to use the `CM1` command to enter `VARIABLE1`. It then does a `READ INPUT`, which tells the switch to temporarily stop reading input from the file and start reading from the terminal. Of course, the drawback to doing this is that you are giving up control of execution to the user until they decide to use the `CM1` command. They could also just enter "`READ PREVIOUS`" and control would be returned to the exec without `VARIABLE1` being initialized. Anyway, it does add a new dimension to what you can do.

Well, that's about it as far as `READ` is concerned. If you're confused, go to the switch and try these little jewels. Just keep playing around and you'll get the hang of it. That's what I do.

-----  
`RECORD`

**All Known Formats :** `>RECORD QUERY`

`>RECORD START ONTO <device>`  
`>RECORD START FROM <device>`  
`>RECORD START FROM <device> ONTO <dev>`

`>RECORD STOP ONTO <device>`  
`>RECORD STOP FROM <device>`  
`>RECORD STOP FROM <device> ONTO <dev>`

**Opposite Command :** No opposite command

**Associated Commands :** `SEND`, `LOGUTIL` (`STARTDEV/STOPDEV`)

**E.G. :** `>RECORD START ONTO PRT0` -- send console output  
`>RECORD START ONTO MAP3` -- to various devices  
`>RECORD START FROM TATSIM2 ONTO PRT1`

-----

The whole idea is to create a fork for re-directing the I/O, but not disconnecting its original direction. That is, if we use `SEND PRT1`, all our console output goes to that device. However, if we use `RECORD START ONTO PRT1`, we still get the output on our terminal, in addition to getting a copy on the printer!

There are several bad things about `RECORD`. You must stop the recording on the very same device that started it. Also, `SOS` does not check to see if the recording is already started by another user. Thus, you can do the following:

```
At MAP1:
>RECORD START ONTO PRT1
```

```
At MAP3:
>RECORD START FROM MAP1 ONTO PRT1
```

And this will of course create two streams, and when you want to stop them, you should do the following:

```
>RECORD QUERY
From MAP1 ONTO PRT1 Started by MAP1
From MAP1 ONTO PRT1 Started by MAP3
```

```
At MAP1:
>RECORD STOP ONTO PRT1
```

```
At MAP3:
>RECORD STOP FROM MAP1 ONTO PRT1
```

That is the only way to stop it. Also, if you start the recording and then log out, the recording link is stopped if and only if your userid and the terminal ID are different. For example:

If in table TERMDEV, there is an entry for the printer called PRT1, and you have previously PERMITTED PRT1 as a username, then when you login on the PRT1 device, you get a message like this:

```
PRT1 logged in on device PRT1 6:33:22 Aug-21-89
```

Now, if someone starts a record start onto that device, it will continue even if you log out.

However, if you log in on a device called PRT2 (in table TERMDEV), and there is not a permitted user called PRT2, then you are prompted for a user-name. Call it BETA. And then when someone does a record start onto BETA (the user name), it will terminate when BETA logs out.

What actually happens is that you can do a RECORD START onto either a device or a user name. In case of a device and a user with the same name, RECORD will assume you mean the device.

---

REPEAT

All Known Formats : >REPEAT ### (any commands)

Opposite Command : No opposite command

Associated Commands : <BREAK> HX -- cancels execution

E.G. : >MAPCI;MTC;TRKS;TTP;POST D DTC 0 1 1 -- post a trunk  
>REPEAT 24 (BSY;BSY INB;NEXT) -- BSY INB ALL

---

REPEAT can also be nested (with care). It will not start executing until all "("s are matched with ")"s.

```
>EDIT ADD_MY_CKTS          -- create a file... just watch
>INPUT
ADD_CKT   CKT? DTC 0 1 ?
>>                                     -- exit input mode
>REPEAT 23 (INPUT LINESTR) -- Remember that LINESTR
                                     -- always contains the
>TOP;TYPE 555                       -- current line. See the
TOF                                       -- 24 copies of the same
  ADD_CKT   CKT? DTC 0 1 ? -- line.
  ADD_CKT   CKT? DTC 0 1 ?
  ....
  ADD_CKT   CKT? DTC 0 1 ?
EOF

>DOWN 1                               -- position on the line
ADD_CKT   CKT? DTC 0 1 ?
>1->II
>REPEAT 24 (CHANGE 2 '?' (NUMTODECSTR II);
>          (II+1)->II;DOWN 1) -- zzzzzzap

>TOP;TYPE 555                       -- voila!!
TOF
  ADD_CKT   CKT1 DTC 0 1 1
  ADD_CKT   CKT2 DTC 0 1 2
  ADD_CKT   CKT3 DTC 0 1 3
  ....
  ADD_CKT   CKT24 DTC 0 1 24
EOF

>FILE                                     -- save it quick
*** FILE --- NO WRITE VOLUME SPECIFIED -- oops
>FILE D010TRAFVOL
```

Neater-er and neater-er!

---

RESTART

```
All Known Formats : >RESTART WARM
                   >RESTART COLD
                   >RESTART RELOAD
                   >RESTARTBASE -- one word!!
```

```
Opposite Command : No opposite command
```

```
Associated Commands : None
```

```
E.G. : >RESTART COLD
       Confirm (YES or NO)
       >YES
```

---

The RESTART is really the same as a boot on most other systems, or an IPL in VM/CMS. It re-initializes the software.

```
>RESTART WARM    -- used to clear up minor problems
>RESTART COLD    -- to clear up major problems
>RESTART RELOAD  -- to clear up really major problems
>RESTARTBASE     -- used in dire circumstances indeed!
```

Otherwise, you can use the control switch on the CPU itself:

1. Flip "ACT/DEACT" switch to "DEACT" (this will take the CPU off-line).
2. Set the thumbwheel to "5".
3. Hit the "RESET" button.
4. Flip "ACT/DEACT" switch back to "ACT"

If you're really interested, the "SuperNode System Description" has a section devoted to restarts. It goes into all the gory details. Come to think of it, if you don't fully understand that section, you probably shouldn't be messing with RESTART anyway.

---

#### RINDEX

All Known Formats : >IF (RINDEX=0) THEN ...

Opposite Command : LINDEX

Associated Commands : LINDEX

E.G. : >IF (RINDEX < 3) THEN (PRINT 'Too few parms')

---

RINDEX is a function which returns the number of parameters to the right of the command. For example, if someone entered the command:

```
>HOWDY A B C D
```

and HOWDY did an RINDEX, 4 would be returned.

RINDEX is mainly used in commands with a variable number of parameters so that the command knows how many to read in.

---

#### SEND

All Known Formats : >SEND <device>  
>SEND SINK  
>SEND PREVIOUS  
>SEND SFDEV  
>SEND SFDEV <filename>  
>SEND <disk-vol> <filename>

Opposite Command : No opposite command

Associated Commands : RECORD, LOGUTIL (STARTDEV/STOPDEV), <BREAK> HT

E.G. : >SEND PRT1 -- redirect all output to a printer

---

SEND is used to create a total branch point in your output, so that it either goes to another device, a file, or the bit bucket. For example:

```
>SEND PRT1
```

would send all of your output to PRT1 (and you cannot see what you're doing). You probably should enter:

```
>RECORD START ONTO PRT1
```

instead.

Now, let's look at some other examples:

```
>SEND PREVIOUS    -- Re-direct the output to where it was
                  -- going before the last SEND command was
                  -- issued.

>SEND SINK        -- discard all output

>SEND SFDEV      -- send output to the file CONSOLE in SFDEV

>SEND SFDEV MY_LIST  -- Same as above, but name of file
                  -- will be "MY_LIST".

>ERASESF CLLI_LIST  -- destroy any old copy of the file
>TABLE CLLI
>SEND SFDEV CLLI_LIST
>LIST ALL
>SEND PREVIOUS
```

This is called "capturing data". All of the junk that would normally be sent to your terminal is sent to the file. Make sure that you do an "ERASESF" first so that the output does not append to the file or create problems.

```
>EDIT CLLI_LIST 13  -- go look at the file, truncate
                  -- at column 13
>END;PRINT LINE    -- get the size of the file
138
>TOP;DOWN 1       -- get ready to make an exec
MDAL2WDTGS01
>REPEAT 138 (CHANGE ' ' 'POS ' ;DOWN 1) -- prefix each
>TOP;TYPE 2
TOF
POS MDAL2WDTGS01
>FILE
```

And to use it, say in trunk group,

```
>TABLE TRKGRP
TABLE TRKGRP:
>READ CLLI_LIST
```

Of course, for TRKSGRP or TRKMEM, we need the sub-group number. However, we could go into table TRKMEM, send to a file, capture all the stuff, and then we edit the file, truncate the output, and save the file.

Then, to make a trunk deleter, we could do the following:

```
>TABLE TRKMEM
>SEND SFDEV DEL_TRKS
>LIST ALL
>SEND PREVIOUS

>EDIT DEL_TRKS 18 -- go look at the file, truncate at column 18

>END;PRINT LINE -- get the size of the file.
142
>TOP;DOWN 1 -- get ready to make an exec
>REPEAT 142 (CHANGE ' 'DELETE ';DOWN 1)
>FILE

>TABLE TRKMEM
>OVER;VERIFY OFF
>READ DEL_TRKS -- Zappo... away go all the trunks which
-- have been previously set to INB at
-- the TTP level.

>QUIT
```

---

#### SETDATE

All Known Formats : >SETDATE date month year

Opposite Command : No opposite command

Associated Commands : DATE

E.G. : >SETDATE 31 7 89 -- July 31, 1989  
>SETDATE 1 2 90 -- February 2, 1990

---

You'd think they would know better, but the year is only two digits. 19xx is assumed. Oh, well.

---

#### SETTIME

All Known Formats : >SETTIME hours minutes

Opposite Command : No opposite command

Associated Commands : TIME, DATE

E.G. : >SETTIME 9 00 -- set time for 9:00am

>SETTIME 21 00 -- set time for 9:00pm

---

I think that about covers it.

---

#### SHOWFL

All Known Formats : >SHOWFL <filename>

Opposite Command : No opposite command

Associated Commands : DSKUT, LISTVOL, PRINT

E.G. : >LISTVOL D010CNTFL ALL  
>SHOWFL GET\_ALTTLIB\_READY

---

See discussion under DSKUT.

---

#### SHOWVOL

All Known Formats : >SHOWVOL <volume>  
>SHOWVOL <volume> ALL

Opposite Command : No opposite command

Associated Commands : DSKUT, LISTVOL, LISTST

E.G. : >SHOWVOL D010CNTFL -- short display  
>SHOWVOL D010IMAGE ALL -- long display

---

See discussion under DSKUT.

---

#### SLEEP

All Known Formats : >SLEEP <seconds>

Opposite Command : No opposite command

Associated Commands : None

E.G. : >SLEEP 3 -- wait 3 seconds before continuing

---

'nuff said.

---

#### TAPE

All Known Formats : >Too many to go into here.

Opposite Command : No opposite command

Associated Commands : MOUNT, DEMOUNT, LIST, TLIST

---

The TAPE command is a very powerful command which allows you to do all sorts of unusual things to a tape. You can rewind it without DEMOUNTing it, you can read an EBCDIC file off the tape and translate it to ASCII at the same time, you can enter data on a tape one record at a time, erase files off a tape, etc.

If you're going to be doing a lot of stuff with tapes, you should probably get familiar with this command. A full explanation is in document NTP 297-1001-509 (of course). You might even want to take a look at NTP 297-1001-118 (*DMS-100 Family Magnetic Tape Reference Manual*). It tells you all about the block structure and all that.

---

TAPECONFIRM

All Known Formats : >TAPECONFIRM ON  
>TAPECONFIRM OFF

Opposite Command : No opposite command

Associated Commands : MOUNT

---

This is a silly little command, but it does come in handy occasionally. You know, when do something like:

```
>MOUNT 0 FORMAT
```

and the switch makes you enter the name of the first file on the tape to confirm that you really want to format the tape? Well, if you say:

```
>TAPECONFIRM OFF
```

you don't have to enter the file name. This may not sound like it's worth an entire command, but it does make it a lot easier when you mount tapes with long file names.

---

THEN (See the discussion under IF)

```
E.G. : IF (X=3) THEN (  
        PRINT '--- Ah-ha, X is three.'  
        ) ELSE (PRINT '--- Well, X is certainly not 3.')
```

---

---

**TIME**

All Known Formats : >TIME

Opposite Command : No opposite command

Associated Commands : DATE, SETTIME

---

TIME displays the time in hours, minutes, and seconds.

---

**TLIST**

All Known Formats : >TLIST <tape drive>

Opposite Command : No opposite command

Associated Commands : MOUNT, DEMOUNT, LIST

E.G. : >MOUNT 0  
>TLIST T0

---

TLIST is basically a fast version of LIST. You can't always use it, though. When you mount a tape, sometimes you'll see the message, "First file = TAPE\$DIR...". If you see that message, you can use TLIST. TAPE\$DIR is created by many different utilities, including XDMSTAPE on the IBM. It is a list of the files on the tape. TLIST reads that list into your CI process symbol table so that you can access the files on the tape.

The reason TLIST is usually preferred over LIST is that LIST traverses the used portion of the tape looking for files. If you have a lot of files on the tape, this could take a while. TLIST takes just a few seconds. As far as I can tell, there is no reason to use LIST instead of TLIST when it's possible to use TLIST. If anybody knows of any advantages in using LIST over using TLIST, let me know.

---

**UNLOAD**

All Known Formats : >UNLOAD <module name>  
>UNLOAD <module name> DISPLAY  
>UNLOAD <module name> NOINFORM  
>UNLOAD <module name> NOWARN  
>UNLOAD <module name> NOERROR

Opposite Command : LOAD

Associated Commands : See DIS document OSA10 LISTING

---

UNLOAD is one of those commands you shouldn't use unless you *absolutely* know what you're doing. If you think you need to know more about this command, DIS "OSA10 LISTING".

---

**UNPERMIT**

All Known Formats : >UNPERMIT <username>

Opposite Command : PERMIT

Associated Commands : SHOW USERS, PRINT USERS, FORCEOUT

E.G. : >FORCEOUT OPERATOR  
>UNPERMIT OPERATOR

---

UNPERMIT discontinues a username on the DMS-100, thus revoking their access to the switch. UNPERMIT will not work as long as the user is logged in. However, if you use FORCEOUT to log the user out, you can then UNPERMIT him.

---

**UNTIL**

All Known Formats : >UNTIL (expr) (cmds)

Opposite Command : No opposite command

Associated Commands : IF, WHILE

E.G. : >UNTIL (A=10) ((A+1)->A;PRINT A)

---

The UNTIL command works the same as an UNTIL statement in any high-level language (Protel included). It is a loop statement with an exit test at the end. Thus:

```
>1->A;UNTIL (A=10) ((A+1)->A;PRINT A)
```

would print the numbers 2 through 10, and:

```
>10->A;UNTIL (A=10) ((A+1)->A;PRINT A)
```

would execute forever, since the first A=10 test is done after (A+1) ->A has been executed.

---

**WHILE**

All Known Formats : >WHILE (expr) (cmds)

Opposite Command : No opposite command

Associated Commands : IF, UNTIL

E.G. : >WHILE (A<10) ((A+1)->A;PRINT A)

---

The `WHILE` command works the same as a `WHILE` statement in any high-level language (Protel included). It is a loop statement with an exit test at the beginning. Thus:

```
>1->A;WHILE (A<10) ((A+1)->A;PRINT A)
```

would print the numbers 2 through 10, and:

```
>10->A;WHILE (A<10) ((A+1)->A;PRINT A)
```

wouldn't print anything, since the first `A=10` test is done before `(A+1)->A` is been executed.

### **NCMS -- A Special Note**

"Why in the world are you talking about that dinosaur, NCMS? Don't you know that everyone uses MPCNET now?" Well, not everyone. I know they have it in RTP, I think they have it in Meriline, and I'm almost certain they have it in Carling (they have everything in Carling). However, we don't have it in Richardson yet. Until MPCNET is universal, this section will remain. Since I'm in Richardson, I can't really tell you much about MPCNET, but the MPCNET user's guide (`DIS "REF JPF"`) is probably what you really need. If you want to see what little I know about MPCNET, see the MPCNET section.

Now, everyone knows how to use NCMS. The real problem is with the transfer of files.

```
>DSKUT
>LISTVOL D010TEST1 ALL
MY_OWN_EXEC
MY_OWN_EXEC_DATA

>LISTSF
MOVE_UP
>PRINT MOVE_UP

% $FILE=MOVE_UP (Exec to move all my files up to the IBM)
COMMAND NCPY (PRINT (SYMTOSTR @1);SOSTOCMS @1 @2 @3)
%
NCPY MY_OWN_EXEC MY_OWN_EXEC SOS_EXEC
NCPY MY_OWN_EXEC_DATA MY_OWN_EXEC_DATA SOS_EXEC
```

Now, the only problem is that on the IBM, we can only have 8-character names, so that the first file will actually appear as "MY\_OWN\_E SOS\_EXEC", and when the next file, "MY\_OWN\_EXEC\_DATA" tries to come in ---- BLAMMO!! So, it is necessary to make sure that the destination file names are of the right length. Thus, the two `NCPY` lines in the exec above should be something like:

```
NCPY MY_OWN_EXEC MY_EXEC SOS_EXEC
NCPY MY_OWN_EXEC_DATA MYX_DATA SOS_EXEC
```

Now, we can `>READ MOVE_UP` and it should do a lot better.

## NCMS -- Wonderlink Between the Switch and the IBM

When you are on the switch, you can use NCMS to link back to the IBM. Normally, the link will be up when you try to use it. If not, see the next page. If the link is up, the following procedure should log you into the IBM:

```
>NCMS
>SMODE          -- get to the IBM
>RCH or TEX    -- depending on whether you want
                -- on BNRRCH or BNRTEX
<BREAK>
?HX           -- you now have about 1 minute to log in to the IBM
>NCMS         -- since HX threw you all the way back to CI
>SMODE        -- back to the IBM again
>L <vmid>     -- log into the IBM
><password>
. . . .
>SMODE        -- get back to the switch
>SCMSTOSOS <fn> <ft> <sos fn> <volume> -- from IBM to switch
or
>SOSTOCMS <sos fn> <fn> <ft>      -- from switch to IBM
```

When you're done, be sure to use:

```
>CMSLOGOFF
```

while you're in switch mode to log you off the IBM.

The process is different if you want to copy a module across to be loaded on the switch. Instead of SCMSTOSOS, you use GETLOAD, or GL for short. It goes something like this:

```
>GL MYMOD
```

This will put your module in SFDEV. Then you can use the LOAD command to load it.

## Common Problems and Mistakes with NCMS

- All copy and logout functions *must* be performed from the switch. Thus, you must be on the switch and *not* the IBM to copy files either way or to use the CMSLOGOFF command. This is because the NCMS routine is controlling the file transfer and it resides on the switch. The IBM has *nothing* to do with the file transfer. If you use SMODE and transfer back to the IBM and then try to do the SCMSTOSOS, you will get an error.
- The location of the link on the switch varies from switch to switch. It is customarily posted somewhere in the lab. It must be on-line (RTS) in order for NCMS to work.
- The datafill looks something like this:

```
>TABLE DLCDEV
>LIST ALL

DLCNUM   IOCNO   IOCCKTNO   BAUD   EQPEC
-----
         0         2         0  B4800  1X67BA
```

If the data isn't there, you will have to add it.

- You may have to reset the circuit in IOD:

```
>MAPCI;MTC;IOD;IOC 2;CARD 0  
>BSY;OFFL;BSY;TST;RTS
```

- A special note to side 1 NCMS users: If when you type NCMS, you're having some trouble getting a link, it could be that NCMS is trying to use the DLC on side 0. There are 2 solutions to this problem. First, you can go into MAPCI and busy out the DLC on side 0. Second, you can just type:

```
>NCMS 1
```

instead of:

```
>NCMS
```

This format just tells NCMS to use the DLC on side 1.

Any time the link acts balky, reset it and log back in.

## The Ultimate NCMS Sin

This part gets a little technical on the VM/CMS side. If you don't know what it means, you probably don't need to know about it.

Many people use VM facilities like WAKEUP to trap console I/O. Many people also DISCONNECT rather than logging off sometimes. If both of these apply to you, you may have some trouble with NCMS.

If you're disconnected and are using WAKEUP, and you log in via NCMS, you have one command before NCMS freezes and you have to do a <BREAK> HX. Make it a good one. Use this one: IPL CMS PARM AUTOOCR. This will, of course, restart your VM session, and whatever installs your WAKEUP stuff should be smart enough to know better when it sees that you're not on a full-screen terminal.

If you forget and hang NCMS, do *not* use PRIVREQ to log yourself off the IBM. I have experienced the consequences of this once. When we queried the userid on VM, it was listed as PENDING LOGOFF. We had to get network services to get the userid logged off, and it took a system IPL of the IBM to get the IBM to release the NCMS link. REALLY NASTY STUFF!!!

A <BREAK> HX is infinitely preferable.

## MPCNET -- What Little I Know

### Speculation

Sorry, folks, but, never having had the opportunity to use MPCNET myself, there's not a lot I can offer you in the way of instruction. This section is basically a brief overview of MPCNET, with comparisons to NCMS. I'll even throw in a few speculations, but I won't guarantee any of them. If you want to know about the commands themselves, you need to be looking in the MPCNET user's

guide. That's "JPF" in DIS.

What is MPCNET, anyway? Well, basically, it's NCMS, only better. MPCNET is a 19.2 kbps link between the switch and another machine (currently, only IBM mainframes are supported: `fnref REFID=spec.`). That's four times as fast as current NCMS speeds.

Besides the speed increase, there are a few other nifty features MPCNET has over NCMS. First, more than one person can use MPCNET at the same time. So you can be logged in on BNRTEX while your buddy's logged in on BNRRCH from the same switch. There can be up to 255 simultaneous sessions on each switch. I'm not an expert, but I can't see how more than one person could log in to the same IBM over the same link at the same `time:fnref REFID=spec.`, so that may be a limitation.

Second, you don't have any limit on the number of records you can transfer over the link. If you've ever tried to transfer something large over NCMS, you know how painful it can be to be almost done and have NCMS abort because it thought there were too many records. Don't have to worry about that with MPCNET!! If you want to transfer the encyclopedia Britannica over the link, go for it!!

Third, you can transfer SuperNode modules over the link. Unless I'm mistaken, there is no facility for that in `NCMS:fnref REFID=spec..` With MPCNET, just issue the magic `GL68K` command, and it happens!

Well, that just about exhausts my knowledge of MPCNET. I hope it helped somewhat.

## **SOS Execs**

SOS execs, like execs on other systems, are basically files containing a list of commands to be executed by the system's command interpreter.

This section is devoted to showing you some of the finer points of exec writing in the DMS-100 environment. For your reading enjoyment, many examples of SOS execs (which actually work!!) have been included.

There are two ways to invoke execs. The orthodox way is by use of the `READ` command, like so:

```
>READ MYEXEC
```

The other way to invoke an exec involves the use of the `COMMAND` command, like this:

```
>COMMAND DOIT MYEXEC    -- Look, Mom!! No parentheses!!
>DOIT                   -- does a READ on MYEXEC
```

Now, you may ask yourself, "What difference does it really make?" Well, here's some food for thought:

```
>EDIT GOFORIT
>INPUT
PRINT 'here I am'
PRINT (SYMTOSTR @1) (SYMTOSTR @2) (SYMTOSTR @3)
>>
```

```

>FILE SFDEV
>LISTSF

>READ GOFORIT A B C
READ -- Wrong number of parameters    -- oops...
                                         -- can't pass parms

>COMMAND DOIT GOFORIT
>DOIT A B C
here I am
A B C

```

So, you can see that making a command point to an exec makes it much easier to pass information into the exec.

Well, enough of this. I'm spoiling all the surprises!! On with the show.

## Exec Files, Some Simple Examples

Let's say that we want to make up some files that will do lots of neat things for us. For example, let's edit and save the following files into SFDEV:

```

>EDIT PART1
>INPUT
% $FILE=PART1 (Example of an exec file) RLT JUN-21-86
PRINT '-- Hi-ho part-one fans... here we go'
COMMAND PIXXY ((SYMTOSTR @1)->$PARM1;
              PRINT '-- Parm-1 set: ' $PARM1)
COMMAND HELP_PART (PRINT PART_HELPF)
COMMAND PART_HELP (PRINT PART_HELPF)
0 -> II
READ PART2
>>
>FILE SFDEV

>EDIT PART2
>INPUT
% $FILE=PART2 (Example of an exec-file) RLT JUN-21-86
PRINT '-- PART2 here. To store an alpha parm, enter:'
PRINT '-- >PIXXY <value>'
PRINT ' '
PRINT '-- For help, enter: >HELP_PART'
PRINT --- To reinitialize, enter: >READ PART1'
PRINT ' '
>>
>FILE SFDEV

>EDIT PART_HELPF
>INPUT
% $FILE=PART_HELPF RLT JUN-21-86
PRINT ' '
PRINT ' O N - L I N E H E L P FOR: PART'
PRINT ' '
PRINT ' The PART family of commands are used to'
PRINT ' illustrate basic practices of writing SOS'
PRINT ' execs. They don't really do anything at all.'
PRINT ' The following commands are available:'
PRINT ' '
PRINT ' >PIXXY <alpha-parm> -- will store $PARM1'
PRINT ' >HELP_PART -- prints this info'
PRINT ' '

```

```

PRINT ' To re-initialize the PART system, enter:
PRINT ' '
PRINT ' >READ PART1'
PRINT ' '
>>
>FILE SFDEV

```

So, on the previous page we see a little family of execs. And how cute they are, too! This is about the bare-bones of what I consider essential to any family of execs. We have a start-up exec (PART1), a main driver exec (PART2), and a baby-bear exec, er, ah, the help file (PART\_HELPF) which is invoked by either HELP\_PART or PART\_HELP (in case the user forgets the format). There is a slicker way to do on-line help, but we won't get into that yet.

Thus, for our PIXXY command, we might use the following as the help file:

```

% $FILE=PIXXY_HELPF                                RLT JUN-21-86
PRINT ' '
PRINT '          O N - L I N E   H E L P           FOR: >PIXXY'
PRINT ' '
PRINT ' The PIXXY command allows the user to store any'
PRINT ' alpha parm in the variable $PARM1'
PRINT ' '
PRINT ' USAGE: >PIXXY <parm>
PRINT ' '

```

And to activate the command, we would add the following to PART1:

```

COMMAND HELP_PIXXY (PRINT PIXXY_HELPF)
COMMAND PIXXY_HELP (PRINT PIXXY_HELPF)

```

Now, on to bigger things:

### **Fool-Proof Execs: In-Line HELP for Commands**

To help the user, you can install an in-line help in the command itself. In the following command, the "guts" are in the ELSE clause. The help is in the first part.

```

COMMAND START (
IF ((RINDEX) = (0)) THEN (
PRINT 'COMMAND START NEEDS 1 PARAMETER: ';
PRINT '          <CIRCUITNAME>|<ALL>'
)ELSE(
START_FUNC -> FUNCTION;
(SYMTOSTR @1) -> DIRNAME;
READ PROCREQ
)
)

```

The way this one works is to see if there is no parm entered on the line. This means that the user needs help. Of course, we can also check for the word "HELP" to the left of the command:

```

>HELP START

```

In this case, we need to check the left-index (LINDEX) instead:

```
COMMAND START (
IF ((LINDEX) = (1)) THEN (
  IF ((SYMTOSTR @-1) = 'HELP') THEN (
    PRINT 'COMMAND START NEEDS 1 PARAMETER: ';
    PRINT '      <CIRCUITNAME>|<ALL>'
  ))ELSE(
    START_FUNC -> FUNCTION;
    (SYMTOSTR @1) -> DIRNAME;
    READ PROCREQ
  )
)
```

Actually, the way most commands determine whether or not to print help is to use LINDEX to see if there's anything before the name of the command on the line. If so, assume it's a "Q" or a "HELP" and print the help file. This keeps it consistent with system commands (commands someone wrote in Protel, compiled, and loaded onto the switch).

### **Fool-Proof Execs: Interactive Execs**

This isn't a very safe thing to do in a SOS exec, but I'll show it to you anyway because you might come up with a better use for it.

You've already seen how, by making a command invoke an exec, you can pass parameters to the exec. But what if you have a rather complex exec and you want to ask the user a question in the middle of it? How do you do that? Well, you do it verrry carefully!

A maneuver like this requires rather tricky use of the READ command. Specifically, the "READ INPUT" variety of the READ command. Now, what "READ INPUT" actually does is tell CI to start looking at the terminal for input. Very much like the "shell" commands you can use in some PC programs to jump into a DOS session without leaving the original program. That means that your exec is suspended until a "READ PREVIOUS" occurs, at which point it takes up where it left off. However, with careful planning and voluminous prompting of the user, it is possible to minimize your risk. Observe the following portion of an exec:

```
>EDIT COPYSF
>INPUT
% $FILE=COPYSF (Example of an exec) v-1.00 BLH NOV-10-89
PRINT 'How many files to copy?'
PRINT 'Enter "NUMFILES n"'
COMMAND NUMFILES (@1 ->NUMFL; -- Command to allow the user
  READ PREVIOUS) -- to specify num of files
READ INPUT -- transfer control to the user.

COMMAND FILEDEV (@1->FNAME; -- Command to allow the user
  @2->DEV; -- to specify file name and
  READ PREVIOUS) -- destination device.

REPEAT NUMFL (
  PRINT 'What's the next filename and device?'
  PRINT 'Enter "FILEDEV <filename> <device>"'
  READ INPUT
  COPY FNAME DEV
)
```

As you can see, when you use this method, you are depending on the user to enter what you tell him when you tell him. Does it work? Yes. Is it risky? Oh, yeah!! But it is an interesting concept.

### **Fool-Proof Execs: WHILE and UNTIL Loops**

As you may have noticed, the previous version of `COPYSF` is rather limiting in that you have to tell it how many files you want to copy before you copy them. Of course, a much better way to do this would be to loop until the user signalled you to stop.

With that in mind, look at this new and improved version of `COPYSF` which uses the `UNTIL` command:

```
>EDIT COPYSF
>INPUT
% $FILE=COPYSF (Example of an exec) v-1.00 BLH NOV-10-89

COMMAND FILEDEV (@1->FNAME;          -- Command to allow the user
                @2->DEV;             -- to specify file name and
                READ PREVIOUS)      -- destination device.
UNTIL (((SYMTOSTR FNAME) = 'NONE') & ((SYMTOSTR DEV) = 'NONE')) (
    PRINT 'What''s the next filename and device?';
    PRINT 'Enter "FILEDEV <filename> <device>"';
    PRINT 'Enter "FILEDEV NONE NONE" to quit.';
    READ INPUT;
    COPY FNAME DEV
)
```

Of course, in this example, if the user changes his mind up front and enters "NONE NONE" the first time through, the exec will fail miserably. That's why God (or IBM or somebody) invented the `WHILE` loop. Take a look at the final version of `COPYSF` which doesn't fail quite so easily:

```
>EDIT COPYSF
>INPUT
% $FILE=COPYSF (Example of an exec) v-1.00 BLH NOV-10-89

COMMAND FILEDEV (@1->FNAME;          -- Command to allow the user
                @2->DEV;             -- to specify file name and
                READ PREVIOUS)      -- destination device.

PRINT 'What''s the first filename and device?'
PRINT 'Enter "FILEDEV <filename> <device>"'
PRINT 'Enter "FILEDEV NONE NONE" to quit.'
READ INPUT

WHILE (((SYMTOSTR FNAME) ^= 'NONE') & ((SYMTOSTR DEV) ^= 'NONE')) (
    COPY FNAME DEV;
    PRINT 'What''s the next filename and device?';
    PRINT 'Enter "FILEDEV <filename> <device>"';
    READ INPUT
)
```

Now you're an expert on `WHILE`, `UNTIL`, and `READ INPUT`. How do you feel? Well, take some aspirin, and maybe it'll go away.

Now, wouldn't it be nice if we could just do away with that silly `FILEDEV` command and specify everything on the command line?

### **Fool-Proof Execs: The PAR Function**

You've seen `PAR` before. You remember the "@" symbol you use to reference parameters? Well, the `PAR` function does that same thing, but by looking at the following exec, you can see the advantage of the `PAR` function in certain situations. This differs from the previous examples in that, because I'll be using parameters, the exec actually defines a command to do the work.

```
>EDIT COPYSFEXEC
>INPUT
% $FILE=COPYSFEXEC (Example of an exec) v-1.00 BLH NOV-10-89

ERASE COPYSF
COMMAND COPYSF (
  2 -> NUMPARMS  %% -- I'll use this as an index to my parms
  WHILE ((RINDEX) >= (NUMPARMS)) (
    COPY (PAR (NUMPARMS-1)) (PAR (NUMPARMS)));
    NUMPARMS + 2 -> NUMPARMS
  )
)
```

You can see how using `PAR` instead of `@` allows me to use `NUMPARMS` as an index to which parm I want to look at. So, there's one more little tidbit of information to stuff into your brain!

### **Fool-Proof Execs: Using Flags**

Now, let's return to our previously scheduled program, `PART1`. Geez... I can hardly remember that far back. Consider a new version of `PART1`:

```
>EDIT PART1
>INPUT
% $FILE=PART1 (Example of an exec) v-1.01 RLT JUN-21-86
PRINT '-- Hi-ho part-one fans... here we go...'
0->$REINIT_FLAG
$REINIT_OK -> $REINIT_FLAG

IF ($REINIT_FLAG = 0) THEN (PRINT '*** TO REINIT, ENTER: ';
  PRINT '    >REINIT_PART'; PRINT ' ')

COMMAND REINIT_PART (1->$REINIT_OK; READ PART1)

COMMAND PIXXY (.....
  .....

0->II
0->$REINIT_OK
READ PART2
```

In this case, we have a fool proof means that the `PART2` exec will not be accidentally re-read. Let's say that we were to try and re-read the `PART1` exec again.

- The value of the `$REINIT_FLAG` would be set to 0, and so we would get the message "\*\*\* TO REINIT, ENTER: ...." etc.

- If we enter the `REINIT_PART` command, it sets the flag and will cause the `READ_PART` to proceed onward.

There is only one more little detail, and that is "What is the value of `$REINIT_FLAG` the very first time `PART1` is read?" Actually, it is probably undefined (unless the user accidentally gave it a value). For this reason, SOS will print a message like "undefined value as parameter -1". To hide this from the user, so that s/he won't get too disturbed, we usually put all of the definitions in a "SEND SINK . . . . SEND PREVIOUS" block. This will send the output (from error messages, prints, etc.) to the bit bucket (known as the `SINK` in SOS -- remember SOS BILGE?). So our exec now looks like this:

```
% $FILE=PART1 (Example of an exec) v-1.01 RLT JUN-21-86
PRINT '-- Hi-ho part-one fans... here we go...'
0->$REINIT_FLAG
SEND SINK
    $REINIT_OK -> $REINIT_FLAG
SEND PREVIOUS
```

### **Fool-Proof Execs: Erase It Before You Define It**

So far, we have assumed that we are running a perfectly debugged system. Ho, ho, ho! There is just one problem with that. What if we (yes, we, ourselves) are trying to debug these confounded commands, and we re-run the `PART1` exec -- after we have changed the definition of a command in the file. SOS will not change the definition!!! This means that, even though we have made a change to the `PART1` file, and re-read the file, the *old* definition is all SOS knows about.

For this reason, it is always a good idea to erase the old commands at the beginning of the file that defines them. And *then* define them anew. Of course, the first time we read the file, we will get an "undefined symbol" error (obviously there is nothing to erase). So, we simply put the `ERASE` command in the `SEND SINK . . . . SEND PREVIOUS` block as well:

```
% $FILE=PART1 (Example of an exec) v-1.01 RLT JUN-21-86
PRINT '-- Hi-ho part-one fans... here we go...'
0->$REINIT_FLAG
SEND SINK
    ERASE HELP_PART HELP_PIXXY PIXXY REINIT_PART
    ERASE ZZZZ_CMD
    $REINIT_OK -> $REINIT_FLAG
SEND PREVIOUS
```

This way the user is not bothered by all those pesky SOS warnings.

In a related topic, remember at the first of this section where I showed you how to make a command invoke an exec so that the exec can take parameters? Well, if you change that exec and re-file it, the command is still pointing where the old version used to be. Consequently, you need to `ERASE` the command and define it again every time you change the exec. Details, details, details!!

### **\$\$PUSH and \$\$POP (Examples)**

Now, let's look at a very useful set of execs that we will define within the edit environment:

```
1 : % $FILE=AUX_CMDS (Helpful editor cmds) RLT JUL-24-86
2 : SEND SINK
```

```

3 : ERASE $$GET $$GETX $$GETSIZE $$MAKEEND $$POPALL
4 : ERASE $$POPX $$PUSHX $$PUT $$RECOVER STACK
5 : EDIT JUNKX
6 : COMMAND $$PUSHX ((1+XPTRX)->XPTRX;
7 :   ('XTEMP'+(NUMTODECSTR XPTRX))->VARX;
8 :   (LINESTR -> (STRTOSYM XVARX)) )
9 : %
10 : COMMAND $$PUT (@1->II;REPEAT II ($$PUSHX;DOWN 1))
11 : %
12 : COMMAND $$POPX (IF ((XPTRX) >= (1)) THEN (
13 :   INPUT (STRTOSYM ('XTEMP'+(NUMTODECSTR XPTRX)));
14 :   (XPTRX-1)->XPTRX) )
15 : %
16 : COMMAND $$GETX (IF ((IPTRI) <= (XPTRX)) THEN (
17 :   INPUT (STRTOSYM ('XTEMP'+(NUMTODECSTR IPTRI)));
18 :   (IPTRI+1)->IPTRI) )
19 : %
20 : COMMAND $$POPALL (IF ((XPTRX) >= (1)) THEN (
21 :   XPTRX->II;REPEAT II ($$POPX)) ELSE (
22 :   PRINT '** NO PARMS ON STACK' ) )
23 : %
24 : COMMAND $$GET (IF ((XPTRX) >= (1)) THEN (
25 :   1->IPTRI;XPTRX->XHOLDXPTR;REPEAT XPTRX ($$GETX)
26 :   0->XPTRX;
27 :   PRINT '--- ' XHOLDXPTR ' pulled from stack.';
28 :   PRINT '   Use $$RECOVER_STACK to do an un-get'
29 :   )ELSE(PRINT '** NO PARMS ON STACK' ) )
30 : %
31 : COMMAND $$RECOVER_STACK (IF ((XHOLDXPTR) > (XPTRX))
32 :   THEN(XHOLDXPTR->XPTRX;
33 :   PRINT '--- Recovered ' XHOLDXPTR ' from stack.'
34 :   )ELSE(PRINT '*** Stack is already at maximum.' ) )
35 : %
36 : COMMAND $$GETSIZE (SEND SINK;END;LINE -> $SIZE;TOP;
37 :   SEND PREVIOUS;PRINT '-- File has ' $SIZE ' lines.')
38 : %
39 : COMMAND $$MAKEEND (END;
40 :   INPUT '%% END -> END %%% END OF FILE')
41 : %
42 : QUIT
43 : SEND PREVIOUS
44 : %%% VALUES & VARS %%%
45 : 0 -> IPTRI;   0 -> XPTRX;   0 -> XHOLDXPTR
46 : 0 -> II
47 : 0 -> $SIZE

```

Now, let's look in some detail at this exec.

First of all, in line 1 we have a comment telling what the name of the exec is, "\$FILE=file-name". I have gotten in the habit of saying "\$FILE=". Then, I can use the editor to locate the beginning of file easily. Notice that I have a brief explanation (not much help here), then my initials, and the last date I changed it.

At line 2, I send the output to the SINK (bit bucket). This way, the user won't see the silly "EDIT:" prompt that will blip up every time the exec creates a command.

Lines 3 and 4 erase the commands I am about to create. This way, any old, spurious definitions for those commands will be sure to be erased.

In lines 5 and 42 we enter and exit the editor. I only want to make sure that the commands are properly linked, so I will edit a dummy file (JUNKX) and *then* create the commands, and then just quit out of the editor.

In lines 6 through 8, I define a nifty little command to simulate pushing the line the editor is currently on into a stack. In reality, I increment a counter (XPTRX) and then build up the name of a variable into which I will store the current line's contents. Let's say that XPTRX was initially 0. In that case:

```
(1+XPTRX) ->XPTRX would set it equal to 1. Then, the nested "(NUMTODECSTR
XPTRX)" turns the integer into a string, giving '1' from the value 1. Next, I build the
name of the variable in the symbol XVARX:
```

```
'XTEMP'+ '1' ->XVARX
```

So line 7 makes XVARX = 'XTEMP1'.

And then, in line 8, I create a symbolic reference to this string variable: (STRTOSYM XVARX) which would accomplish the same thing as if I had typed in "LINESTR -> XTEMP1".

And LINESTR is, of course, the editor system variable wherein the contents of the current line are always stored (done internally by the editor).

Thus, we have the first 72 characters of the current line stored in a variable called XTEMP1. And it is parameterized based on the current value of XPTRX. This leads us to...

Line 10, where we make a multiple store command called \$\$PUT. To use it, we position ourselves in the editor and enter >\$\$PUT 5 -- which would store the next five lines for us. Now, notice two things. First, the parm 5 is a numeric value and is *not* converted to a symbol (as would be the case in general). Next, the value appears inside the \$\$PUT command as "@1" (since it is the first parm passed to it when we evoke the command). Next, I store the value in a temporary variable called "II" and use it to start a repeat loop: \$\$PUSHX and then move down one line in the file.

Before we go too much further, I should make it clear that when we READ the file on the second previous page, none of these commands is actually executed. The only thing that is done is to create a new definition in your read/write directory that you can use later (at any time up until you logout). That is what the "COMMAND" command does. It creates a new command! Now, back to our file...

So, of course, what we store, we can retrieve. To do this, you just do the inverse of what the \$\$PUSHX did by decrementing the pointer XPTRX. Of course, we need to be careful that it isn't negative or zero:

```
12 :  COMMAND $$POPX (IF ((XPTRX) >= (1)) THEN (
```

That is, if and only if XPTRX is still greater than 0, do the following:

First, build the variable's name back up: 'XTEMP1' or whatever (in the case where

```
XPTRX = 1).
```

Then, convert the string to a symbol so that when we INPUT it, we get the VALUE of the variable, not the name of the variable. For example, if we were to do the following, this is what we would see:

```
>PRINT XPTRX
1
>PRINT (NUMTODECSTR XPTRX)
1 -- we can't see it, but this is really '1'
>PRINT ('XTEMP'+(NUMTODECSTR XPTRX))
XTEMP1
>PRINT (STRTOSYM ('XTEMP'+(NUMTODECSTR XPTRX)))
Mary had a little lamb.
```

And there it is, the value of the line that we stored when we were editing the file. (Assuming things went something like on the next page.)

And finally in line 14, we decrement the counter XPTRX.

Now, the only problem with the \$\$POPX command is that the lines will come out in the *opposite* order in which we put them in! (Just like any good push-pop stack should.) So, we create a "normal" command \$\$GET, which will correspond to our \$\$PUT <nlines> command:

```
24 :   COMMAND $$GET (IF ((XPTRX) >= (1)) THEN (
25 :     1->IPTRI;XPTRX->XHOLDXPTR;REPEAT XPTRX ($$GETX)
26 :     0->XPTRX;
27 :     PRINT '--- ' XHOLDXPTR ' pulled from stack.';
28 :     PRINT '      Use $$RECOVER_STACK to do an un-get'
29 :   )ELSE(PRINT '** NO PARMS ON STACK' )
```

Well, that's a mouthful! First, in line 24, we check for stack underflow (i.e., a negative value for XPTRX). Then we save a copy of XPTRX (just in case we want to cheat and get them again later). Notice that we use XPTRX as the value of the repeat in line 25, and the thing we repeat is itself another previously created command:

```
16 :   COMMAND $$GETX (IF ((IPTRI) <= (XPTRX)) THEN (
17 :     INPUT (STRTOSYM ('XTEMP'+(NUMTODECSTR IPTRI)));
18 :     (IPTRI+1)->IPTRI)
```

Here, you will notice that I used "IPTRI" instead of "XPTRX" just to make sure that things don't get trashed. Further, I check that there is no underflow by comparing IPTRI to XPTRX. One thing to note is that the IF statement does take a considerable time to execute. If I was going to do lots of this stuff, I probably would be better off taking it out and checking it only in the \$\$GET command.

Anyway, the \$\$GETX command is pretty straightforward, and it insures us of getting the lines in the correct order.

Lines 27 and 28 tell us what's going on. And, again, an important note is that if we close up the "THEN" clause on line 26 without opening the "ELSE (" , the SOS CI parser will fail to ever evaluate the ELSE clause:

```

OK:  IF (condition-of-some-sort) THEN (
      things to do if true;
      more things; and more things)ELSE(  <-----+
      things to do if false)              |
                                          |
BAD:  IF (condition-of-some-sort) THEN (
      things to do if true;
      more things; and more things)      |
      ELSE(things to do if false)        |
                                          |

In the second case, the ELSE clause will NEVER (EVER) |
be executed!!! The SOS strangeness strikes again.    |
So, programmers, leave those pending parentheses open |
ON THE SAME LINE.  -->-----+

```

Yes, yes, all well and good. But how do I use those commands??? |La questione eccellente!

```

>READ AUX_CMNDS      -- first we read the exec
>EDIT LAMB_FILE     -- edit our favorite file and
>INPUT              -- put some junk in it
INPUT MODE:
>Mary had a little lamb.
>Little lame lamb.
>Whose degree was in maths.
>>      -- hit an extra return to exit input mode
EDIT:
>TOP
>DOWN 1
>$$PUT 2
>END
>$$GET
--- 2 pulled from stack.
    Use $$RECOVER_STACK to do an un-get
>TOP;TYPE 999
TOF:
Mary had a little lamb.
Little lame lamb.
Whose degree was in maths.
Mary had a little lamb.
Little lame lamb.
EOF:
>
>END;INPUT '%%'     -- go to the end and enter a marker
>$$RECOVER_STACK   -- retrieve the holding pointer
--- Recovered 2 from stack
>$$GET
--- 2 pulled from stack.
    Use $$RECOVER_STACK to do an un-get
>TOP;TYPE 999      -- and let's see the file now...
TOF:
Mary had a little lamb.
Little lame lamb.
Whose degree was in maths.
Mary had a little lamb.
Little lame lamb.
%%
Mary had a little lamb.
Little lame lamb.
EOF

```

Pretty nifty, eh? By the way, you can put several commands on a line like that using the good old ";" between them: TOP;TYPE 3333.

## Fool-Proof Execs: Setting Up Your Execs As A System

Plan your work and work your plan. This really applies to SOS execs since there may be several different environments, etc. To handle this, it's usually a good idea to separate things. Specifically, it's a good idea to put command and variable definitions in a separate file from the actual execs. For example:

This is the first file, which really starts up the system. Notice that I READ the initialization procs to set up various parms.

```
% $FILE=XINIT_RAMSES (Start up RAMSES) RLT MAY-21-85
PRINT '---- RAMSES ----'
READ XRAM_CMND_INIT
% ----- THIS ROUTINE IS USED ON THE RAMSES TERMINAL
READ XRAM_HDW_INIT
READ XRAM_USRS_INIT
% ---- SYSTEM VARIABLES FOR THE RAMSES TERMINAL ITSELF
'<NULL>'->$NULL
'ACTIVE'->$ACTIVE
$ACTIVE->$RX_ALIVE
.....
PRINT '----- RAMSES is up.'
MSG ALL '----- RAMSES started.'
```

Let's look at a couple of the initialization procs:

```
% $FILE=XRAM_HDW_INIT RLT MAY-21-85
% THE HARDWARE VARS ARE STORED HERE FOR D250-COMB CONFIG
3->$RX_NPOOLS
%
'TM8 2'->$RX_XX1
'TM8 3'->$RX_YY1
%
'TM8 4'->$RX_XX2
'TM8 5'->$RX_YY2
%
'TM8 6'->$RX_XX3
'TM8 7'->$RX_YY3
%
.....

% $FILE=XRAM_CMND_INIT RLT JUN-1-85
COMMAND CM COMMAND
CM $$NTDS NUMTODECSTR
CM $$SP (SEND PREVIOUS)
CM $$SS (SEND SINK)
CM QUERY_PM (RECORD START ONTO PRT1;MAPCI NODISP;MTC;PM;
POST TM8 $RX_TM8;QUIT ALL;RECORD STOP ONTO PRT1)
.....
```

By planning your execs carefully, you can restrict where the changes need to be made to one specific file. For example, all commands are defined in the self-same file. However, there may be times when you need to create commands linked to a given sub-system (e.g., `EDIT`, `CRTSIM`, `DSKUT`, etc.). This means that defining a command *before* the sub-system is up will do you no good. To get around this, we use directories to create working environments. Read on...

### **Fool-Proof Execs: Using Directories**

When you create a directory, you are creating an artificial partition in your `SFDEV` memory. The neat thing about a directory is that when you create new commands, they automatically go into the `TOP-MOST` read/write directory. This means that you can override existing system commands or earlier definitions.

This does create some problems. For example, let's say that you have read the standard profile (`STD_PROF`) to define commands for the editor, and one of the commands is "D" for "DOWN". Then, when you `proc up` `PMIST` with the `PUP1` file, if there is a new read/write directory on top and it redefines "D" as "dump buffer", you can't get to your old "D" = "DOWN" command. You can, but you must specify the directory path; eg, `USERS.MAP3.D` instead of `DOWN`. Well, let's get on with it...

### **Fool-Proof Execs: Using Other People's Directories**

As we saw earlier, we can specify the directory path name. This makes it easy to use commands and things from other people's directories. You just fully qualify the directory path. Say, for instance, that you're `MAP1` and `MAP3` has some nifty doodle new command in his `UTILDIR` that he wants you to try out. You don't have to put the command in your own directory to try it out. You can just type:

```
>USERS.MAP3.UTILDIR.NIFTYCMD
```

Of course, if that command expects certain variables to exist and they exist in `MAP3`'s directory, `NIFTYCMD` won't work.

That does bring up another interesting point, though, because variables also exist in directories. So, if you wanted to know the contents of the variable `HISVAR` in user `MAP3`'s directory, just type:

```
>PRINT (USERS.MAP3.UTILDIR.HISVAR)
```

Neat, huh?

Of course, if you wanted easy access to everything in the directory, you could always pull this little trick:

```
>ATTACH USERS.MAP3.UTILDIR
```

Now, if all this has gotten you really jazzed about directories, you might want to go back to the `ATTACH` section and read about the `ATTACH` command. It gets into some of the neat stuff you can do with directories.

## Conclusions

Well, hopefully, you now know enough about SOS and CI and execs to do some reasonably productive things on the switch. I want to take a few lines here to recommend a few other documents in DIS which contain useful information. Some of these are referenced in this document, and some are not:

CPGUIDE	Call Processing Guide for the DMS-250
OSA10	SOS Loader Reference Manual
OSLC	COPY Command on NT40
W190	EXECUTE Command
DW114	Restarts
OSG	SOS Editor
DCML1	Log System - User View
OSNC	NCMS
N1001509	DMS-100 Family Command Reference Manual (NTP 297-1001-509)
S1001509	DMS-100 Family Supplement 10 of Commands Manual (NTP 297-1001-509)
N1001526	DMS-100 Family Disk Maintenance Subsystem Reference Manual (NTP 297-1001-526)
SLMUG	SuperNode System Load Module SLM User's Guide
N1001118	DMS-100 Family Magnetic Tape Reference Manual (NTP 297-1001-118)
JPF	DMS-100 MPCNET User's Guide
SYSDESC	DMS SuperNode System Description

---

# Vehicle Tracking Beacon

---

## Overview

Build a simple RF tracking beacon ("bumper beeper") using a slightly modified Family Radio Service (FRS) radio. This will use a 555-timer to pulse the radio's Push-to-Talk (PTT) key (to transmit for about 1.5 seconds) every 20 seconds or so. The transmitting beacon can then be tracked using normal RF direction finding techniques (doppler, rotating Yagis, body-beam, etc.). The pulsing control circuitry is small enough to fit inside the FRS radio, and can be powered from the radio's internal batteries. The pulsing control circuit is based around the 555-timer, and is the same circuit which was used for the **Mosque Time-Lapse Surveillance Camera** project in *GBPPR Zine*, Issue #12. Review that article for a more detailed explanation of the 555-timer operation and the equations needed for tweaking the transmit/wait times.

Attach powerful magnets to the radio so you can easily (and covertly) mount it to the underside of a vehicle. Hiding it *inside* the vehicle's dashboard is actually the best, as nobody will think to look in there and the RF can escape through the windshield. This does require one to physically enter the vehicle though.

## Uses

This device will be useful for tracking human rights abusers:



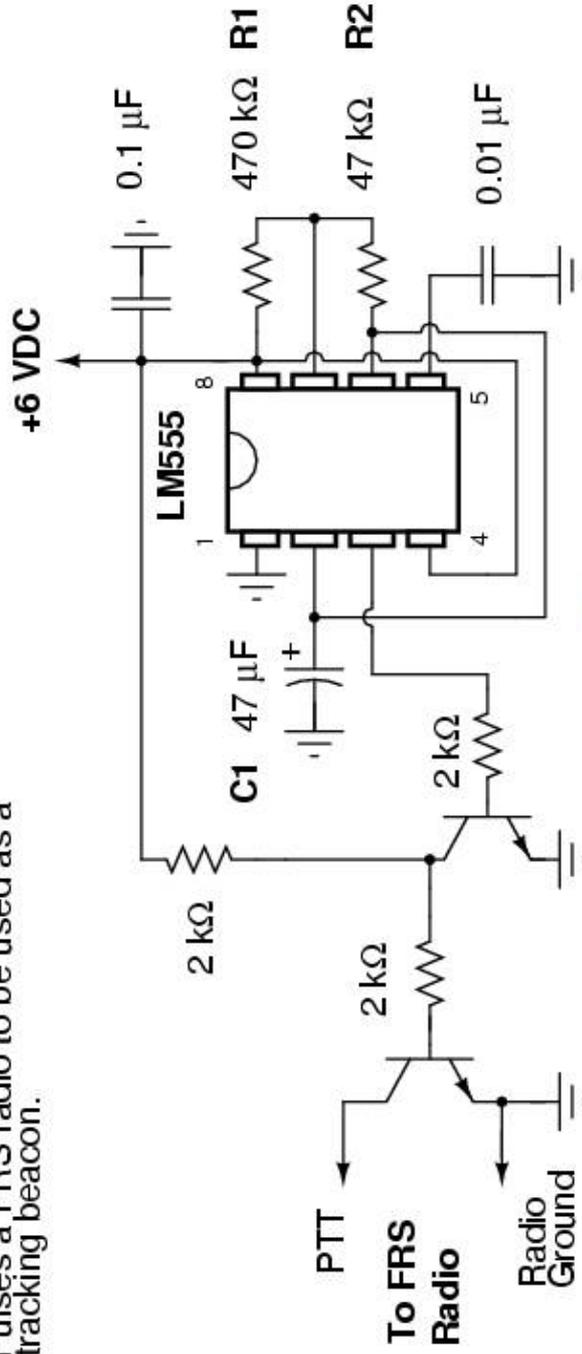
... or even for tracking terrorists:



Pulse Control Schematic

**Vehicle Tracking Beacon Control Circuitry**

Pulses a FRS radio to be used as a tracking beacon.



**Timer**

Approximate Wait Time = 17 seconds

Approximate Pulse Time = 1.5 seconds

**2N3904**

**Pulse Level Inverter**

**2N3904**

**PTT Control**

*Use a common ground between the pulse circuit and the radio.*



Internal view of the Cobra PR3500DX FRS/GMRS radio used. Just about any radio will probably work, but this particular radio has the "high-power" 2 Watt setting, which will be *very* useful for tracking beacon purposes. Note the large open space just below the battery compartment on the lower left.

The addition of an external antenna jack may also be useful. When mounted on the underside of a vehicle, vertical antenna polarization will be more effective.

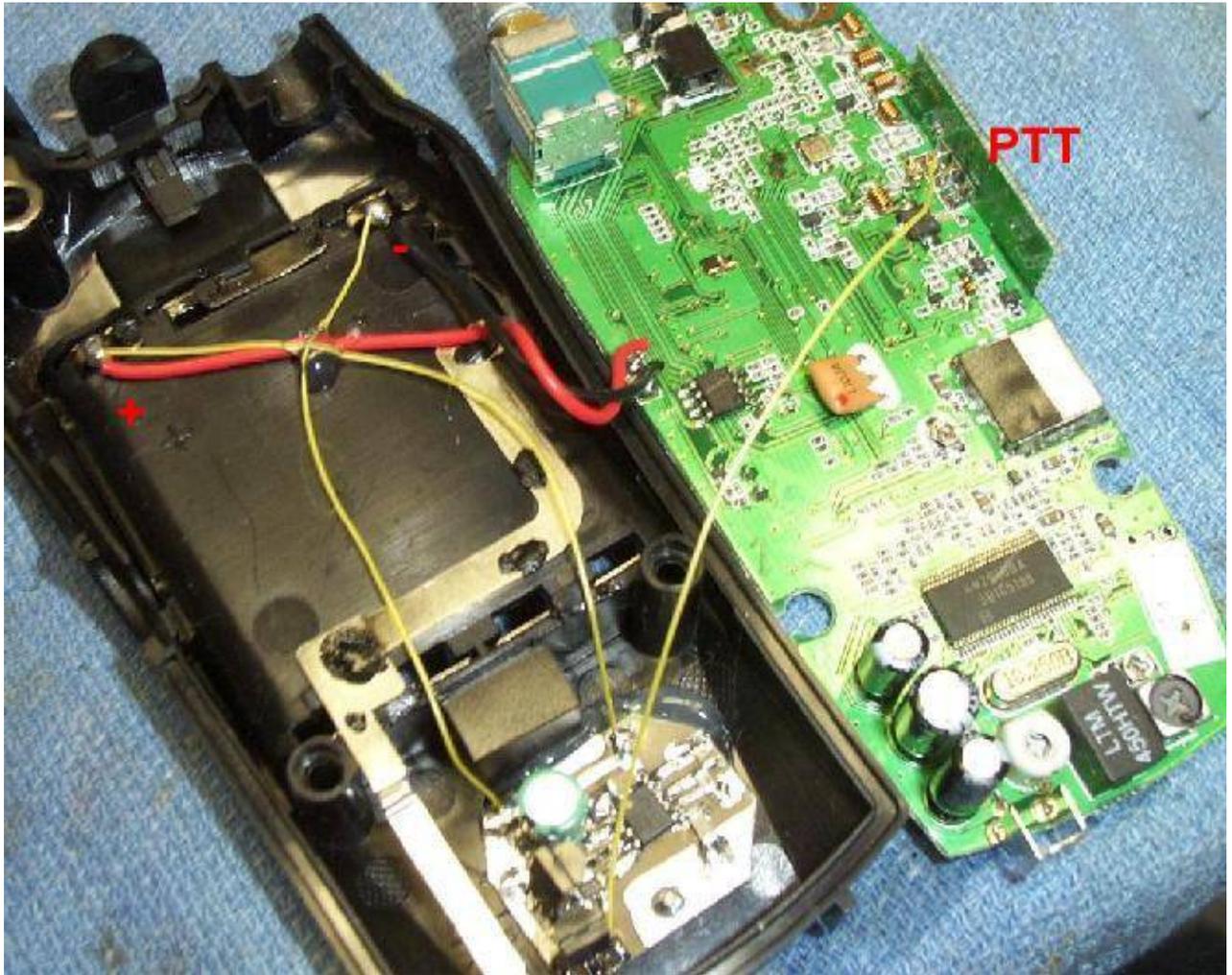


Picture showing the pulsing circuitry added. The circuit uses mostly surface mount components to keep it physically small. Only three wires are needed for the pulsing circuit: a wire running to the +6 VDC **POSITIVE** terminal on the battery compartment, a wire running to the **NEGATIVE** or **GROUND** terminal (which is common throughout the radio), and a wire running to the **PTT** switch. When the PTT switch is grounded, the radio will transmit.



Overview of the wiring connections.

Note that the vibrator motor and the speaker are removed to reduce the weight of the radio slightly. Leave the microphone in, as it will receive any ambient noise. If you hear road noise, the vehicle is moving. If you hear crickets chirping, the vehicle is parked somewhere. Pretty fucking clever, eh?



Another overview of the wiring connections. The **RED** wire is **POSITIVE**, the **BLACK** wire is **GROUND**. The wire for the **PTT** is connected to the top solder terminal on the little circuit board sticking up.



Closeup picture of the PTT connection. There are three solder terminals, solder to the *top* one (closest to the antenna).

# Outside Plant Symbols: General

**BELL SYSTEM PRACTICES**  
AT&TCo Standard

**SECTION 620-040-011**  
Issue 4, December 1980

## GENERAL OUTSIDE PLANT SYMBOLS

CONTENTS	PAGE
1. GENERAL .....	1
2. SYMBOLS FOR USE ON MAPS .....	1
3. CAUTION SYMBOL .....	4
4. REVISION SYMBOL .....	4

### 1. GENERAL

**1.01** The symbols included in this and succeeding sections through Section 620-040-019 are for use on construction work prints, plant location records, maps, etc, to indicate the various kinds of outside plant. The symbols in Part 2 of this section are primarily for use on maps but some may be used on work prints and records. The symbols in the balance of these sections are primarily for use on construction work prints, plant location records, etc, but some may also be employed on maps. A complete list of abbreviations used in conjunction with outside plant symbols is contained in Section 620-040-020.

**1.02** This section is being revised to add a symbol for revisions. Revision arrows are used to emphasize the more significant changes.

**1.03** The following symbols are employed in these sections to indicate whether the plant is existing, proposed, future, or to be removed:

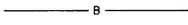
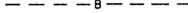
Existing (light line)	
Proposed (heavy line)	
Future	
To be removed	x x x

**1.04** In several cases, the same symbol is used to indicate two or more kinds of plant. However, distinction between types of plant so designated is clearly indicated by the associated symbols and details as well as by the title and general type of work print or record on which they are used.

### 2. SYMBOLS FOR USE ON MAPS

**2.01** In view of the difficulty of mechanically reproducing colors on maps, symbols which rely on the interpretation of color for their meaning are not recommended. Emphasis on certain features may be obtained by employing suitable comparative line weights, positioning and segregation, and providing suitable legends.

**2.02** The following symbols are for use on maps:

- |                                                          |                                                                                       |
|----------------------------------------------------------|---------------------------------------------------------------------------------------|
| (a) Underground conduit or underground cable in conduit. |  |
| (b) Future underground conduit.                          |  |
| (c) Cable — aerial, block, or building.                  |  |
| (d) Cable — submarine.                                   |  |
| (e) Cable — buried.                                      |  |
| (f) Future cable.                                        |  |
| (g) Wire — aerial (open or multiple).                    |  |
| (h) Wire — buried.                                       |  |

### NOTICE

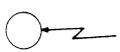
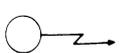
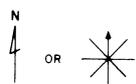
Not for use or disclosure outside the  
Bell System except under written agreement

Printed in U.S.A.

Page 1

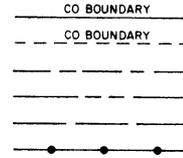
# Outside Plant Symbols: General

**SECTION 620-040-011**

- (i) Wire — drop. 
- (j) Bell company central office. 
- (k) Independent company central office. 
- (l) Future Bell company central office. 
- (m) Bell company central office toll center. 
- (n) Independent company central office toll center. 
- (o) Repeater station in central office. 
- (p) Repeater station — attended. 
- (q) Repeater station — unattended. 
- (r) Radio repeater station, including microwave — attended. 
- (s) Radio repeater station, including microwave — unattended. 
- (t) Radio telephone receiving station. 
- (u) Radio telephone transmitting station. 
- (v) Arrow for orientation (generally shown to the top or left on maps or records). 

(w) Boundary lines.

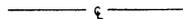
The following line symbols should be used to indicate boundary lines. They may be identified by legends on the map or notations adjacent to the symbol. The rule of showing heavy lines for proposed and light lines for existing plant (paragraph 1.03) does not apply for boundary lines.



(x) Railroad.



(y) Center line.

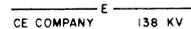


(z) Property line.



**2.03** When facilities other than those owned by telephone companies are shown on maps, the following codes should be used:

Aerial (electric shown)



Underground or buried (electric shown)



The following codes should be used to identify the type of facility:

- CATV Community antenna television
- E Electric
- G Gas
- M Municipally owned
- P Pipeline
- PO Privately owned

# Outside Plant Symbols: General

ISS 4, SECTION 620-040-011

- |    |                         |                                                                                                                                                                                    |
|----|-------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| R  | Railroad                | <b>2.04</b> When further identification is necessary, such as company name, state, county, or city, the symbol may be supplemented with suitable words, abbreviations, or legends. |
| S  | Sewer                   |                                                                                                                                                                                    |
| W  | Water                   |                                                                                                                                                                                    |
| WU | Western Union Telegraph | <b>2.05</b> Ownership of telephone plant is indicated as shown in the following examples:                                                                                          |

	OWNERSHIP	UG CABLE	BURIED CABLE	AERIAL CABLE	OPEN WIRE
(a)	Associated company.		- B - B -	_____	- - - - -
(b)	Associated company (toll).		- B - T - B -	_____ T _____	- - - - - T - - - - -
(c)	Other company (independent).		- B - I - B -	_____ I _____	- - - - - I - - - - -
(d)	AT&T Company (Long Lines).*		- B - A - B -	_____ A _____	- - - - - A - - - - -
(e)	Joint associated company and AT&T Company (Long Lines).*		- B - (A) - B -	_____ (A) _____ 	- - - - - (A) - - - - -
(f)	Associated company.†		- B - B - B -	_____ B _____	- - - - - B - - - - -
(g)	Joint associated company and other company.		- B - (W) - B -	_____ (W) _____	- - - - - (W) - - - - -
(h)	Telephone company exchange aerial cable on foreign pole line. Type of plant (open wire or cable) is indicated by the appropriate line symbol. Use of the plant (exchange or toll) is indicated by the appropriate notation.	W                  EXCH                  W _____			

\*For use on associated company maps.

†For use on AT&T Company maps (Long Lines).

# Outside Plant Symbols: General

## SECTION 620-040-011

**2.06** The conventional symbols employed on a topographic map of the United States may be used to designate state, county, township, city, village, and borough lines, as well as other details such as streams and trails when these designations are desirable.

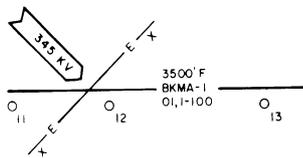
### 3. CAUTION SYMBOL

**3.01** This part introduces standard caution symbols to be used generally on construction work prints. These symbols may also be used on plant location records and maps where permanent potential hazards or special conditions exist. The symbols are to be used at all locations on construction work prints where it is appropriate to get the special attention of construction, maintenance, and engineering personnel. It will act as a signal for indicating the need for extra caution and should be accompanied by appropriate notes explaining the hazardous conditions. All notes, whether on construction work prints or plant location records, should be placed in a prominent place on the drawings.

- (a) Caution symbol.  
(Refer to note number on work print if applicable.)



- (b) Illustration of caution symbol used on construction work print.



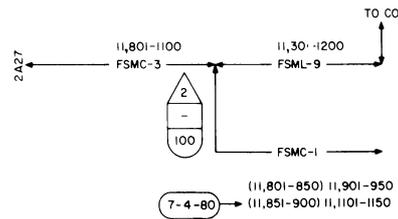
### 4. REVISION SYMBOL

**4.01** This part introduces the standard revision symbol to be used on construction work prints. This symbol is used at each location being revised on a construction work print to alert construction, maintenance, and engineering personnel. The symbol should be placed in a prominent place near the location being revised and should contain appropriate notes explaining the revision.

- (a) Revision symbol.  
Refer to the date revision is made and to appropriate notes, if applicable.



- (b) Illustration of symbol used on construction work print.



# Outside Plant Symbols: Poles & Associated Equipment

**BELL SYSTEM PRACTICES**  
Plant Series

**SECTION 620-040-012**  
Issue 3, December 1969  
AT&TCo Standard

## OUTSIDE PLANT SYMBOLS POLES AND ASSOCIATED EQUIPMENT

CONTENTS	PAGE
1. GENERAL . . . . .	1
2. CODES . . . . .	1
3. SYMBOLS . . . . .	2

### 1. GENERAL

**1.01** The identification codes and symbols listed in this section pertain to poles and associated equipment. They are for use primarily on construction work prints and plant location records, although some of them may be used on maps. A complete list of abbreviations used in conjunction with outside plant symbols is contained in Section 620-040-020.

**1.02** This section is reissued to add and revise codes and symbols. Since this is a general revision, arrows ordinarily used to indicate changes have been omitted.

**1.03** Some of the codes contained in this section have been revised to be compatible with the various accounting machines which produce tabular pole records from punched cards and tapes. The explanation of these codes is also found on the tabular pole record form under the column headed "Codes."

### 2. CODES

**2.01** The following codes are to be used in connection with poles:

- (a) Pole class designations, from 1 through 10.

*Example:* 35'-5 indicates a 35-foot, class 5 pole.

- (b) Species and preservatives (treatments). Poles supplied under standard specifications have a letter code included in the brand, which identifies the type of timber and treat-

ment. The following are codes for the standard species and certain nonstandard poles which may be used occasionally:

SPECIES	CODE
Chestnut	CH
Cypress	CY
Douglas fir*	DF
Eastern cedar	EC
Jack pine*	JP
Juniper	JU
Lodgepole pine*	LP
Northern pine	NP
Ponderosa pine (western)	WP
Red pine*	RP
Redwood	RW
Sawed redwood	SRW
Southern pine*	SP
Split cedar	SC
Western hemlock	WH
Western larch (tamarack)*	WL
Western red cedar*	WC

\* Standard species.

PRESERVATIVE	CODE
Criton	G
Creosote	C
Creosote-pentachlorophenol	A
Double treatment (salt and creosote)	D
Copper naphthenate	N
Pentachlorophenol	P
Chemonite or greensalt	S
Water-gas-tar (creosote mixture)	W
50-50 mixture of creosote and petroleum	X

# Outside Plant Symbols: Poles & Associated Equipment

**SECTION 620-040-012**

**2.02** The following is an example of a standard brand which is located 10 feet from the butt on poles 50 feet or less and 14 feet from the butt on poles 55 feet or longer.

Supplier's code or trade-mark.	
Plant location and year of treatment, as Old-town-1967.	0-67
Species and preservative, as southern pine-creosote-penta solution.	SPA
Size, as class 5, 35-foot pole.	5-35
If desired, size may be shown thus.	5 35

	12 kV or less (Cont)	Over 12 kV (Cont)
(1) Rental or joint-use contract	— ×	— *
(2) Free attachment rights	— × —	— * —
(d) Bell and electric company jointly owned pole		
(1) Existing (indicated by light circle and light symbol)	⊗	⊗
(2) Proposed or replaced		

**3. SYMBOLS**

**3.01** The following symbols are to be used in connection with poles:

(a) Proposed Bell company pole	●
(b) Existing Bell company pole	○
(1) Rental or joint-use contract	— ○
(2) Free attachment rights	— ○

**SUPPORTING  
PHASE-TO-PHASE  
VOLTAGES (See Note)**

	12 kV or less	Over 12 kV
(c) Existing electric company pole	×	*

- To be set by Bell company (indicated by heavy circle and light symbol)
- To be set by electric company (indicated by light circle and heavy symbol)

**Note:** The symbols represent only two categories of voltages supported by the pole. The engineer must specify by adequate notations on the construction work prints the exact voltages involved. This is required so the construction forces can provide proper vertical clearances and take the necessary safety precautions when working in the area. See Section 620-040-011 for use of the caution symbol.

# Outside Plant Symbols: Poles & Associated Equipment

ISS 3, SECTION 620-040-012

(e) Ownership

	JOINTLY BY BELL COMPANY AND ANOTHER COMPANY	SOLELY BY ANOTHER COMPANY
(1) AT&T Company pole (Long Lines). *	Ⓐ	A
(2) Associated company pole. †	Ⓑ	B
(3) Independent company pole.	Ⓘ	I
(4) Western Union Telegraph Company pole.	Ⓜ	W
(5) Municipal pole.	Ⓜ	M
(6) Street railway pole.	Ⓣ	T
(7) Railroad pole.	Ⓡ	R
(8) Community antenna television company pole.	Ⓥ	V
(9) Pole owned by customer.		C
(10) Pole owned by multiparties. Number indicates number of joint owners.	③	

\* Shown on associated company drawings.

† Shown on AT&T Company drawings (Long Lines).

**3.02** The following are illustrations of poles:

(a) Proposed Bell company pole. Length and class indicated. \*  30'-5

(b) Proposed Bell company pole. Length, class, and number indicated. \*

 P432  
30'-5

(c) Existing Bell company pole number 432 to be replaced. Data (length of pole and date of placement) above line is for retirement purposes. Data below line indicates length and class of proposed pole. \*

 P432  $\frac{(30' 26)}{35'-5}$

(d) Existing Bell company pole to be removed. Number, length, and date of placement indicated. \*

 P432 (30' 26)

(e) Existing Bell company pole to be pulled, cut off, and reset.

 P432  $\frac{(35'-5)}{30'-5}$  PRST

(f) Existing Bell company pole to be top cut.

 P432  $\frac{(35'-5)}{30'-5}$  TOP CUT

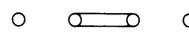
(g) Existing Bell company pole to be moved. Number indicates distance.

 3'

(h) Bell company H fixture across line.



(i) Bell company H fixture along line.



(j) Bell company A fixture across line.



(k) If pole material is other than wood, indicate by appropriate notation.

 STEEL

\* Species and preservative information is not required for poles being removed. The placing forces will provide this information for poles being placed.

# Outside Plant Symbols: Poles & Associated Equipment

## SECTION 620-040-012

**3.03** The following symbols are to be used in connection with anchors and guys:

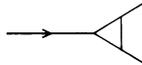
(a) Anchor only



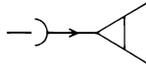
(b) Guy only



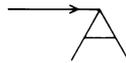
(c) Anchor and guy



(d) Anchor and insulated guy



(e) Sidewalk anchor and guy



(f) Tree guy



(g) Rock anchor and guy



(h) Push brace



(i) Push and pull brace



(j) Anchor and guy wholly owned by another company



(k) Jointly owned anchor



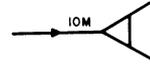
(l) Jointly owned anchor and Bell company guy



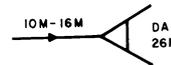
(m) Jointly used guy



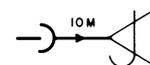
(a) One 10M guy and anchor.



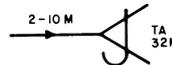
(b) One 10M and one 16M guy proposed on an anchor equipped with a 26M double thimble eye rod.



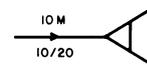
(c) One Bell company 10M insulated guy attached to an existing jointly owned anchor.



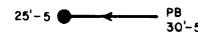
(d) Two proposed Bell company 10M guys attached to a proposed jointly owned anchor. The proposed anchor is 32M, equipped with triple thimble eye rod to accommodate guying requirements of the other company. (If anchor is other than standard for specified guy, this size must be shown on work print.) See Note.



(e) One 10M guy and anchor. Lead of 10 feet and height of 20 feet. If lead-height ratio is other than 1/1, it must be shown on the construction work print.



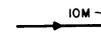
(f) One proposed Bell company pole with push brace.



(g) One 6M tree guy.



(h) One Bell company 10M guy attached to existing other company owned anchor.



**Note:** The number of guys should be shown only when more than one is being placed or removed.

**3.04** The following are illustrations of anchors and guys:

# Outside Plant Symbols: Wire

BELL SYSTEM PRACTICES  
Plant Series

SECTION 620-040-013  
Issue 4, December 1969  
AT&TCo Standard

## OUTSIDE PLANT SYMBOLS

### WIRE

CONTENTS	PAGE	
1. GENERAL . . . . .	1	H High strength
2. SYMBOLS FOR WIRE . . . . .	1	F Extra high strength
3. SYMBOLS FOR WIRE TERMINALS OR PROTECTORS . . . . .	2	D-TW D tree wire (copper steel)
4. SYMBOLS FOR WIRE LOADING AND BRIDGE LIFTERS . . . . .	3	RC River crossing strand
5. OPEN-WIRE SYMBOLS FOR MISCELLANEOUS EQUIPMENT AND USAGE . . . . .	3	2.02 The following are illustrations of open wire:
6. USE OF SYMBOLS IN CIRCUIT DIAGRAMS AND TOLL WIRE RECORDS. . . . .	4	(a) Ten 104 copper line wires. (The 10 indicates the num- ber of wires.)
1. GENERAL		----- 10-104 -----
1.01 The identification codes and symbols listed in this section pertain to wire plant and associated items. They are for use primarily on construction work prints and records, although some of them may be used on maps. A complete list of abbreviations used in conjunction with outside plant symbols is contained in Section 620-040-020.		(b) Six 109 steel line wires.
1.02 This section is reissued to add and revise codes and symbols. Since this is a general revision, arrows ordinarily used to indicate changes have been omitted.		----- 6-109 -----
2. SYMBOLS FOR WIRE		(c) Ten 128 copper- steel line wires having 30 percent con- ductivity.
2.01 <i>Open wire</i> should be illustrated by a 3- digit number to indicate the diameter of the wire in mils. If the type of wire is not ap- parent from this number, the following supple- mental codes should be used for identification:		--- 10-128CS-30 ---
C Copper		(d) Ten 109 high strength steel line wires.
CS Copper steel		----- 10-109H -----
S Steel		(e) Two 109 extra high strength steel line wires.
		----- 2-109F -----
		(f) Four 080 D-type copper-steel tree wires.
		----- 4-080D-TW -----
		(g) Six 165 steel river crossing strand.
		----- 6-165S-RC -----
		(h) Method of indi- cating a change in the type or num- ber of wires.
		--- 4-109H ->- 2-109F ---
		<i>Note:</i> The number of wires precedes the size and type of wire.

# Outside Plant Symbols: Wire

## SECTION 620-040-013

**2.03 Multiple line or drop wire** supplied by the Western Electric Company is to be coded on construction work prints as follows:

- (a) **First Letter:** Sequence of standardization.
- (b) **Subsequent two or three letters:** Type of multiple line or drop wire, that is:

DRW D rural wire  
 CRW C rural wire  
 CMDW C multiple drop wire

- (c) **Number of Pairs**

**Note:** If more than one multiple line wire is to be indicated, the number of multiple line support wires should precede the code letters of the wire.

**2.04** The following are illustrations of multiple line or drop wire:

- (a) Two 6-pair D rural wires ---- 2 DRW - 6 ----
- (b) A 12-pair E rural wire ---- ERW-12 ----
- (c) A 1-pair C rural wire ---- CRW-1 ----
- (d) A 6-pair C multiple drop wire ~~~~~ CMDW-6 ~~~~~

**Note:** The manufacture of urban multiple line wire has been discontinued. Multiple line wire not supplied by Western Electric Company should be coded as MLW (multiple line wire), followed by number of pairs and gauge. For example,

A 12-pair, 22-gauge multiple line wire -- MLW-12 PR 22 --

**2.05** Block or drop wire is shown as follows:

Three drop or block wires ~~~~~ 3 ~~~~~

**2.06** Buried wire should be shown by type of wire, followed by the number of pairs in the wire. If more than one buried wire is to be indicated, the designation is preceded by the number of such wires. The B in the wire symbol line indicates buried wire.

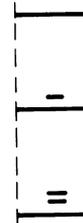
**2.07** The following are illustrations of buried wire:

- (a) Two buried 1-pair D underground wires -- B --- 2-D1 --- B --
- (b) Two buried 2-pair B service wires -- B --- 2-B2 --- B --
- (c) Slack loop placed in buried wire -- B --- (e) --- B --

## 3. SYMBOLS FOR WIRE TERMINALS OR PROTECTORS

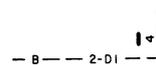
**3.01** The following basic symbols are used for wire terminals or protectors:

- (a) Nonprotected wire terminal
- (b) Wire terminal or protector equipped with cable protection units
- (c) Wire terminal or protector equipped with station protection units



**3.02** The following are illustrations of wire terminals and protectors:

- (a) Two buried D underground wires terminated in a D buried wire terminal.



- (b) A 116-type protector used as a protected wire terminal on 6-pair D rural wire.



# Outside Plant Symbols: Wire

ISS 4, SECTION 620-040-013

(c) A 108-type wire terminal equipped with cable protection terminating a 12-pair E rural wire. (The capacity of the terminal is 16 pairs.)

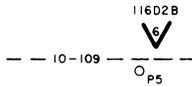


(d) A 105-type 1-pair nonprotected wire terminal.



**Note:** The pair capacity and the type of terminal or protector are indicated on construction work prints.

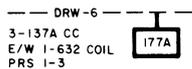
**3.03** The symbol for indicating auxiliary protection on open-wire circuits. The number of protector units (6) and type of protective equipment (116D2B) are indicated. (The protector is located at pole number 5.)



## 4. SYMBOLS FOR WIRE LOADING AND BRIDGE LIFTERS

**4.01** The following symbols are used in connection with wire loading and bridge lifters:

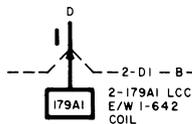
(a) Coil case on multiple line wire. Illustrated is a 177A coil case containing three 137A coil cases equipped with 632 load coils. (Pairs 1 through 3 are loaded.)



(b) A 178A1 coil case containing one 632 loading coil on C rural wire.



(c) 179-type coil cases installed in a D buried wire terminal. (The arrow indicates that cases are located in the terminal.)



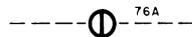
(d) Bridge lifters on multiple line wire. Illustrated is a 177A coil case containing two 137B coil cases equipped with 1574A inductors.



## 5. OPEN-WIRE SYMBOLS FOR MISCELLANEOUS EQUIPMENT AND USAGE

**5.01** The following symbols represent miscellaneous equipment associated with open wire, along with the usage of open-wire circuits:

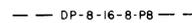
(a) The presence of a repeating coil in a branch circuit is shown by the use of the coil-type designation beside the symbol that indicates the branch circuit.



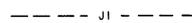
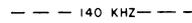
(b) Filter of any type. Additional data should be furnished in a supplementary note when necessary.



(c) Method of indicating open-wire circuit specifications. Type of insulator (DP), wire spacing on the crossarms (8-16-8), and 8-inch point transposition brackets (P8) are shown.



(d) Open-wire pair transposed for carrier operation. Numeral indicates the top of the nominal frequency range (140 kilohertz). The actual type of carrier transposition system (J1) may be indicated if desired.



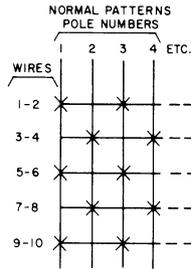
# Outside Plant Symbols: Wire

**SECTION 620-040-013**

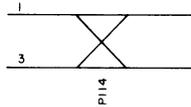
**6. USE OF SYMBOLS IN CIRCUIT DIAGRAMS AND TOLL WIRE RECORDS**

6.01 The following illustrate the use of symbols in connection with open-wire circuit diagrams and toll open-wire records:

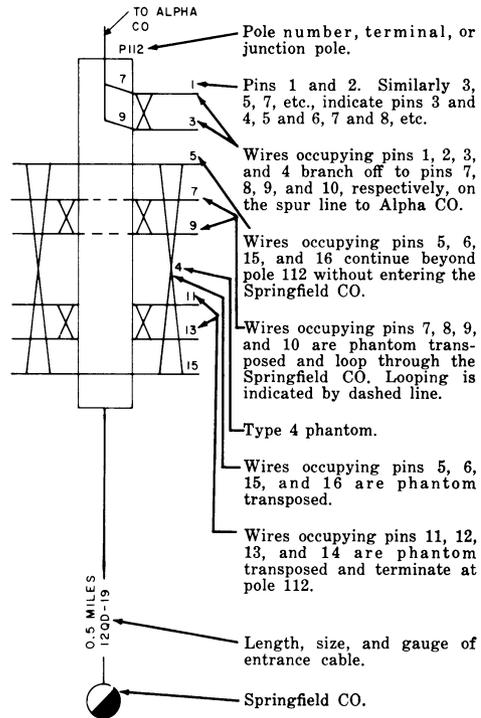
(a) The crossed lines ( $\times$ ) on this symbol indicate that the circuit is transposed, e.g., wires 1 and 2 are transposed at poles 1 and 3, etc.



(b) The crossed lines ( $\times$ ) of this symbol indicate that two circuits are phantom transposed, e.g., the circuit on wires 1 and 2 is phantom transposed with the circuit on wires 3 and 4 at pole 114.



(c) Typical circuit diagram.



# Outside Plant Symbols: Underground Conduit & Manholes, Building Conduit & Housings

BELL SYSTEM PRACTICES  
Plant Series

SECTION 620-040-014  
Issue 3, December 1969  
AT&TCo Standard

## OUTSIDE PLANT SYMBOLS UNDERGROUND CONDUIT AND MANHOLES BUILDING CONDUIT AND HOUSINGS

	CONTENTS	PAGE
1. GENERAL . . . . .		1
2. SYMBOLS FOR UNDERGROUND CONDUIT AND MANHOLES . . . . .		1
3. SYMBOLS FOR BUILDING CONDUIT AND HOUSINGS . . . . .		3

### 1. GENERAL

**1.01** The symbols listed in this section pertain to underground conduit and manholes and to building conduit and housings. These symbols are for use primarily on construction work prints and records, although some of them may be used on maps. A complete list of abbreviations used in conjunction with outside plant symbols is contained in Section 620-040-020.

**1.02** This section is reissued to add and revise symbols. Since this is a general revision, arrows ordinarily used to indicate changes have been omitted.

### 2. SYMBOLS FOR UNDERGROUND CONDUIT AND MANHOLES

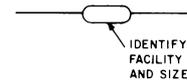
**2.01** The following symbols are to be used in connection with underground conduit and manholes:

- (a) Underground conduit
- (b) Manhole
- (c) Underground conduit, manhole, and subsidiary conduit

(d) Splicing chamber, service box, or handhole



(e) Subsurface structure and/or facility of other utility company

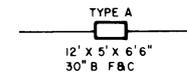


The following codes should be used to identify the type of facility:

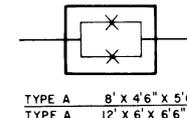
- CATV Community antenna television
- E Electric
- G Gas
- M Municipally owned
- P Pipe line
- PO Privately owned
- S Sewer
- Water

**2.02** The following are illustrations of manholes:

(a) Proposed manhole. Type, length, width, headroom, and type of frame and cover indicated.



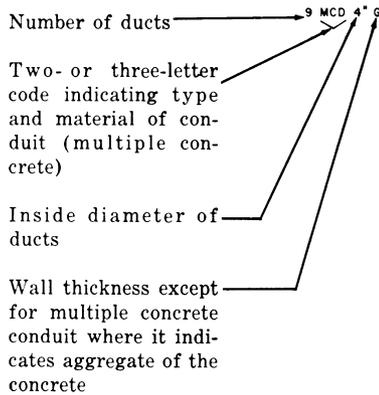
(b) Existing manhole to be rebuilt. Length, width and headroom specified.



**2.03** The method of indicating conduit is as follows:

# Outside Plant Symbols: Underground Conduit & Manholes, Building Conduit & Housings

**SECTION 620-040-014**



**2.04** The following are illustrations of conduit:

- (a) Plastic conduit, type B (thin wall) ——— 9 PC 4" B ———
- (b) Plastic conduit, type C (heavy wall) ——— 6 PC 4" C ———
- (c) Plastic conduit, type D (for exposed locations) ——— 4 PC 4" D ———
- (d) Fiber conduit, type B (thin wall) ——— 8 FD 4" B ———
- (e) Fiber-cement conduit, type B (thin wall) ——— 6 CEM B ———
- (f) Multiple concrete conduit, type F (lightweight aggregate) ——— 9 MCD 4" F ———
- (g) Multiple tile conduit ——— 6 MTD ———
- (h) Single tile conduit ——— 2 STD ———
- (i) Sewer pipe conduit ——— 1 SP 4" ———

- (j) Steel pipe conduit ——— 1 STL 4" ———
- (k) Creosote wood conduit ——— 2 CWD 4-1/2" ———
- (l) Method of indicating trench feet of conduit (wall-to-wall measurement) type of conduit (multiple tile, 6 ducts), and year of placement
- (m) Standard and oversize multiple tile conduit ——— 8 MTD 4-1/2" / 4 MTD 4" ———
- (n) One cast iron bend, type 3L
- (o) One sewer pipe bend, 4-inch type 3-1/2 D

**2.05** The following are miscellaneous symbols associated with underground conduit construction work prints:

- (a) Catch basin at curb.
- (b) Water line. Valve in line. Valve off main line. Fire hydrant.
- (c) Right-of-way and bench marker.
- (d) Property line stake or iron pipe.
- (e) Traffic light signal post.
- (f) Traffic light controls, pedestal mounted.

# Outside Plant Symbols: Underground Conduit & Manholes, Building Conduit & Housings

ISS 3, SECTION 620-040-014

(g) Traffic light controls, underground.



(h) Sign post, street, road, stop, etc.



(i) Electric transformer.



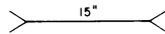
(j) Mail box.



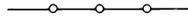
(k) Parking meter.



(l) Culvert (size indicated).



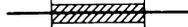
(m) Fence.



(n) Tree or bushes (trunk diameter indicated).



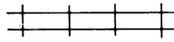
(o) Section of pavement, driveway, or sidewalk to be replaced.



(p) Push and remove pipe, tunnel, auger, or place casing.



(q) Surface railroad tracks.



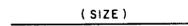
(r) Amount of ground cover.



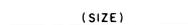
### 3. SYMBOLS FOR BUILDING CONDUIT AND HOUSINGS

**3.01** The following symbols are to be used in connection with building conduit and housings:

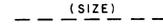
(a) Conduit, concealed in ceiling or wall.



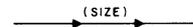
(b) Conduit, concealed in floor.



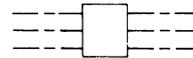
(c) Conduit, exposed.



(d) Conduit, home run to panel board.



(e) Underfloor duct and junction box, triple system. Number of lines entering a box in the header duct run indicates number of systems (telephone, electric, TV, etc.).



(f) Conduit riser.



(g) Riser sleeve.



(h) Floor outlet.



(i) Ceiling outlet.



(j) Wall outlet box, telephone.



(k) Primary entrance location.



(l) Primary entrance outlet.



(m) Conduit for placing ground wire.



(n) Backboard.



(o) Metal wall cabinet.



(p) Telephone panel (circuit).



(q) Switchboard.



(r) Pull box.



---

# *Outside Plant Symbols: Underground Conduit & Manholes, Building Conduit & Housings*

---

**SECTION 620-040-014**

(s) Cellular floor  
with trench head-  
er feed.



(t) Cellular floor  
with header duct  
feed.



# Outside Plant Symbols: Cable, Cable Terminals, Closures, & Interfaces

WISCONSIN BELL PRACTICES  
Wis. Bell, Inc.

ADDENDUM 620-040-015 WB  
Issue A, March 1985

## OUTSIDE PLANT CODES AND SYMBOLS USE AND APPLICATION

### CABLE, CABLE TERMINALS, CLOSURES, AND INTERFACES

#### 1. GENERAL

1.001 This addendum supplements Section 620-040-015, Issue 8. Place this pink sheet ahead of Page 1 of the section. Do not remove this addendum when a new Section is issued.

1.002 This addendum is issued to replace the information on Lightguide Cable and Equipment with Wisconsin Bell standards.

#### 2. CHANGES TO SECTION

2.001 Fig. 7 through 11 are obsolete.

New Fig. 7 through 10 are shown in this addendum.

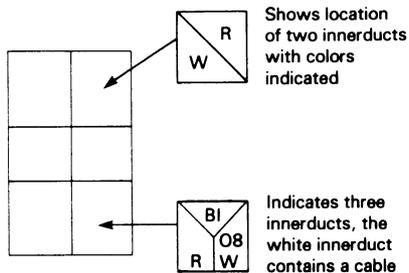


Fig. 7 - Innerduct Symbols

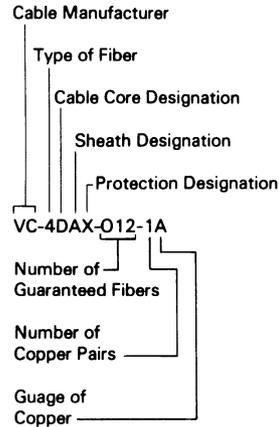


Fig. 8 - Cable Classification

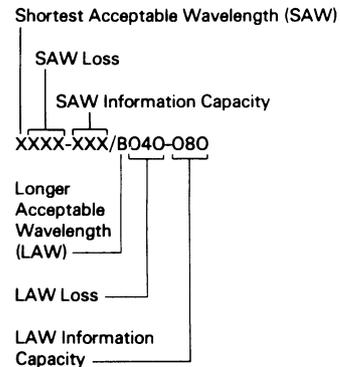


Fig. 9 - Transmission Characteristics

**NOTICE**  
Not for use or disclosure outside the  
Wisconsin Bell, Inc. except under written agreement

Printed in U.S.A.

Page 1

# Outside Plant Symbols: Cable, Cable Terminals, Closures, & Interfaces

Addendum 620-040-015 WB

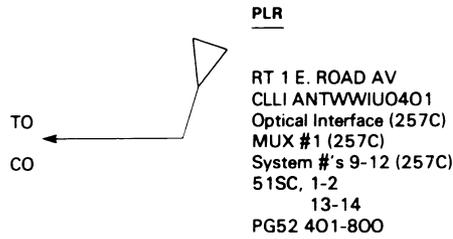
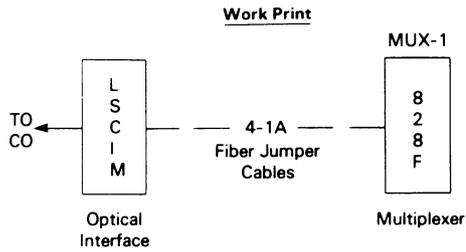


Fig. 10 Remote Terminal Equipment

**NOTICE**  
Not for use or disclosure outside the  
Wisconsin Bell, Inc except under written agreement

Printed in U.S.A.

# Outside Plant Codes & Symbols: Cable, Cable Terminals, Closures, & Interfaces

BELL SYSTEM PRACTICES  
AT&TCo Standard

SECTION 620-040-015  
Issue 8, September 1983

## OUTSIDE PLANT CODES AND SYMBOLS USE AND APPLICATION CABLE, CABLE TERMINALS, CLOSURES, AND INTERFACES

CONTENTS	PAGE	CONTENTS	PAGE
1. GENERAL . . . . .	2	H. Lightguide Cable and Equipment . . . . .	16
2. TERMS, ABBREVIATIONS, AND SYMBOLS . . . . .	2	4. CABLE CODE DESIGNATIONS . . . . .	19
A. Cable . . . . .	2	A. Cable With Code Designation BHBA-2-BT . . . . .	19
B. Modular Splicing Systems . . . . .	4	B. Types of Outer Sheath Protective Coverings . . . . .	19
C. Special Symbols—CONECS . . . . .	4	C. Exchange-Type Cables . . . . .	20
D. Grounding and Bonding (Shown on Work Prints Only) . . . . .	5	D. Special Cables . . . . .	20
E. Taper Codes . . . . .	6	E. Composite Cables Containing Exchange Pairs . . . . .	20
F. Line and Splicing Symbols . . . . .	6	F. Cables Designated by Drawing Number . . . . .	20
G. Cable Terminals, Closures, and Inter- faces . . . . .	7	G. Other Cables Not Coded or Specified by Cable Drawing Number . . . . .	21
3. EXAMPLES OF APPLICATION . . . . .	9	H. Conversion From Obsolete Numerical Coding to Letter Coding . . . . .	21
A. Strand-, Pole-, or Wall-Mounted Distri- bution Terminals . . . . .	9		
B. Distribution Terminals—Buried Plant . . . . .	10	<b>Figures</b>	
C. Feeder-Distribution Interfaces (FDIs) . . . . .	11	1. Example of Feeder-Distribution Interface . . . . .	11
D. Building Terminals . . . . .	13	2. Example of Rural Area Interface . . . . .	12
E. Main Distributing Frame (MDF)—Cen- tral Office or PBX . . . . .	14	3. Example of Terminating Cable on an MDF Using Stubs . . . . .	14
F. Taper Code . . . . .	15	4. Example of Terminating Cable on MDF Using Protectors . . . . .	15
G. Optional Method of Recording Terminal Address . . . . .	16	5. Example of Taper Coding . . . . .	16

**NOTICE**

Not for use or disclosure outside the  
Bell System except under written agreement

Printed in U.S.A.

Page 1

# Outside Plant Codes & Symbols: Cable, Cable Terminals, Closures, & Interfaces

**SECTION 620-040-015**

CONTENTS	PAGE
6. <b>Optional Method of Recording Terminal Address</b> . . . . .	16
7. <b>Lightguide Cable and Equipment</b> . . . . .	17
8. <b>Duct Liner Symbols</b> . . . . .	17
9. <b>Lightguide Cable Classification</b> . . . . .	18
10. <b>Lightguide CO Equipment</b> . . . . .	18
11. <b>Remote Terminal Equipment</b> . . . . .	19

**1. GENERAL**

**1.01** The identification codes and symbols in this section pertain to cable, cable terminals, closures, and interfaces. Primarily, they are for use on construction work prints and plant location records, although some may be used on maps. The symbols are basic in nature and may be used in combination with other terms, symbols, or notes to portray any of the various equipment or installation configurations existing or forthcoming. Typical examples of the application portion of this section show how some combinations may be indicated. The abbreviations used in conjunction with outside plant symbols are contained in the Common Language Standard Abbreviations, Sections 751-410-101 and -102. A description of the common language standard abbreviations and an index of abbreviation subset Bell System Practices (BSPs) are included in Section 751-410-100.

**1.02** The reasons for reissuing this section are listed below.

- (a) Adds lightguide cable designations, symbols, and cable coding
- (b) Adds symbol to identify line work operations by job step to be used in Distribution Service Design Centers (DSDCs) which have the Job Management Operations System (JMOS).

In addition to the above specific changes, this section has been completely reorganized and is considered a general revision. As a result, revision arrows are not used.

**2. TERMS, ABBREVIATIONS, AND SYMBOLS**

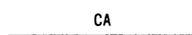
**2.01** The following are the basic graphic or letter symbols and abbreviations to be used in design-

ating the various components of cable, cable terminals, closures, and interfaces. Except where shown, the use is common to both work prints and records.

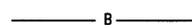
**A. Cable**

**2.02** The various symbols to be used with the cable line are as follows:

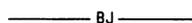
- (a) Aerial, building, or underground.



- (b) Buried.



- (c) Buried in joint trench. Buried joint (BJ) indicates buried cable plant installed in a trench used jointly with one or more utility companies. This applies to either random or normal separation between the telephone cable and the facilities of the other companies. Notes will be required to indicate the ownership and type of facilities using the joint trench. This symbol will also apply to situations where power and telephone cables are in separate trenches, but treatment for bonding and power separation is the same as for joint trench.

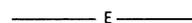


- (d) Submarine cable.



- (e) Cables other than those owned by telephone companies. The codes used to identify the type or ownership of the cables are listed in Section 620-040-011.

- Aerial (electric code shown)



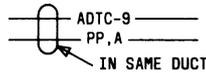
- Underground or buried (electric code shown).



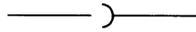
# Outside Plant Codes & Symbols: Cable, Cable Terminals, Closures, & Interfaces

ISS 8, SECTION 620-040-015

(f) An oblong shaped symbol is used to depict a gas feeder pipe in the same duct with underground cable. This symbol may also be used to show that two aerial cables are lashed to the same strand or that two cables are in the same duct.



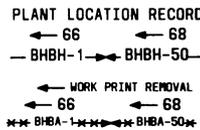
(g) Insulating joint.



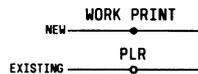
(h) Arrows in both directions indicate change in cable characteristics, mortality data, or count change (size, gauge, count, type of sheath, conductor insulation, sequence of standardization, year of placement).



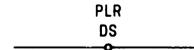
(i) Method of indicating year of placement. (Arrow points toward central office [CO].) Always shown on plant location records. Shown on work prints when cable is removed from continuing property record (CPR).



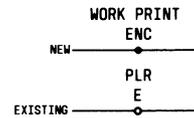
(j) Method of indicating splice. No change in cable characteristics, mortality data, or count. Solid dot for new and open dot for existing.



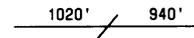
(k) Method of indicating an existing duct splice (DS) in an underground cable. This includes all types of cables.



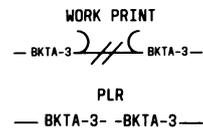
(l) Splice encapsulated.



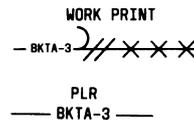
(m) Point on cable (other than a splice) where a division of measurements or point of record is required, eg, tax district boundaries, accounting classification, or structural design.



(n) Cable to be cut, pair ends cleared, and cables capped.



(o) Portion of cable to be removed. Remaining cable end cleared and capped.



# Outside Plant Codes & Symbols: Cable, Cable Terminals, Closures, & Interfaces

**SECTION 620-040-015**

(p) Cable pairs cut and ends cleared in sleeve.



(q) Clear and preconnect cable pairs.



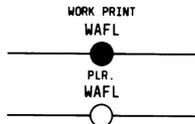
(r) Cap placed on proposed cable.



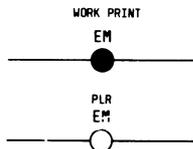
(s) Cable looped through terminal, interface, etc.



(t) Water alarm and fault location.



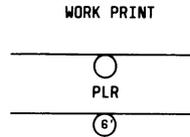
(u) Electronic marker-tuned circuit encapsulated in splice as an aid in locating cable.



(v) Buried cable, wire, or manhole marker. Number indicates the marker number. Arrow points toward cable.



(w) Loop in aerial, building, buried, or underground cable. No splice involved or sheath removed. The 6' represents length of loop in feet.

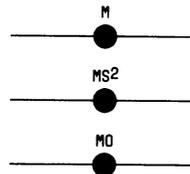


**B. Modular Splicing Systems**

**2.03** Depending upon the manufacturer, the modular splice symbols are as follows:

- (a) Bell System modular
- (b) 3M modular
- (c) Other modular.

**Note:** The letter symbols are generally for work print use only. Modular symbols may be placed on records if required by the operating company.



**C. Special Symbols—CONECS**

**2.04** The following symbols are used for work print placing/splicing instructions only. They may

# Outside Plant Codes & Symbols: Cable, Cable Terminals, Closures, & Interfaces

ISS 8, SECTION 620-040-015

be shown on an attachment such as a route sketch or as an instructional insert.

- (a) Connectorized cable end.



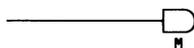
- (b) Wire holding index strip connectorized end.



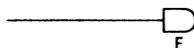
- (c) Bottomless splice module connectorized end.



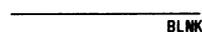
- (d) Male connectorized cable end.



- (e) Female connectorized cable end.



- (f) Blank cable end.



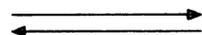
- (g) Pulling eye end.



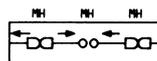
- (h) Direction of placing (or pull).



- (i) Direction of placing is optional.



- (j) Placing block.



## D. Grounding and Bonding (Shown on Work Prints Only)

2.05 The various ways of showing bonding and grounding are as follows:

- (a) Ground.



- (b) Supplemental letter symbols may be added to indicate the termination of the ground. Length of ground wire may also be shown. For example, for ground to a cold water pipe, length of ground wire is shown.



- (c) Other supplemental grounding letter symbols for:

- (1) Grounding plant to power multigrounded neutral vertical ground wire.



- (2) Grounding plant to power multigrounded neutral.



- (3) Grounding plant to power company ground rod.



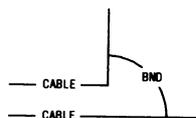
# Outside Plant Codes & Symbols: Cable, Cable Terminals, Closures, & Interfaces

## SECTION 620-040-015

- (4) Grounding plant to telephone company ground rod.

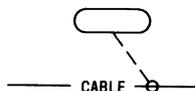


- (d) Bond between separate cable strands.



### E. Taper Codes

**2.06** The taper code is used to identify points in the cable network at cross sections that are of major concern. This code provides a means of identifying the sections across which the fills are taken. For further details, see Section 936-312-110. The taper code symbol is shown as follows:



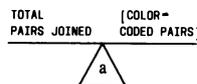
### F. Line and Splicing Symbols

**2.07** The following are symbols required in conjunction with the Network Cost Results Plan (NCRP), JMOS, and Construction Force Management System (CFMS). To determine the number of pairs to be shown for splicing credit, refer to the Outside Plant Construction section of the NCRP. **These symbols are applicable for use on work prints only.**

- (a) Splicing location for work operations other than pair joining work, eg, clear and cap end of cable, building a plastic-insulated conductor (PIC) cable connection point, certain removal operations, etc. The letter inside the symbol is used to indicate where the splicing location number is placed.



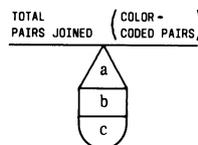
- (b) Splicing location for work operations where pairs are joined. The number inside the symbol and the data required above the symbol are identified as follows:



- (1) Total Pairs Joined—Enter the total number of pairs joined.
- (2) Color-Coded Pairs—Enter the total number of color-coded pairs joined.
- (3) Enter splicing location number.



- (c) Splicing location for work operations involving cable pair, loading coil, or inductor transfers. The letters inside the symbol and the data required above the symbol are identified as follows:



- (1) Enter splicing location number.



- (2) Enter number of regular pairs transferred.



- (3) Enter number of special pairs transferred.



# Outside Plant Codes & Symbols: Cable, Cable Terminals, Closures, & Interfaces

ISS 8, SECTION 620-040-015

(4) Pairs Joined—Enter the total number of pairs joined.

(5) Color Coded—Enter the total number of color-coded pairs joined.

**2.08** The symbols identified in this part of the section are to be used on engineering work prints designed for those DSDCs that convert to the JMOS. These symbols will be used to identify job steps associated with both line and splicing operations and locations. The job step will provide specific information needed to develop basic work unit and standard time increment (STI) calculations. These symbols are intended for use on work prints only.

(a) Line job step symbol. A trapezoid is used to identify line work locations and the material to be placed at the location. Enter line job step number.

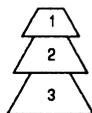


(b) When line and splice work operations appear at the same location and the work will be completed by the same work force, a combined triangle and a trapezoid may be shown. Enter both a line and splicing job step number within the respective symbol.



(c) Line work locations at which two or more work operations are involved may be shown with the job step symbols stacked and job step numbers assigned.

**Example 1:** Job Step 1: Place the pole.  
Job Step 2: Place the anchor.  
Job Step 3: Place the guy.



**Example 2:** Job Step 1a: Dig a splice pit.  
Job Step 1b: Backfill a splice pit.



## G. Cable Terminals, Closures, and Interfaces

**2.09** Outside terminals, closures, and interfaces of 100-pair capacity or larger and building terminals and interfaces, regardless of size, are property retirement units and must be recorded on the continuing property records.

(a) Cross-connect (jumper wire cross-connect) capability.



(b) Serving area interface.



(c) Rural area interface.



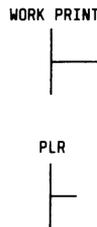
(d) Ready-access terminal.



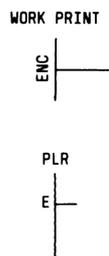
# Outside Plant Codes & Symbols: Cable, Cable Terminals, Closures, & Interfaces

**SECTION 620-040-015**

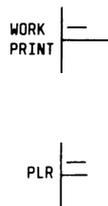
(e) Distribution cable terminal.



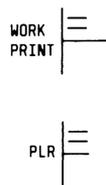
(f) Encapsulated terminal.



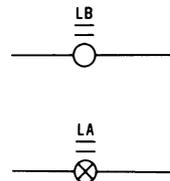
(g) One horizontal line above the terminal symbol indicates the presence of protector units suited to **cable protection**.



(h) Two horizontal lines above the terminal symbol indicates the presence of protector units suited to **station protection**.



(i) Terminals containing fuse chambers such as LA or LB type. (For record purposes only since these terminals are obsolete).

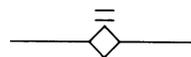


(j) This symbol, when placed adjacent to the terminal symbol, indicates the presence of protector units in an auxiliary housing. [See detailed example in paragraph 4.03(d)].

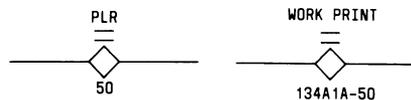
**Note:** For (i) and (j), if the operating company policy is to show information on the Exchange Customer Cable Record (ECCR), it may be deleted from location records.



(k) This symbol denotes protectors used for station protection in buildings served by exposed cable. This unique symbol is retained since it serves a twofold purpose of having not only carbon station protection but also an IN-OUT cable fuse link (134-type protectors).



(l) The 134-type protector and the 190-type protector are used for station protection in buildings served by exposed cable. The number indicates the size. The protector is equipped with separate IN and OUT stubs.

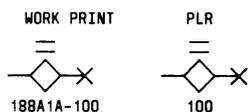


# Outside Plant Codes & Symbols: Cable, Cable Terminals, Closures, & Interfaces

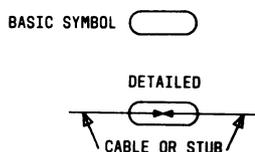
ISS 8, SECTION 620-040-015

(m) The 188- and the 189-type modular protectors are used for station protection in buildings served by exposed cable and have connecting blocks built into terminal building cable or wire.

**Note:** Although these protectors are not re-termination units, they should be shown on outside plant location records for engineering purposes.



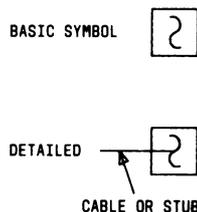
(n) Control point.



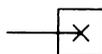
(o) Cross-connected terminal converted to a control point.



(p) Access point.



(q) Cross-connect terminal converted to an access point.



### 3. EXAMPLES OF APPLICATION

**3.01** The following examples illustrate the use of symbols and abbreviations in conjunction with cable terminals, closures, and interfaces on work prints and outside plant location records (PLRs). Since it would be an impossible task to show every example that one may encounter, only a sampling of the uses are illustrated. The 928 Division of the Bell System Practices depicts detailed actual usage for work prints and records. Along with each symbol or combination of symbols used, other information may be necessary to meet the requirements of federal or state regulatory bodies and to conform with the operating company policy or Bell System Practices. This consists of items such as:

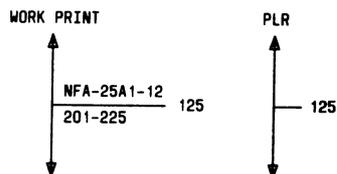
- Informational notes
- Types of hardware
- Accounting codes
- Cable counts and/or other pair assignments
- Wiring limits
- Address or other location identification
- Tax district
- Mortality information
- Detailed pair configurations.

**3.02** Selected data from the list in paragraph 3.01 is used in instances where it is considered necessary to further clarify an example.

#### A. Strand-, Pole-, or Wall-Mounted Distribution Terminals

**3.03** The various types of strand-, pole-, or wall-mounted terminals are shown as follows:

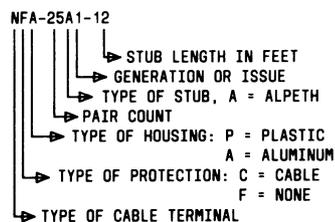
(a) N-type, 25-pair capacity, fixed-count distribution terminal.



# Outside Plant Codes & Symbols: Cable, Cable Terminals, Closures, & Interfaces

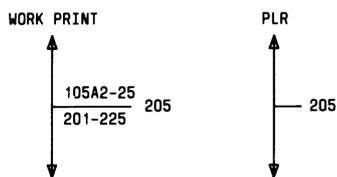
## SECTION 620-040-015

The N-type cable terminals have a coding scheme as follows:

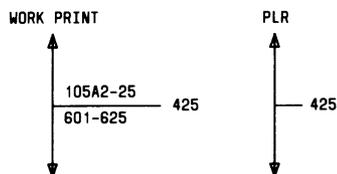


The number shown at the end of the terminal line is the nearest house address.

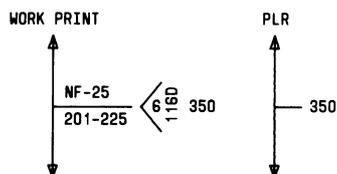
(b) 105-type terminal, 25-pair capacity, fixed-count terminal, pair count 201 through 225.



(c) 105-type terminal, 12-pair capacity, fixed-count terminal.



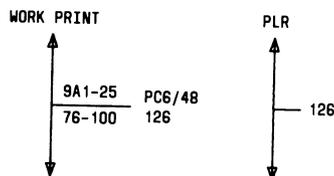
(d) The symbol  $\sphericalangle$  adjacent to the terminal symbol indicates the presence of protector units in an auxiliary housing, such as the 116D protector mounting. The number in the symbol indicates the number of pairs for which protector units have been provided.



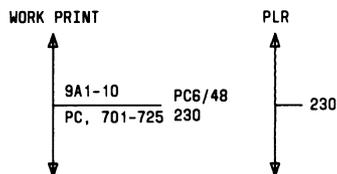
## B. Distribution Terminals—Buried Plant

3.04 Distribution terminals on buried plant are shown in the following ways:

(a) Fixed-count, 25-pair, distribution terminal using the PC 6/48 closure equipped with 1-9A1-25 terminal block, count 76 through 100.



(b) Fixed-count, 10-pair, distribution terminal using the PC 6/48 closure equipped with 1-9A1-10 terminal block.



# Outside Plant Codes & Symbols: Cable, Cable Terminals, Closures, & Interfaces

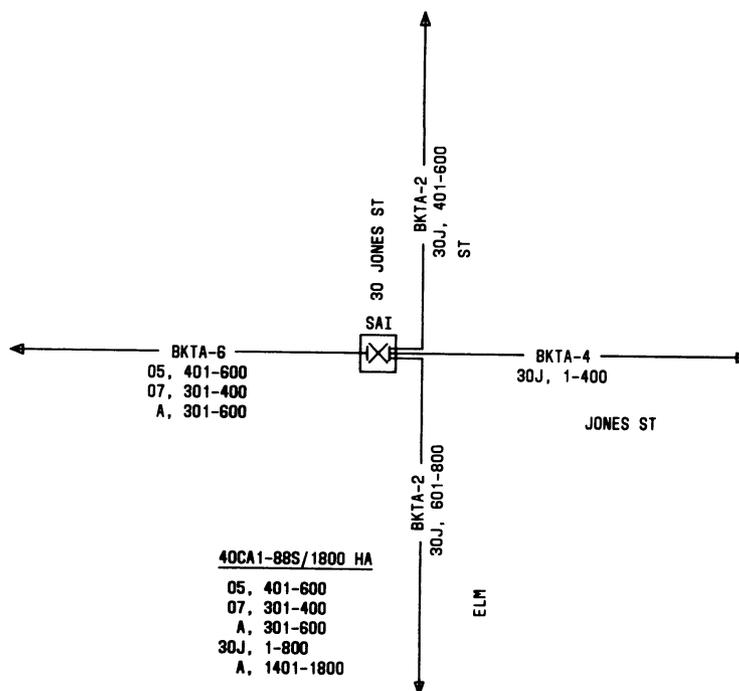
ISS 8, SECTION 620-040-015

**C. Feeder-Distribution Interfaces (FDIs)**

**3.05** The various feeder-distribution interfaces are shown as follows:

- (a) Serving area interface using a 40C-type cabinet with 1800-pair capacity. Full coding is required for both work prints and records. (See Fig. 1.)

EXAMPLE: 40CA1-88S/1800 HA  
 40C—type of cabinet  
 A—aerial mounting  
 1—design issue  
 88—type of connectors  
 S—standard connector option  
 1800—number of pairs capacity  
 HA—raw-ended harness termination method.



NOTE: Illustration depicts PLR posting only.

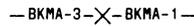
Fig. 1—Example of Feeder-Distribution Interface



# Outside Plant Codes & Symbols: Cable, Cable Terminals, Closures, & Interfaces

ISS 8, SECTION 620-040-015

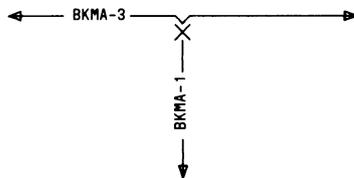
(c) Cross-connecting terminal having IN and OUT pairs in separate cables. IN and OUT pairs are terminated on separate connecting or terminal blocks. Interconnections between pairs are made by means of cross-connecting wire.



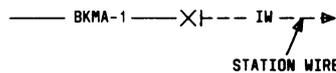
(d) Cross-connecting terminal having IN and OUT pairs in the same cable stub. Interconnections are made by means of cross-connecting wire. The example below shows a factory-terminated, sealed stub.



(e) Cross-connecting terminal having one cable "looped through" and another cable originating at the interface. Cross-connecting wires are used for making connections between cable pairs. The symbol indicates that the BKMA-3 cable is looped through. The terminal count identifies the pairs terminated.



(f) Cross-connecting terminal containing separate connecting or terminal blocks for terminating outside plant cables and station cables or wire. Cross-connecting wires are used for making connections between the outside plant cable and the station wiring.

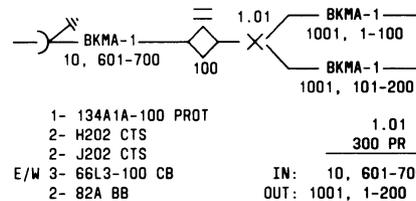


**Note:** If a stub to a cross-connecting terminal is factory-terminated, it is not necessary to record the stub data such as type, size, gauge, mortality, and length. If field-terminated, it is necessary to show this data on the location record.

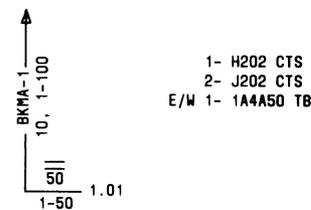
## D. Building Terminals

3.06 Many types of terminals are used in buildings and are shown as follows:

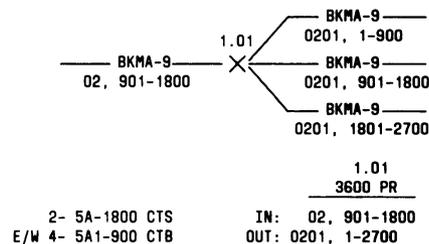
(a) Protected cross-connecting terminal used to connect an exposed entrance feeder cable to building cables. The 1.01 indicates that this is the first terminal on the first floor. An insulating joint with a ground is placed at the entrance to the building.



(b) Protected distribution terminal consisting of a 1A4A terminal block.



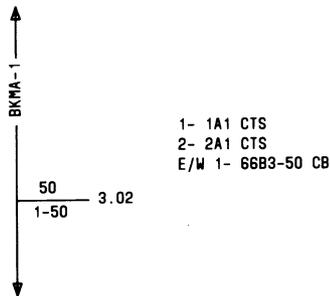
(c) Unprotected cross-connecting terminal installation with 5A1-type cable; terminal blocks mounted on 5A-1800 frame-type cable terminal sections.



# Outside Plant Codes & Symbols: Cable, Cable Terminals, Closures, & Interfaces

**SECTION 620-040-015**

(d) Unprotected distribution terminal consisting of 66-type connecting blocks housed in a cable terminal section.

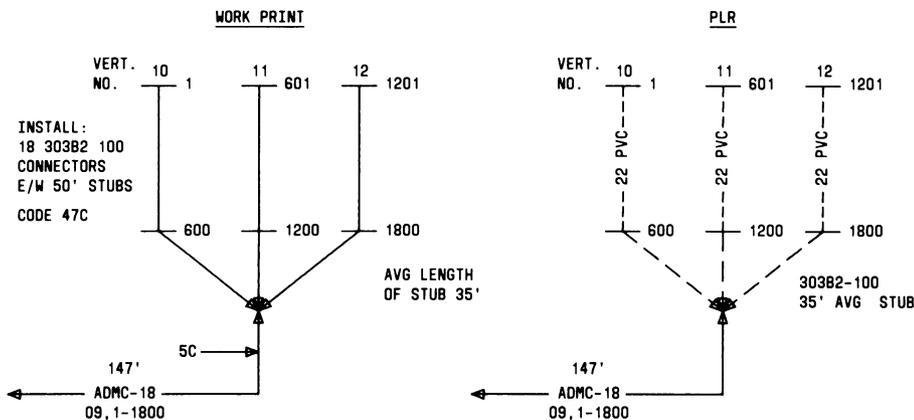


**E. Main Distributing Frame (MDF)—Central Office or PBX**

**3.07** The following examples illustrate the use of symbols in conjunction with terminating outside plant cables at central office or main frame locations.

(a) Terminating an 1800-pair 24-gauge cable by using 303B2, 100-pair connectors equipped with 50-foot, 22-gauge polyvinyl chloride (PVC) stubs. (See Fig. 3.)

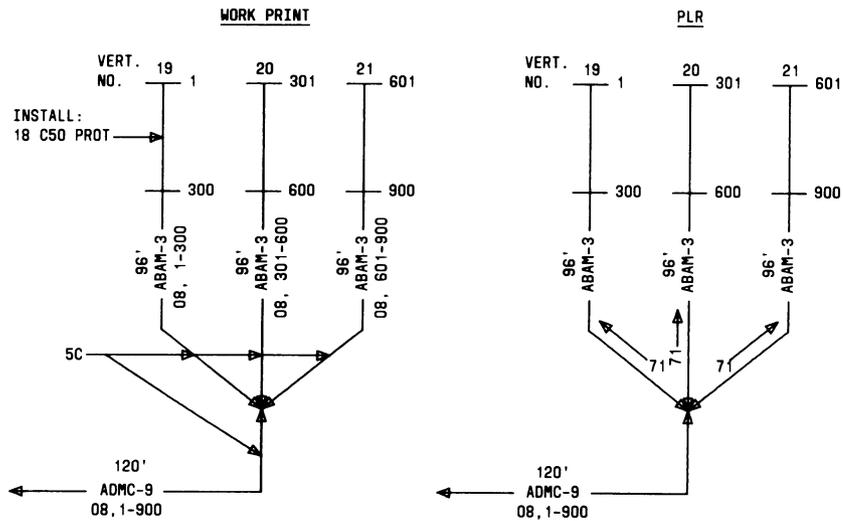
(b) Terminating a 900-pair cable using C50 protectors. (See Fig. 4.) The protectors are not equipped with stubs. The 300-pair ABAM stubs, which are charged to the outside plant cable accounts, are used to terminate the cable.



**Fig. 3—Example of Terminating Cable on an MDF Using Stubs**

# Outside Plant Codes & Symbols: Cable, Cable Terminals, Closures, & Interfaces

ISS 8, SECTION 620-040-015



**Fig. 4—Example of Terminating Cable on MDF Using Protectors**

**F. Taper Code**

**3.08** Figure 5 illustrates how the taper code symbol is shown on an outside plant cable location

record. For further details, refer to Section 936-312-110. The numerical designations #84, #85, and #86, as shown on the illustration, are sections of plant (SOP) codes as described in Section 936-312-100.

# Outside Plant Codes & Symbols: Cable, Cable Terminals, Closures, & Interfaces

SECTION 620-040-015

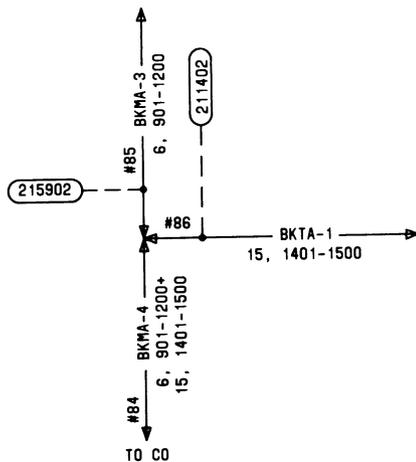


Fig. 5—Example of Taper Coding

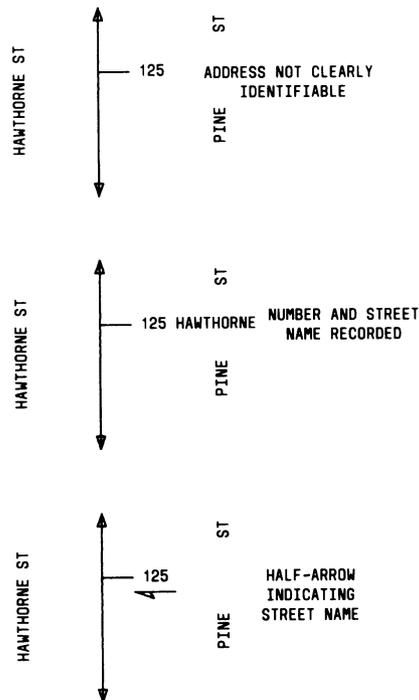


Fig. 6—Optional Method of Recording Terminal Address

### G. Optional Method of Recording Terminal Address

**3.09** Many times, due to congestion or other restriction of space, the street name where a terminal is addressed is not clearly identifiable and both the terminal number and street name must be shown. A line with a half-arrow pointing to the applicable street eliminates the need to include the street name with the terminal number. (See Fig. 6.)

### H. Lightguide Cable and Equipment

**3.10** Lightguide cable, interfaces, terminating units, and the remote multiplexer are shown in Fig. 7 through 11.

# Outside Plant Codes & Symbols: Cable, Cable Terminals, Closures, & Interfaces

ISS 8, SECTION 620-040-015

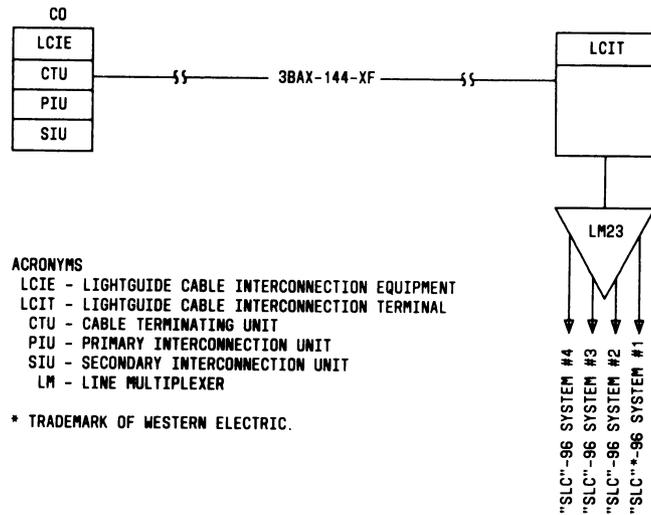


Fig. 7—Lightguide Cable and Equipment

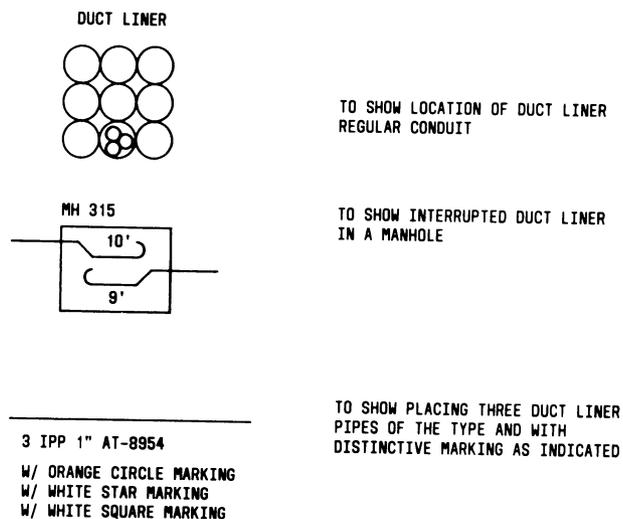


Fig. 8—Duct Liner Symbols

# Outside Plant Codes & Symbols: Cable, Cable Terminals, Closures, & Interfaces

SECTION 620-040-015

THE NEW LIGHTGUIDE CABLE CODE IS OF THE FORM NLLL-NNN  
WHERE N IS A NUMBER AND L IS A LETTER.

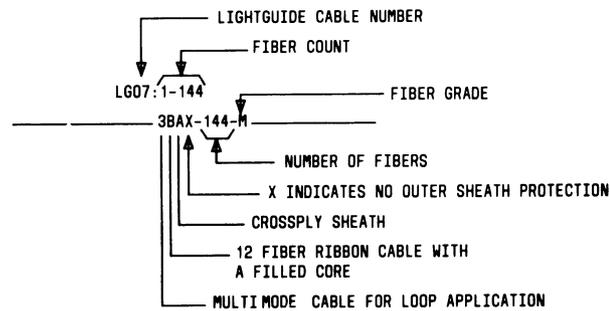


Fig. 9—Lightguide Cable Classification

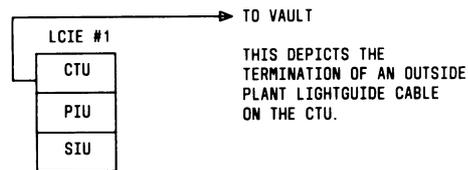


Fig. 10—Lightguide CO Equipment

# Outside Plant Codes & Symbols: Cable, Cable Terminals, Closures, & Interfaces

ISS 8, SECTION 620-040-015

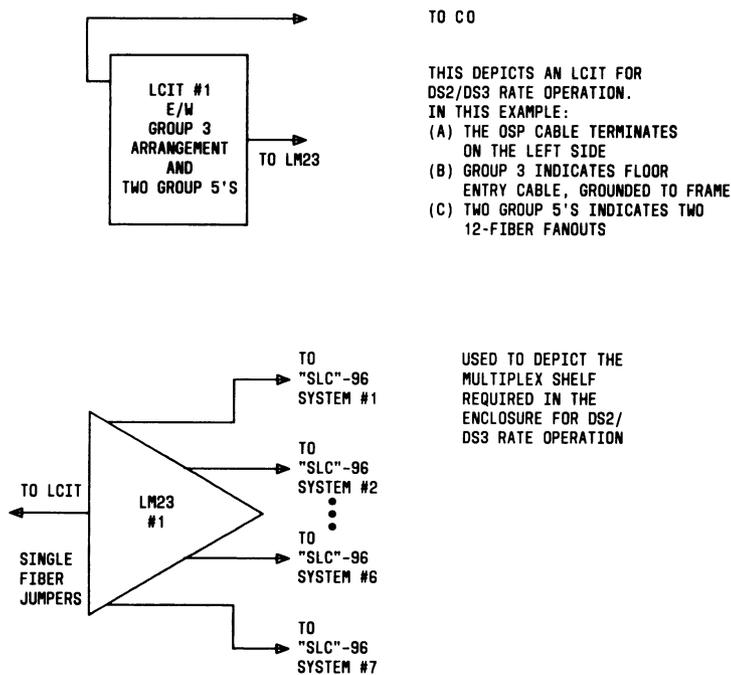


Fig. 11—Remote Terminal Equipment

#### 4. CABLE CODE DESIGNATIONS

##### 4.01 Coded Exchange-Type Cables: A

4-letter code designation is used with exchange-type cables. The 4-letter code consists of:

- (a) First letter—sequence of standardization
- (b) Second letter—type of conductor insulation
- (c) Third letter—gauge and material of conductors
- (d) Fourth letter—type of sheath.

##### A. Cable With Code Designation BHBA-2-BT

4.02 The initial letter, B, represents the sequence of standardization of cables having certain physical characteristics of the core. The H signifies

that the conductor insulation is polyethylene. The third letter, B, represents 19-gauge copper conductors and the fourth letter, A, indicates an alpheth sheath. The 2 indicates there are 200 pairs contained in the cable.

—— BHBA-2-BT ——

##### B. Types of Outer Sheath Protective Coverings

4.03 Outer protective coverings placed over cable sheaths are designated by 2-letter codes. In the example of the cable designation in paragraph 4.02, the suffix BT indicates buried tape armor placed over the sheath of the cable. Section 626-020-011 contains complete details related to this cable coding.

4.04 In the examples of exchange and composite cables shown in paragraphs 4.05 through 4.13,

# Outside Plant Codes & Symbols: Cable, Cable Terminals, Closures, & Interfaces

## SECTION 620-040-015

the cable designation numbers, counts, and in some instances, the gauge of the cable makeup are not a part of the basic symbol. Therefore, they are not shown except to clarify multiple gauge, video, and coaxial groups. Cable designations and counts are covered in Section 928-100-030.

### C. Exchange-Type Cables

4.05 There are two basic formats, depending upon size, as follows:

- (a) **Cables Containing Less Than 100 Pairs**—The pair size is literally written and the code PR is added to the designation of size. For example, a BHBA cable of 16 pairs would be shown as follows:

—— BHBA-16 PR ——

- (b) **Cables Containing 100 Pairs or More**—This example, an ADTC cable of 2700 pairs, applies to cable sizes which are multiples of 100 pairs. For example, 27 indicates the number of 100-pair complements contained in the cable.

—— ADTC-27 ——

### D. Special Cables

4.06 In special cables, the letter S is used as a suffix to the total pairs in the cable, ie, sized other than the normal sizing or a composite cable which may contain more than one gauge or type of conductor. For example:

- (a) A 37-pair special cable.

—— 37S ——

- (b) A 600-pair composite cable.

—— 600-S ——

### E. Composite Cables Containing Exchange Pairs

4.07 A 600-pair composite cable makeup detail may be associated with the proper cable designation in a specific application by indicating the total number of pairs as shown within the break of the cable line. The cable drawing number and the type of sheath are indicated below the cable line. The number of pairs of each gauge of cable, the cable number, and the pair count of the cable are shown to provide further clarity. See Section 928-100-030 for details concerning cable designations and pair counts.

—— 600-S ——  
CA-3113-C  
2-22, 20, 1-200+  
4-24, 36, 1-400

### F. Cables Designated by Drawing Number

4.08 Cables which are manufactured with numbered cable drawings are further described as to *type of sheath* and *sheath protective covering*. **The designation shown is for a cable manufactured according to cable drawing CA-3002. The cable has a lead sheath (L) with jute protection (JP).**

—— 606-S ——  
CA-3002-L-JP

4.09 **Composite Cables Containing Video and Exchange Pairs:** For a 606-pair composite cable, the total number of pairs (606) is shown within the break of the cable line. The cable drawing number and the type of the sheath are indicated below the cable line. The video pairs, conductor insulation, video pair count, size and gauge of nonquadded pairs, cable number, and pair count follow in that sequence.

—— 606-S ——  
CA-3103-H  
6 PR VID PEV-L, 132,  
V1-V6  
6-24, 41, 1-600

# Outside Plant Codes & Symbols: Cable, Cable Terminals, Closures, & Interfaces

ISS 8, SECTION 620-040-015

**4.10 Composite Cables Containing Quads and Exchange Pairs:** For a 306-pair composite cable, the total number of pairs is shown within the break of the cable line. The cable drawing number and the type of sheath are indicated below the cable line. The number, gauge, cable number, and pair count of the quads; the number of 101-pair complements; and the gauge of the exchange cable, cable number, and pair count then are indicated in that sequence.

**Note:** This is for record purposes only. The manufacture of this particular cable makeup has been discontinued.

```

——— 306-S ———
      CA-590-L
52 QD-19, 141, 1-104
2-24, 37, 1-200
    
```

**4.11** The following types of composite cables are designated by the following symbols:

- (a) Composite cables containing spiral-four, disc-insulated quads listed first, and standard quads.

```

——— 78-S ———
      CA-577-C
3 DIQD-16, 176, 1-6
36 QD-19, 176, 7-78
    
```

- (b) Composite cables containing coaxials of size 0.375 inches, interstitial conductors, and standard quads.

**Note:** This is for record purposes only. The manufacture of this particular cable makeup has been discontinued.

```

——— 86-S + 8 COAX ——
      CA-1284-L-LA
8 COAX-0, 375, 342, 1-8
8 COND-19, 342, 9-12
41 QD-19, 342, 13-94
    
```

- (c) Composite cables containing coaxials and exchange pairs. The size of the coaxial cable (0.375) and the use community antenna television

(CATV) have been added. This data may be considered as optional if the need does not exist.

**Note:** Refer to the 626 Division of the Bell System Practices to determine current cable makeups.

```

——— 200-S + 1 COAX ——
      CA-3070-H
1 COAX-0, 375 CATV
2-22, 02, 1-200
    
```

**G. Other Cables Not Coded or Specified by Cable Drawing Number**

**4.12** Cables purchased from commercial sources that are not made to Bell System specifications can be coded for gauge, insulation, and sheath. These are the last three letters of the 4-letter code. An X is shown for the first letter of exchange-type cables. An example is a 50-pair, alpeh sheathed, 19-gauge plastic-insulated conductor cable manufactured by other than Western Electric or its normal suppliers.

```

——— XHBA-50 ——
    
```

**H. Conversion From Obsolete Numerical Coding to Letter Coding**

**4.13** The present standard 4-letter exchange cable code is recorded on work prints and location records. When working on cable location records, the number codes indicating the size and gauge of the cable should be converted to the standard 4-letter code. For example:

- (a) The code 12-26 could become BSTL-12 and 4-24 could become DSML-4.

```

NUMERICAL CODING
— 12-26 —→— 4-24 —
    
```

- (b) The type of sheath, L, indicates that the cable has a lead sheath.

```

LETTER CODING
— BSTL-12 —→— DSML-4 —
    
```

---

# TV-B-Gone Jammer

---

## Overview

This is a simple electronics project which can be used to disable the use of any *TV-B-Gone*-type devices – or any infrared (IR) TV remote control, for that matter.

The whiny, rich, fascist, \$2600 reading nutcases have found a new toy. This toy is called the "TV-B-Gone", and it's essentially just a universal TV remote control which is used to turn televisions off. Of course, all the whiny rich kids are going around turning off *other* people's televisions – a direct violation of personal privacy. Hey! Wait a minute... Don't \$2600 readers complaining about their personal privacy all the time? Shouldn't people be allowed to watch TV as they wish?

Well, this little hack consists of several fairly intense pulsed infrared LEDs which can be used to "confuse" (i.e. jam) the infrared receiver on most TV, VCR, stereos, cable boxes, etc.



**The 1933 version of the *TV-B-Gone*.**

## Construction

This construction is very simple. Most of the parts are available at Radio Shack. It just consists of a 555-timer IC configured to output a series of pulses at around 38 kHz. These pulses then trigger four infrared LEDs, which radiate their energy and jam any IR remote receivers in the vicinity. The 38 kHz signal is actually quite critical, as that is the carrier frequency (or something close to it) which is used by normal IR remote controls. By encoding the remote's control data stream onto a high-frequency carrier like this, any interference (especially sunlight – which is DC) is ignored.

When constructed and powered, place the jammer in the field-of-view of the TV you wish to protect. This should prevent the TV from receiving any external IR remote signals. When you need access to the remote, just remove the battery from the jammer. This could be very useful for protecting a large store display of TVs, or any TVs in a bar or restaurant setting.

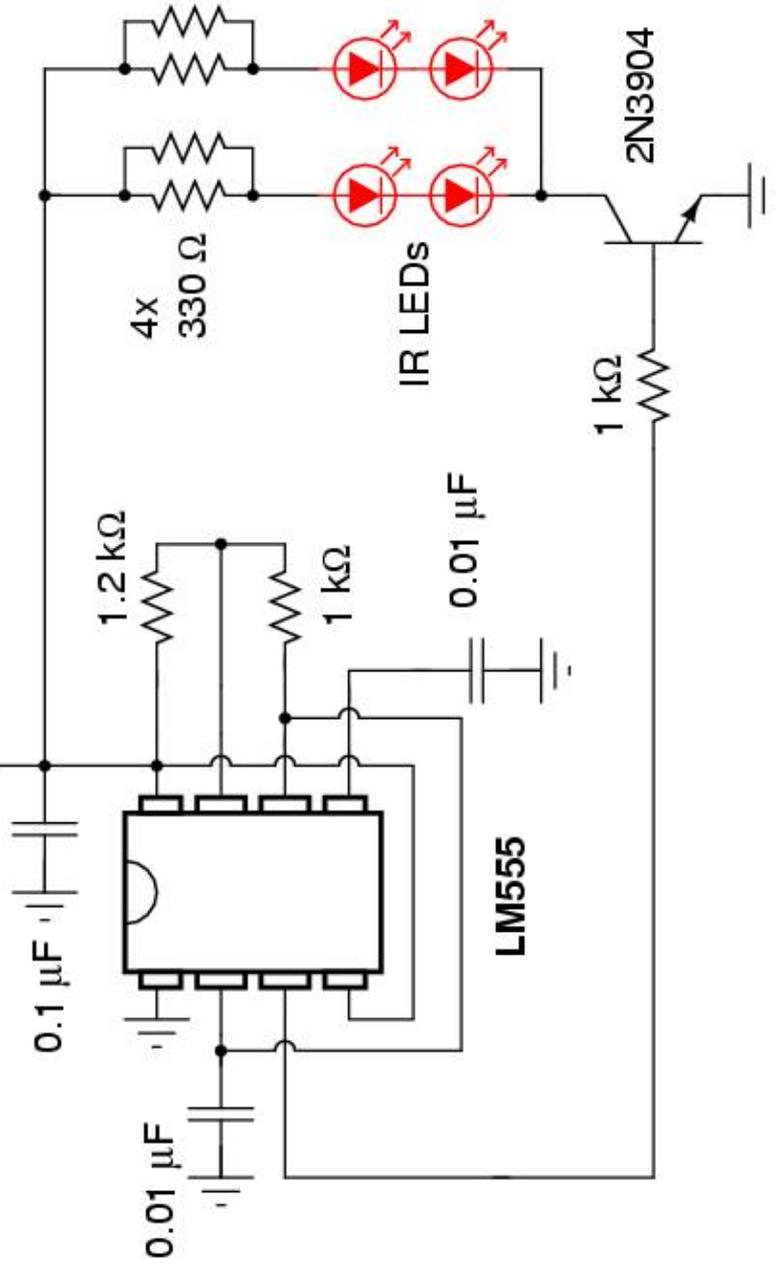
Schematic

# TV-B-Gone Jammer

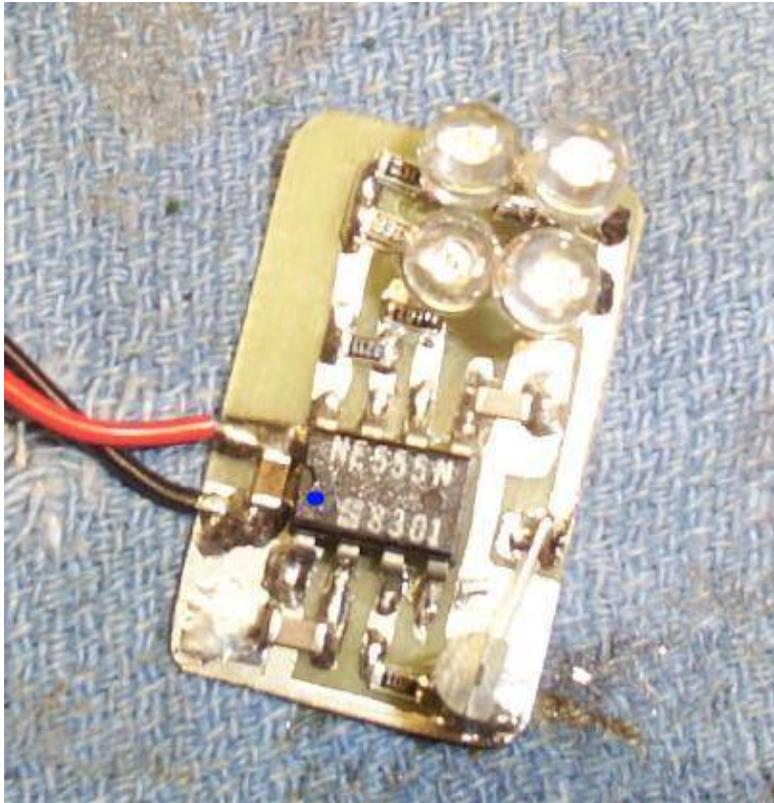
(Infrared Remote Control Jammer)

Tune to approximate  
38 kHz

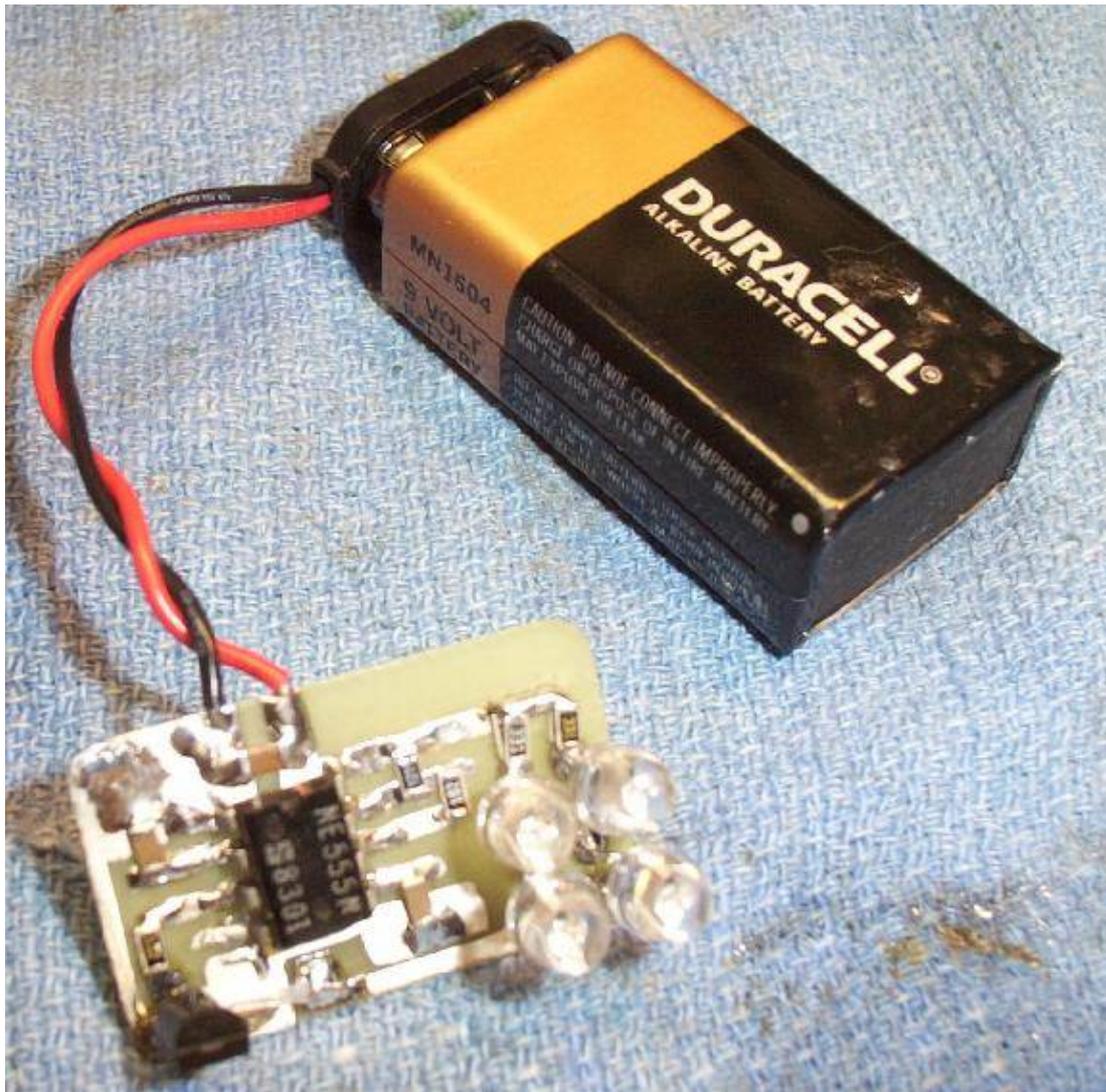
+9 VDC



## Pictures



Closeup picture of the IR jammer. Pin 1 of the 555-timer is marked with the little blue dot. The circuit is powered directly from a 9 Volt battery. You can salvage IR LEDs from old remote controls. Using the resistor/capacitor values in the schematic will get you very close to the required 38 kHz oscillator frequency. Fine tune the 1.2 kohm resistor if the circuit does not oscillate correctly.



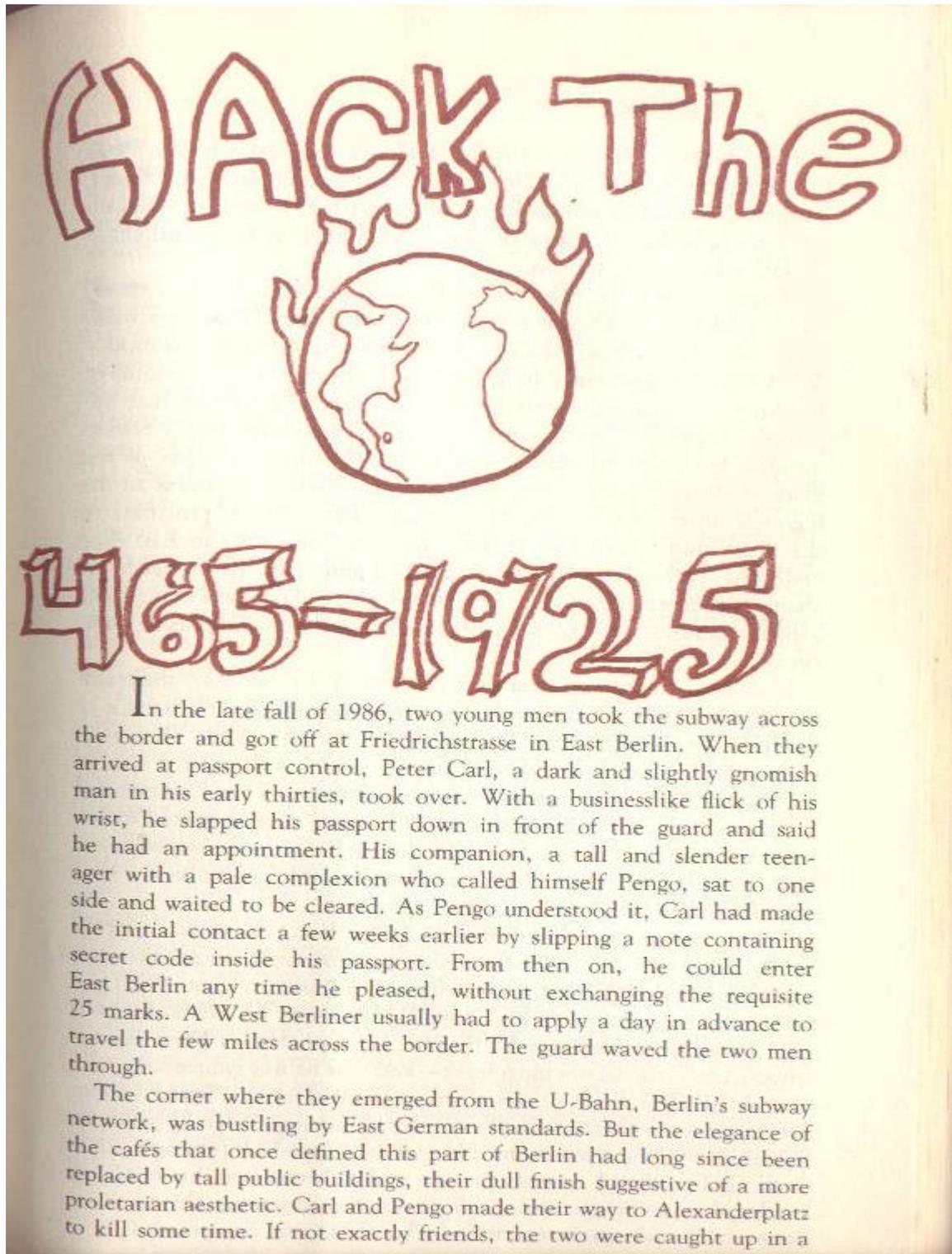
Alternate view. Leave a little bit of lead length on the IR LEDs to allow you to "tweak" the direction of jamming.

---

## Bonus

---

Found in the library:



---

## End of Issue #14



Any Questions?

---

### Editorial and Rants

*But I was told by those rich college professors that Marxism is the future! Eric Corley told me white people cause all the problems in the world! LOL!*

Title : A Morsel of Goat Meat (Black Zimbabweans Want the Return of White Rule)  
Source : NY Times - <http://www.nytimes.com/2005/03/23/opinion/23kristof.html?oref=login>  
: <http://www.libertypost.org/cgi-bin/readart.cgi?ArtNum=89121>  
Published : Mar 23, 2005  
Author : NICHOLAS D. KRISTOF

The hungry children and the families dying of AIDS here are gut-wrenching, but somehow what I find even more depressing is this:

Many, many ordinary black Zimbabweans wish that they could get back the white racist government that oppressed them in the 1970's.

"If we had the chance to go back to white rule, we'd do it," said Solomon Dube, a peasant whose child was crying with hunger when I arrived in his village. "Life was easier then, and at least you could get food and a job."

Mr. Dube acknowledged that the white regime of Ian Smith was awful. But now he worries that his 3-year-old son will die of starvation, and he would rather put up with any indignity than witness that.

An elderly peasant in another village, Makupila Muzamba, said that hunger today is worse than ever before in his seven decades or so, and said: "I want the white man's government to come back. ... Even if whites were oppressing us, we could get jobs and things were cheap compared to today."

His wife, Mugombo Mudenda, remembered that as a younger woman she used to eat meat, drink tea, use sugar and buy soap. But now she cannot even afford corn gruel. "I miss the days of white rule," she said.

Nearly every peasant I've spoken to in Zimbabwe echoed those thoughts, although it's also clear that some still hail President Robert Mugabe as a liberator. This is a difficult place to gauge the mood in, because foreign reporters are barred from Zimbabwe and promised a prison sentence of up to two years if caught. I sneaked in at Victoria Falls and traveled around the country pretending to be a tourist.

The human consequences of the economic collapse are heartbreaking. I visited a hospital and a clinic that lacked both medicines and doctors. Children die routinely for want of malaria medication that costs just a few dollars.

At one maternity ward, 21 women were sitting outside, waiting to give birth. No nurse or doctor was in sight, and I asked the women when they had last eaten meat, eggs or other protein. They laughed uproariously. Lilian Dube, a 24-year-old who had hiked 11 miles to get to the hospital, said that she had celebrated Christmas with a morsel of goat meat.

"Before that, the last time I had meat was Christmas the year before," she said. "I just eat corn porridge and mnyi," a kind of wild fruit.

An elementary school I visited had its fifth graders meeting outside, because it doesn't have enough classrooms. Like other schools, it raises money by charging fees for all students – driving pupils away.

"Only a few of the kids who started in grade one are still with me in school," Charity Sibanda, a fifth-grader, told me. "Some dropped out because they couldn't pay school fees. And some died of AIDS."

As many as a third of working-age Zimbabweans have AIDS or H.I.V., and every 15 minutes a Zimbabwean child dies of AIDS. Partly because of AIDS, life expectancy has dropped over the last 15 years from 61 to 34, and 160,000 Zimbabwean children will lose a parent this year.

AIDS is not President Mugabe's fault, but the collapse of the health system has made the problem far worse.

The West has often focused its outrage at Mr. Mugabe's seizure of farms from white landowners, but that is tribalism on our part. The greatest suffering by far is among black Zimbabweans.

I can't put Isaac Mungombe out of my mind. He's sick, probably dying of AIDS, and his family is down to one meal a day. His wife, Jane, gave birth to their third child, Amos, six months ago at home because she couldn't afford \$2 to give birth in the hospital. No one in the family has shoes, and the children can't afford to attend school. They're a wonderful, loving family, and we chatted for a long time – but Isaac and Jane will probably soon die of AIDS, and the children will join the many other orphans in the village.

When a white racist government was oppressing Zimbabwe, the international community united to demand change. These days, a black racist government is harming the people of Zimbabwe more than ever, and the international community is letting Mr. Mugabe get away with it. Our hypocrisy is costing hundreds of Zimbabwean lives every day.