TI92 and TI89 structures, variables, and information.

© 2000 Gareth James

(Preliminary Version)

Contents

About

I thought that It would be useful to release it now so that I can get some feedback. If you have any suggestions or additions please email me at:

mailto:<Gareth.James@bigfoot.com>

Updates to this document will be available at
http://www.bigfoot.com/~Gareth.James

Preliminaries

Document Conventions

Code and structers are contained in certain coloured boxes:

	RAM: .bss member (static)
	RAM: export member non-constant extern data
	ROM: static function, const data, or enumeration
	ROM: exported function
	Variable type description
	Non-strict Description
	Enumeration, or define
	Code Example
	Resource (non-local)

General naming conventions:
Vectors/Interrupts have the postfix _int, ASM routines are generally prefixed by _. The posfix _aux is generally used for the reentrant function of a recursive routine, where the entry point has the given name.

This document has existed in various forms over many different ROM versions. Currently most of the information is based around the TI92+ AMS 1.01. Hopefully it is clear when information applies to different

ROM versions. Usually TI92+/TI89 specific information is placed withing "#ifdef PLUS" ... "#endif" blocks. AMS 2 specific information, although limited for now, should be marked "AMS2".

The following name changes may have occured for types for the TI92+/TI89 range:
Boolean -> BOOL
EStackIndex -> ESI ?

On AMS2 several entries in the jump table have been replaced, here are a few:
NoCallback -> ERD_hSym
caddcert -> bss plotgraph.o e.g. zFit, seqmode
EM_blockErase -> ?

NA_WORD mean Non Aligned word.

Compiler Conventions

C functions are passed paramaters via the stack. Byte values are aligned to the nearest word. So byte paramaters are pushed as word, and the LSB is read from the stack. Structures are copied directly to the stack frame, generally this is avoided for efficiency reasons.

Local variables are stored in a stack frame. For small frames this is generally done by adjusting the stack pointer (sp), although for larger frames the link instruction is used via a6, variables and paramaters are then addressed relative to a6. This stack frame also acts as a buffer for the return value of called functions:

Returned types less than 4 bytes are returned in d0, pointers are returned in a0. Larger structures are stored in the calling functions stack frame accessed via (a6) [this is why link gets its name, the stack frames are in fact linked lists, so (a6) gives the address of the previous functions stack frame]. These structures are saved at the beginning of the stack frame, following this are the paramaters.

This return buffer is used by the bcd_math package, which uses it for its registers FP0-FP7 which are allocated as required by the function.

Linking:
Object files are linked progressivly, i.e. the RAM segments and ROM segments will be in the same order. Generally there is a 4 byte alignement between these segments. Note that on PLUS versions the modules are approximately in alphabetical order.

In the RAM we have the .bss segment is used for non-const static (non-external) data; static return values of functions and uninitialised static data. Following this in the RAM is the non-const external uninitialised data (this also has 4 byte alignment).

In the ROM we have the .text segment, which contains the code (functions) and const data, there is no specific organisation of this segment.

Standard Includes

```
#ifndef TRUE
#define TRUE  (1u)
#define FALSE (0u)
#endif

typedef unsigned char  BYTE;
typedef unsigned char  UCHAR;
typedef unsigned short BOOL;
typedef unsigned short WORD;
typedef unsigned short USHORT;
typedef unsigned int   UINT;
typedef unsigned long  DWORD;
typedef unsigned long  ULONG;
typedef signed char    SBYTE;
```

```
typedef signed char    SCHAR;
typedef signed short   SWORD;
typedef signed short   SSHORT;
typedef signed int     SINT;
typedef signed long    SDWORD;
typedef signed long    SLONG;

typedef double FLOAT;
typedef DWORD size_t;

typedef union {
  void *v;
  BYTE *b;
  WORD *w;
  ULONG *l;
  SBYTE *sb;
  SWORD *sw;
  SLONG *sl;
} PTR_ALL;
```

```
//Proc pointers:
typedef (* const ProcPtr)(void);

#define ProcTableVector (200)

#define ASAP_init \
        ProcPtr * const ProcTable = *(ProcPtr **) ProcTableVector

// A method to define ASAP functions, so future changes to the table dealt with
#define ASAP_ref(_index, type) (*(type)ProcTable[(index)])
#define ASAP_UNKNOWN (*const)
```

Example of Proc references:

```
#define DrawStr ASAP_ref(425, void (*const)(WORD, WORD, const char *, WORD))
void _main()
{
  ASAP_init;
  DrawStr(0, 0, "Hello World!", 0);
}
```

OS Hardware Interface

- OS Variables and Initialisation
- General OS Routines
- Queues (IO Buffering)
- Link Control
- System Timers
- Keyboard Handling

OS Variables and Initialisation (init.o)

In the first sector very little resides. Following the certificate and vector table is this module and the flash module. This routine hooks all the unused vectors and points them to ErrorLockup. Vectors are hooked by relevevant routines that need them, most of which are in OS.o (in the second sector). Auto-ints tend to have specific uses and therefore a specific module encapsulating their use. Anyway, here is the trivial stuff:

```
void _init_all( void ); /* Reset vector */
```

```
// init stubs called by _init_all:
static void _init_Heap( void );
static void _init_Timers( void );
static void _init_Screen( void );
static void _init_Link( void );
static void _init_Keys( void );
```

```
static void _init_APD( void );
static void _init_BATT( void );
static void _init_Home( void );

DWORD _rom_size( void );
/* 0x020000 or 0x040000 */

//unsupported vectors (lockup)
static void _bus_error_int( void );
static void _address_error_int( void );
static void _illegal_instr_int( void );
static void _zero_divide_int( void );
static void _chk_instr_int( void );
static void _trapv_instr_int( void );
static void _priviledge_int( void );
static void _trace_int( void );

void _linea_int( void );
// Throw error from (0xAeee)

//lockup:
static void _linef_int( void );
static void _spurious_int( void );
static void _auto3_int( void );

void _mem_protect_int( void ); /* auto-int 7 */
/* Low memory protection (memory below 0x400 written)
 "Throws Protected Memory Violation" if not caused by USP (stack overflow)
 else Throws "Memory" variant (673)
*/

WORD _setSR_int( void );  /* Trap #1 */
/* d0 = new SR */

//lockup:
static void _trap5_int( void );
static void _trap6_int( void );
static void _trap7_int( void );

void _ToSupervisor_int( void ); /* Trap #12 */
/* Switch to supervisor mode */

//lockup:
static void _trap13_int( void );
static void _trap14_int( void );
static void _trap15_int( void );

void ErrorLockup( void );

//stubs for ErrorLockup calls:
static void _FontSetSys( void );
static void _strlen( void );
static void _DrawStr( void );
static void _OSClearBreak( void );
static void _idle( void );

void _trap8_int( void );
/* used by following function: */

void _run_link_prog(address);
/* uses trap 8 which runs a block of the following form in supervisor mode:
   LE_WORD Size;
   BYTE Code[];
the code called (passed a5 with the end address), trap #2 is then called (put on the
stack as the return address, called directly if not PLUS)
It is easy enough to avoid the reset from the ASM code, so this should allow an easier
method to take control of non-plus calculators.
See High Level (link.o)
*/
```

```
void _MEMTable_int( void ); /* Trap #9 */
/*
input:  d0.w = address id
return: a0.l = pointer

ids:
0 OSContrastUp
1 WinOpen
2 OSLinkReset
3 TIMERV *OSTimerVectors
4 BYTE *OSContrast //end of LCD_MEMORY
5 WinStr
6 KEY_QUEUE *KeyBuffer
7 OSqclear
8 CHARTYPE* CharTbl;
9 OSContrastUp
10 OSContrastDn
11 OSClearBreak;
12 KEYCODE* KeyCodes;
13 OSCheckBreak;
14 LCD_MEM;
15 OSdequeue
16 RAMTest
17 WinMoveTo
*/

BYTE OSContrast;
BYTE unknown; //padding
BYTE OSOnBreak;
BYTE OSDisableOnBreak;
WORD OSOldSR; /* idle (Trap #0) */
WORD TimerMagic; /* for some check or other, not used on AMS2 */

TIMER OSTimers[7];
BOOL OSTimersExpired[7];
VTIMER OSVTimers[7];

union { /* This func used to reside in RAM, but locks out flash */
  WORD Buf[12];
  void Func( void );
} SetPowerState;

LINK_QUEUE OSLinkTxQueue;
LINK_QUEUE OSLinkRxQueue;
BYTE OSLinkFlushed;
BYTE OSLinkIntActive; /* stop reentry into interrupt (5) */
BYTE OSLinkRxFlag;
BYTE OSLinkTxFlag; /* ngetchx waits for tx buf to clear if this is set */
BYTE OSLinkOpenFlag;
BYTE OSLinkHasBeenClosedBefore?; /* only ever cleared once */

WORD OSLinkOldSR;
WORD OSScreenNotUsed[4]; /* 0x400, 0[3] */

// ... new object? ...

DWORD OFF_OldSSP;
DWORD OFF_Magic;
DWORD OFF_ChkMem; /* SumStoChkMem() */

BYTE BATT_Level; /* hardware level, makes sure state is stable */
BYTE BATT_Counter; /* so that runs every 3 ticks of timer */
BYTE BATT_STLevel; /* 3-hardware_value -> status value */

BYTE OSContrastMask; /* max value */
BYTE MemUnprotectedFlag; /* set on reset, according to ($600000).2 */

void *EndOfMemory;
```

```
// ... new object? ....

ER_FRAME* ER_CurFrame;
```

General OS Routines (OS.o)

Power handling (including contrast), break flag, and keyboard scanning.

```
enum OSStatKeys {2ND, DIAMOND, SHIFT, HAND };

void _OSFunc_int( void ); /* Trap #0 */
/* jumps to other supervisor mode functions here, offset d0 from next func
These routines should never need to be referred to directly or via this offset, as these
routines merely support the external routines that follow.
*/

// OSFunc 1 (d0=0):
static void _off( void ); /* Trap #4, and Trap #0 */

/* Used in off and idle to set the power state, switches flash to low power mode, so
needs to execute from RAM */
static void _set_power_state( void );
static void _set_power_stateR( void );
/* the RAM function */

// OSFunc 2:
static void _idle( void );
// OSFunc 3:
static void _ClearBreak( void );
// OSFunc 4:
static void _CheckBreak( void );
// OSFunc 5:
static void _Reset( void );
// OSFunc 7:
static void _ErrorLockup( void );

void OSLockup( void );

void OSReset( void );
#define OSReset ASAP_ref(660, void (*const)( void ))
void OSContrastUp( void );
#define OSContrastUp ASAP_ref(662, void (*const)( void ))
void OSContrastDn( void );
#define OSContrastDn ASAP_ref(663, void (*const)( void ))
void OSContrastSet( void );

void OSKeyDown_int( void ); /* auto-int 2 */
void OSOnKeyDown_int( void ); /* auto-int 6 */

WORD OSKeyScan( void );
#define OSKeyScan ASAP_ref(664, WORD (*const)( void ))

static const WORD BitIndex[9] = { 0, 1, 2, 0, 3, 0, 0, 0, 4 }; /* Number of set bit */
static void OSKeyDelay( void );

WORD OSGetStatKeys( void );
#define OSGetStatKeys ASAP_ref(665, WORD (*const)( void ))
/* Return status line flags, see OSStatKeys enum */

void off( void );
#define off ASAP_ref(666, void (*const)( void ))
void idle( void );
#define idle ASAP_ref(667, void (*const)( void ))

void OSClearBreak( void );
#define OSClearBreak ASAP_ref(237, void (*const)( void ))
BOOL OSCheckBreak( void );
```

```
#define OSCheckBreak ASAP_ref(236, void (*const)( void ))
void OSDisableBreak( void );
#define OSDisableBreak ASAP_ref(239, void (*const)( void ))
void OSEnableBreak( void );
#define OSEnableBreak ASAP_ref(238, void (*const)( void ))
WORD OSSetSR(WORD NewSR);
#define OSSetSR ASAP_ref(668, WORD (*const)( WORD ))
void OSScreenReset( void );
void SumStoChkMem( void );
#define SumStoChkMem ASAP_ref(661, void (*const)( void ))

static void _SumChkMem( void ); /* sum 0x400 to 0xFFF + Heap to EndOfMemory */
static void __SumChkMem( void ); /* sum from a0 to a1 */
```

Queues (IO Buffering) (Queue.o)

Keyboard and Link IO is buffered though Queues, these have the same structure, and interface provided by this module. This size of the queue is variable, but for the purposes of this module it is probably defined as follows

```
typedef struct
{
  WORD Head;
  WORD Tail;
  WORD Size;
  WORD Used;
  WORD Buffer[2];
} DEF_QUEUE;

#define QUEUE(s) typedef struct { WORD Head, Tail, Size, Used, Buffer[s/2]; }

BOOL OSenqueue(WORD data, QUEUE *q);
#define OSenqueue ASAP_ref(937, BOOL (*const)(WORD, QUEUE *))
static WORD * _enqueue( void ); /* asm func, a0 = * queue */

BOOL OSdequeue(WORD *dest, QUEUE *q);
#define OSdequeue ASAP_ref(938, BOOL (*const)(WORD *, QUEUE *))
static WORD * _dequeue( void );

BOOL OSqinquire(WORD *dest, QUEUE *q);
#define OSqinquire ASAP_ref(939, BOOL (*const)(WORD *, QUEUE *))
/* same as deque but returns ! result */

static WORD _inquire( void );

WORD OSqhead(WORD *dest, QUEUE *q);
#define OSqhead ASAP_ref(940, WORD (*const)(WORD *, QUEUE *))
/* dest not used here, instead returns result */

static WORD _head( void );

void OSqclear( void );
#define OSqclear ASAP_ref(941, void (*const)( void ))
/* resets QUEUE to {0, 0, 2, 0} */
```

Link Control

Firstly "OSLink.o" provides buffering of the link port into the link queue. "link.o" both extends the OS routines by automating reading and writing large amounts of data. And provides the interface to the home screen where packets are checked for valid commands. These commands are also processed in "link.o". Further there is another object providing the commands to access the link port.

- Low Level
- High Level
- TI-BASIC Interface

Low Level (OSlink.o)

Queueing and dequeueing of link operations.

```
#define LINK_QUEUESIZE 128

/* See OS Queues */
QUEUE(LINK_QUEUESIZE) LINK_QUEUE;
```

```
void link_int( void ); /* auto-int 4 */

void OSLinkReset( void );
#define OSLinkReset ASAP_ref(588, void (*const)( void ))
void OSLinkClose( void );
#define OSLinkClose ASAP_ref(590, void (*const)( void ))
void OSLinkOpen( void );
#define OSLinkOpen ASAP_ref(589, void (*const)( void ))
void _OSLinkQInit( void );
/* a0 = address */

WORD OSLinkTxQueueInquire( void );
#define OSLinkTxQueueInquire ASAP_ref(593, WORD (*const)( void ))
/* returns number bytes free in buffer (size-cursize) */

BOOL OSLinkTxQueueActive( void );
#define OSLinkTxQueueActive ASAP_ref(594, BOOL (*const)( void ))

BOOL OSWriteLinkBlock(const void *src, WORD size);
#define OSWriteLinkBlock ASAP_ref(592, BOOL (*const)(const void *, WORD))
/* writes src to linkblock, if size will not fit in buffer, routine will return;
returns non-zero if error */

WORD OSReadLinkBlock(void *dest, WORD size);
#define OSReadLinkBlock ASAP_ref(591, WORD (*const)(void *, WORD))
/* returns actual size read */

void OSLinkSetSR( void );
void OSLinkRestoreSR( void );

const char * OSLinkActive = "LINK TRANSMISSION ACTIVE: ON KEY ABORTS TRANSFER";
const char * OSLinkError = "LINK TRANSMISSION ERROR";
const char * OSLinkComplete = "LINK TRANSFER COMPLETE";
```

High Level (link.o)

This deals with packets in TI's format, see Link., and specifically Packets.
I haven't had the chance to look at the module yet. The information is duplicated here for consitstency.

Ater any VAR_CMD is send the recipient should send OK, follows by WAIT_DATA. After which the DATA_PART is sent.

```
enum LIO_CMD {
  VAR_HEADER = 0x06,     /* VAR_CMD */
  WAIT_DATA = 0x09,      /* optional size (DataLen can store this length) */
  SEND = 0x0D,           /* ?VAR_CMD - not used */
  DATA_PART = 0x15,      /* all Data (with checksum), i.e. sent after VAR_CMD packet */
  REFUSED = 0x36;
  OK = 0x56,
  CHK_ERROR = 0x5A,
  ISREADY = 0x68,
  SCREEN_DUMP = 0x6D,
  CONTINUE = 0x78,
  DIRECT_CMD = 0x87,     /* WORD(KEY) (stored in packet word [dataLength]) *
  EOT = 0x92,
  REQUEST = 0xA2,        /* VAR_CMD (var size 0) */
  REQUEST_SIZE = 0xB7,   /* VAR_CMD (request with size specified) */
```

```
  EXT_VAR_HEADER = 0xC9,      /* VAR_CMD (PLUS) */
};

//Used in VAR_HEADER struct sent after standard packet
enum LIO_VAR_CMD { /* or type */
  EXPR_VAR = 0,
  LIST_VAR = 4, MATRIX_VAR = 6,
  DATA_VAR = 0xA,
  TEXT_VAR = 0xB,
  STRING_VAR = 0xC,
  GDB_VAR = 0xD,
  FIG_VAR = 0xE,
  PIC_VAR = 0x10, GET_PIC_VAR = 0x11,
  PRGM_VAR = 0x12, FUNC_VAR = 0x13,
  MAC_VAR = 0x14,
  EXEC=0x15,            /* In Link protocol, cause execution of assembly block */
  DIR_LIST = 0x19,      /* recursive list, unless SINGLE_DIR follows tag */
#ifdef PLUS
  SINGLE_DIR = 0x1A,    /* list the folder table entries */
  FOLDER_LIST = 0x1B,   /* SINGLE_DIR should follow, list contents of given folder */
  OTHER_VAR = 0x1C,
#endif
  BACKUP = 0x1D,
  FOLDER_VAR = 0x1F,    /* in group files; folder entry symbol */
#ifdef PLUS
  GETCERT = 0x20;       /* (REQUEST) FL_getCert (request like any file) */
  ASM_VAR = 0x21,
//EXT_VAR
  ADDFLASH = 0x23;      /* FL_download */
  ADDCERT = 0x25;       /* FL_addCert */
#endif
};

/* Not actually used
typdef struct {
  DWORD VarLen;
  BYTE VarType;   /* can be extended to 2 bytes for DIR_LISTS */
  BYTE NameLen;
  char Name[1];   /* should be [NameLen] */
} LIO_VAR_CMD_HEADER;
*/

typedef struct {
  BYTE DevType;
  BYTE Cmd;
  WORD DataLen; /* not including checksum */
} LIO_PACKET;

//char Data[1];   /* should be [] */
//WORD checksum;  /* follows data */

typedef struct {
  HANDLE hVar;
  union {
    const void *pVar;   /* if hVar == H_NULL */
    struct {
      WORD FindFlags;
      WORD NameSym;     /* length of Name */
    } DirVars;
  } extra;
  const char *VarName;
  WORD VarSize;
  WORD Index;     /* for CBL access - data sent as float array */
  WORD Unknown;   /* sent after Index */
  BYTE VarType;
  BYTE VarCompat; /* This trails the var name */
} LIO_CTX;

typedef WORD LIO_ERR;
```

```
const char * DefaultVarName = (char*){0xFF, 0};

LIO_ERR LIO_Send(LIO_CTX *file, BYTE DevType);
#define LIO_Send ASAP_ref(86, LIO_ERR (*const)(LIO_CTX *, BYTE))
LIO_ERR LIO_Get(LIO_CTX *file);
#define LIO_Get ASAP_ref(87, LIO_ERR (*const)(LIO_CTX *))
/* 1 senddata error
 * 5 name too long
 * 7 check error
 */

LIO_ERR LIO_GetMultiple(LIO_CTX *);
#define LIO_GetMultiple ASAP_ref(89, ASAP_UNDEF)

static void SendRefused(BYTE code, BYTE DevType);
/* Send 5 byte Refused structure: {5, 0, code, 0, 0 } */

static const ER_CODE LIO_ProductErr[4] = { 185, 965, 885, 305 };
/* Checksum error, Unlicensed product code, Signature error, Expired product code */
static LIO_ERR GetDigestThrow(WORD Size, BYTE DevType);

static LIO_ERR SendVarCmd(LIO_CTX *file, BYTE DevType);
/* Send LIO_VAR_CMD_HEADER etc. from file info */

static LIO_ERR SendVarData(LIO_CTX *file, BYTE DevType);
/* Send CMD_HEADER (name etc.) and DATA */

LIO_ERR LIO_SendData(const void *src, size_t size);
#define LIO_SendData ASAP_ref(90, LIO_ERR (*const)(const void *, size_t))
/* Send size bytes from src, calls OSWriteLinkBlock as necessary to send all of data
 * return 1 on error
 */

LIO_ERR LIO_SendProduct(LIO_CTX *, BYTE);
#define LIO_SendProduct ASAP_ref(595, LIO_ERR (*const)(LIO_CTX *, BYTE))
/* called from VarLink */

LIO_ERR LIO_RecvData(void *dest, size_t size, DWORD WaitDelay);
#define LIO_RecvData ASAP_ref(91, LIO_ERR (*const)(void *, size_t, DWORD))
/* Registers LIO timer (3) for WaitDelay
 * Returns 0 if no error
 */

static LIO_ERR CheckOK(LIO_CTX *file);
/* return 0 if OK received OK */

static LIO_ERR SendPacket(BYTE Cmd, WORD PacketSize, BYTE DevType);

static WORD ConvertEndian(WORD w);

static WORD SumChk(const void *data, WORD size);
/* Calculate checksum */

static WORD GetLEWord(const WORD *w);
/* get (not necessarily aligned) word stored in Little Endian */

int OSCheckSilentLink( void );
#define OSCheckSilentLink ASAP_ref(586, int (*const)( void ))
/* Returns link state, from:
```

0	none
1	var header, ext var header
2	direct command, screen dump, request with size, request
3	is ready

```
*/

void OSLinkCmd(BOOL NormalState);
#define OSLinkCmd ASAP_ref(587, void (*const)(BOOL))
/* Called from ngetchx, kbhit, handles link commands for home screen etc.
```

```
NormalState set from BOOL QstandardState( void ) (event handler) */

static BOOL is_in_link_state( void );
/* EV_appA == AP_ALGEBRA && EV_appB == AP_NULL */

LIO_ERR LIO_Receive(LIO_CTX *, BYTE DevType, BOOL skipmuch);
#define LIO_Receive ASAP_ref(88, LIO_ERR (*const)(LIO_CTX *, BYTE, BOOL))

static LIO_ERR DirList(BYTE DevType, LIO_CTX *);

static SYM_ENTRY *MatchTypeNext(BYTE VarType, BYTE *foundVarType);
/* Finds Next Sym Matching VarType, if VarType==DIR_LIST Next (valid) Sym is returned
VarType is defined in enumeration LIO_VAR_CMD
Used to parse the folder tree in DirList
*/

static LIO_ERR RecursiveDirList(BYTE DevType, LIO_CTX *);

BYTE DevTypeValidate(BYTE DevType);
/* Returns 0 if not a valid type */

static BOOL push_varname(const char *s);

static LIO_PACKET HeaderPacket;

WORD OSLinkBusy;
LIO_PACKET OSLinkPacket;
```

TI-BASIC Interface (flink.o?)

```
void push_getkey( void );
#define push_getkey ASAP_ref(790, void (*const)( void ))
void cmd_getcalc(EStackIndex var);
#define cmd_getcalc ASAP_ref(849, void (*const)(EStackIndex))
void getcalc(EStackIndex var);
#define getcalc ASAP_ref(84, void (*const)(EStackIndex))
void cmd_get(EStackIndex var);
#define cmd_get ASAP_ref(848, void (*const)(EStackIndex))
void cmd_printobj(EStackIndex);
#define cmd_printobj ASAP_ref(880, void (*const)(EStackIndex))
void cmd_sendchat(EStackIndex var);
#define cmd_sendchat ASAP_ref(905, void (*const)(EStackIndex))
void cmd_sendcalc(EStackIndex var);
#define cmd_sendcalc ASAP_ref(904, void (*const)(EStackIndex))

static void send(EStackIndex var, BYTE DevType);

BOOL sendcalc(EStackIndex var, BOOL allowSysVars, BYTE DevType, BYTE *status);
#define sendcalc ASAP_ref(85, BOOL (*const)(EStackIndex, BOOL, BYTE, BYTE *))

void cmd_send(EStackIndex esi);
#define cmd_send ASAP_ref(903, void (*const)(EStackIndex))
```

System Timers (timer.o)

The system timer is maintained by auto-int 5 which by default runs at 20hz. System timers can be set to any multiple of this period. The system maintains 6 countdown timers and 2 vector timers (numbered 1 to 6). Although these work independently of each other, they are registered and used exclusively, e.g. you cannot register a vector timer if a countdown timer is already registered with the same ID.

So we can assume that we have 6 timers available (1 to 6), timers 1 & 2 can optionally register a vector to be called after the interval.

Space is reserved for 7 of each of the timer structures, but the extra timers are not used. Unused timers are marked with a starting value of –1.

The following table gives the default timer assignment. Only timer 6 is free for use in programs, although you may use the other timers if you restore the previous settings.

Hardware:
Given the processor running at 9.875 MHz, and the rate generator is incremented every 6250 clock cycles, set with an initial value 0xB2 (BYTE:0x600017) triggering auto-int 5 after 79 (0x100 - 0xB2 + 1) increments.

9 875 000 / 6250 / 79 = 20

These assumptions appear to be correct, corroborated by the "CyclePic" instruction which performs:

OSRegisterTimer(5, (WAIT > 16000 ? 16000 : WAIT)*20);

Purpose	ID	Type	Time
Battery Level Monitor	1	Vector	1 second
Auto Power Down	2	Countdown	5 mins
LIO (when active)	3	Countdown	1/5 second
Cursor	4	Countdown	½ second
System General Purpose	5	(Countdown)	N/A

Enumerations

```
enum {
  BATT_TIMER = 1,
  APD_TIMER = 2,
  LIO_TIMER = 3,
  CURSOR_TIMER = 4,
 };
```

```
typedef struct {
  DWORD Ticks;
  DWORD CurVal;
  void (*TimerProc)( void );
} VTIMER;

typedef struct {
  DWORD Ticks;
  DWORD CurVal;
} TIMER;
```

```
void OSTimersReset( void );
/* called on calculator reset */

WORD OSRegisterTimer(WORD num, DWORD ticks);
#define OSRegisterTimer ASAP_ref(240, WORD (*const)(WORD, DWORD))
/* registers countdown timer (num) */

WORD OSVRegisterTimer(WORD num, DWORD ticks, void (*callback)( void ));
#define OSVRegisterTimer ASAP_ref(644, WORD (*const)(WORD, DWORD, void (*)( void )))
/* registers vector timer (num) with callback
 * 0 if unable to register, i.e. if used or num out of range
 * note if timer set by OSRegisterTimer it cannot be used without freeing it
 */

BOOL OSFreeTimer(WORD num);
#define OSFreeTimer ASAP_ref(241, BOOL (*const)(WORD))
BOOL OSVFreeTimer(WORD num);
#define OSVFreeTimer ASAP_ref(645, BOOL (*const)(WORD))

//Timer checking functions:
DWORD OSTimerCurVal(WORD num);
#define OSTimerCurVal ASAP_ref(242, DWORD (*const)(WORD))
BOOL OSTimerExpired(WORD num);
```

```
#define OSTimerExpired ASAP_ref(243, BOOL (*const)(WORD))
DWORD OSTimerRestart(WORD num);
#define OSTimerRestart ASAP_ref(244, DWORD (*const)(WORD))

void timer_int( void ); /* auto-int 5 */
void OSRegisterAPDTimer( void );
```

Keyboard Handing

There are 3 object files listed here which handle keypresses at various different levels, they are as follows.

- Low Level
- High Level
- General Key Routines

Low Level (hkey.o)

This module uses OSKeyScan, and queues the keypresses. Delays are properly maintained and can be controlled from routines provided here.

Examples

	OS Default (OSqReset)	Geom Default	Geom Cursor
KeyInitDelay	0x150 (336)	0x8C (140)	0x32 (50)
KeyBetweenDelay	0x30 (48)	0x30 (48)	8
Approx Init Delay	1 second	½ second	1/8 second
Approx Repeat Rate	8	8	50

A slight bug: Note that the Geometry value overrides the OS Intended values in some cases, e.g. it sets its values when leaving geometry, and as a side effect of the event handler this is the value that dominates on a reset. When the calculator is turned “ON” OSInitKeyDelays is called, and the correct delays are set.

Hardware:
Given the processor running at 9.875 MHz, and the rate generator is incremented every 6250 clock cycles, triggering auto-int 1 every 4 increments.

9 875 000 / 6250 / 4 = 395

```
#define KEY_QUEUESIZE 32

QUEUE(KEY_QUEUESIZE) KEY_QUEUE;
```

```
enum KeyModifiers { KBM_2ND=0x1000, KBM_DIAMOND=0x2000, KBM_SHIFT=0x4000,
KBM_HAND=0x8000, KBM_ALPHA=0x8000 };

enum { KB_SCANCODE = 0x800 } ; /* Scancode sent as keypress, OSKeyPhase = 2 */

typedef WORD KEYCODE;
/*
typedef struct {
  int Mod : 5;
  int Key :11;
} KEYCODE;
*/
```

```
/* ScanCode to KeyCode tables */
const WORD LowerKeys[81]; /* I won't bore you with the details! */
const WORD UpperKeys[81];
const WORD _2ndKeys[81];
const WORD LowerGreek[26];
const WORD UpperGreek[26];
```

```
void keytimer_int( void ); /* auto-int 1 */
```

```
staticKEYCODE ScanConvert(WORD ScanCode);
/* Returns keycode to send */
```

```
static KEYCODE KeyConvert(KEYCODE KeyCode);
/* Maps KeyCode to special characters etc */

static KEYCODE _ngetchx( void ) { ngetchx(); }

void OSInitKeyDelays( void );

WORD OSInitKeyInitDelay(WORD delay);
#define OSInitKeyInitDelay ASAP_ref(584, WORD (*const)(WORD))
WORD OSInitBetweenKeyDelay(WORD delay);
#define OSInitBetweenKeyDelay ASAP_ref(585, WORD (*const)(WORD))

static void showModKey(BOOL showMod);
/* Updates modifiers on status line, calling ST_modKey
if (showMod == FALSE) modifier is erased
*/

static void LogKey(KEYCODE KeyCode);
/* KeyCodes are sent to here before Queueing, so probably just for debugging purposes
static void LogKey(KEYCODE KeyCode) { *((WORD*)(0xC00000)) = KeyCode; }
*/

static WORD CtrlFlags;
static WORD KeyCode;
static WORD CurKeyDelay;
static WORD Count;
static WORD OldMod, OldScan;
static WORD Modifiers;
static WORD SpecialKey; /* last special key e.g. diamond-G */
static BYTE LockFlag; /* 1 if Lock is on */
static BYTE Flags;

WORD OSScanCode; /* scanned from bottom left, i.e. = (9-row)*8+col */
WORD OSModifier;
BYTE OSKeyPhase; /* Used to get control over keypresses (Trace, Geom) */
WORD OSKeyInitDelay;
WORD OSBetweenKeyDelay;
KEY_QUEUE OSKeyQueue;
BOOL OSKeyDown;
```

The scancode conversions give the following results.
(without any modification, i.e. shift does not subtract 'a'-'A' capital letter goes through)

Code	Nmenonic
00D	KB_ENTER
020	KB_SPACE
028	(
029	)
02A	*
02B	+
02C	,
02D	-
02E	.
02F	/
030-03A	0-9
3C	<
3D	=
3E	>
041-05A	KB_A - KB_Z
05E	^
061-0x7A	(lower-case mapped into here)

088	_THETA_
0AD	(-) minus
101	KB_DEL <-
102	KB_STO
103	KB_SIN
104	KB_COS
105	KB_TAN
106	KB_LN
107	KB_CLEAR
108	KB_ESC
109	KB_APPS
10A	KB_MODE
10B	KB_ON
10C-113	KB_F1-KB_F8
114	(-) ANS
151	KB_LEFT (scancode 0x51)
152	KB_UP
154	KB_RIGHT
158	KB_DOWN
15A	KB_CAPS

High Level (nkey.o)

```
void pushkey(KEYCODE code);
#define pushkey ASAP_ref(80, void (*const)(KEYCODE))

KEYCODE ngetchx( void );
#define ngetchx ASAP_ref(81, KEYCODE (*const)( void ))

void OSAlexOut(const char *text);
```

/* Output text to linkport, used in AnsOut (Applications

- Home
- Y= Editor
- Window
- Graph
- Table
- Data/Matrix Editor
- Geometry
- Text
- Numeric Solver
- Self-Test

Home

- Algebra Application
- Home Stack

```
Algebra Application (apalgebra.o))
trailing zero is not sent, -1 (0xFF) is sent instead
*/

KEYCODE kbhit( void );
#define kbhit ASAP_ref(82, KEYCODE (*const)( void ))

static WORD SRold; //OSAlexOut
WORD ScanFlag;
```

General Key Routines (gkey.o)

```
const WORD ModeKeys[14]; /* e.g diam-q home etc. */
const WORD SysKeys[5]; /* e.g. catalog */

WORD GKeyIn(SCR_RECT *Cursor, WORD Flags);
#define GKeyIn ASAP_ref(382, WORD (*const)(SCR_RECT *, WORD))
WORD GKeyDown( void );
#define GKeyDown ASAP_ref(383, WORD (*const)( void ))
void HelpKeys( void );
#define HelpKeys ASAP_ref(385, void (*const)( void ))
/* Displays the HelpKeys window, showing extra modifiers etc. */

BOOL QModeKey(KEYCODE Key);
#define QModeKey ASAP_ref(386, BOOL (*const)(KEYCODE))
BOOL QSysKey(KEYCODE Key);
#define QSysKey ASAP_ref(387, BOOL (*const)(KEYCODE))
void GKeyFlush( void );
#define GKeyFlush ASAP_ref(384, void (*const)( void ))
BOOL WordInList(WORD Find, WORD *List);
#define WordInList ASAP_ref(388, BOOL (*const)(WORD, WORD *))
/* returns TURE if Find is present in zero terminated WORD array List */

BYTE GOK_Flag;
```

Flash Memory

Here is a general memory map of the 92+ calculators:

```
0x000000
  vector table (auto-int 7 protects this, throws error:
      Memory: if caused by stack overflow (usp<=0x400)
      Protected Memory Violation: otherwise
0x000400
  RAM
0x040000
  alternative access points of RAM
0x200000
  internal rom layout, same as external rom
0x400000
  //boot sector
  DWORD Vectors[0x100];
  void (*downloadcert)( void );
  ParmBlock* pBaseParmBlock;
  ParmBlock BaseParmBlock;

  //code consisting of init stuff + a few of the standard modules:
  //abs, mem???, bignum, MD5, glib (note a few routines removed)
0x410000
  certificate
0x412000
  flash reset code
0x418000
  reserved certificate region?
0x4A0000
  reserved flash reset code region?
0x420000
  base code
0x590000
  flash memory on HW1 calcs, Hardware version 2 uses max available?
0x600000
  ports
0x700000
  HW2 extra ports?
```

- EEPROM Programming
- Memory Protection
- Flash Memory Routines

- Extended Memory Management

EEPROM Programming

Writing to Flash EEPROMs is much slower than writing to RAM, and requires writing control codes to the address space of the chip to perform these operations. Apparently the Flash EEPROM used is the "SHARP LH28F160S3-L", which has many features (some may be disabled) including writing (multiple) words/bytes, erasing 64kb blocks, and locking of blocks.

For more information on this chip see Johan Borg's file at http://d188.ryd.student.liu.se/ftp/calculator/ti89/tech/flashrom.txt, he also has put up the datasheet at ftp://d188.ryd.student.liu.se/calculator/ti89/tech/f160s3.pdf which is worth a look.

Fortunately only a subset of this chips features are used; writing words and erasing blocks. Write operations take 12.95 micro-seconds, read operations 100ns, so you can see why a special mode of operation is required.

To program the EEPROM we need to enter a special mode by writing certain control codes to the address range. Two operations can be perfored, erase (64kb) sector, and write a word. In this write mode reading from the address range gives the status register of the EEPROM, generally the only value that is used is that bit 7 of the read word will be set when complete. After the operations are complete Read mode is then set by writing another control code.

The drawback of this method is that the ROM cannot be read whilst write operations are being performed. Therefore the writing code must execute from the RAM. Trap 11 handles all of this on the calculator, copying the relevent portions to a buffer then executing them.

The following subset of operations is used:

Code	Function
0x5050	Clear Status Register
0x1010	Write Setup (next word will be written)
0x2020	Erase Setup
0xD0D0	Erase Conform (address)
0xFFFF	Read Memory

Erasing and writing require to writes, e.g. 0x2020, 0xD0D0 to perform erase to prevent accidental operations.

Note
The flash memory is, by default, write protected by the system. And can only be written to under special circumstances. See Memory Protection.

Examples

```
;erase 64kb block in which ERASE_ADDR resides
      lea         (ERASE_ADDR),a2
      move.w      #0x5050,(a2)      ;Clear Status Register
      move.w      #0x2020,(a2)      ;Erase Setup
      move.w      #0xD0D0,(a2)      ;Erase Confirm
write_state_busy:
      move.w      (a2),d0           ;Read Status Register
      btst        #7,d0             ;1=Ready
      beq         write_state_busy
      move.w      #0xFFFF,(a2)      ;Read
```

```
;write VALUE to WRITE_ADDR
      lea         (WRITE_ADDR),a2
      move.w      #0x5050,(a2)      ;Clear Status Register
      move.w      #0x1010,(a2)      ;Write Setup
      move.w      #VALUE,(a2)       ;Erase Confirm
write_state_busy:
      move.w      (a2),d0           ;Read Status Register
      btst        #7,d0             ;1=Ready
      beq         write_state_busy
```

```
      move.w      #0xFFFF,(a2)      ;Read
```

Memory Protection

(Internal ROM addresses)

Further, the flash memory is protected by another chip, demonstrating the use of intelligent memory. This controls read and write access to the flash memory (Intelligent memory perfors operations other than simple read and write cycles). This can be used to prevent accidental programing (of the flash memory) and erasure during a faulty power-up or power-down sequence. It is also used to "hide" the certificate.

Reading and writing values in the range 0x040000-0x200000 (i.e. address space that maps to the RAM) in particular to the most significant word in the RAM performs (address&0x1F00000) sets the flash into special states. e.g. reading a word from (0x1C????) write protects the entire flash, and read protects the certificate region (0x210000-0x211FFF) [and the secondary certificate region 0x218000-0x219FFFF]. However, this only works if this is done from code in the boot sector (0x200000-0x20FFFF), the base setup sector (0x212000-0x217FFF) [or the secondary base setup sector (0x21A000-0x21FFFF). Writing a word to (0x1C????) will disable write protection and read protection of the regions.

Since writing in this range (0x040000-0x200000) will affect a word in the RAM, an arbitrary address is set aside for this purpose by the OS. Usually the boot code uses 0x004400, and the base (main) code uses 0x005E00.

Trap 11 does all flash operations in the base code. Writing to (0x1C5E00) on entry – disabling write and read protection, copying the programming code to RAM and executing it, then reading from (0x1C5E00) to protect the memory again.

This means that it is, at least, difficult to get read access to the certificate as we cannot run our own code in these regions without a certified ROM image. On AMS2 Trap 11 contains a lot more code, so it should be easier to get around this, however, it is very difficult on previous versions, and may only be possible with careful use of trace.

Known uses of intelligent memory:

Flags	address	Purpose
0x18	185E00	low power state for flash (or protect ?)
0x1C	1C5E00	write protect state for entire flash read protect state for certificate regions

Examples

Changing power state: (0x18)

```
      move.w      ($185E00),d1
      move.b      d0,($600005)      ; * set power mode (IDLE=0x10, OFF=0x00, …)
      nop
      move.w      d1,($185E00)
      rts
```

Note: This is just done inline with the (*) line on non-flash calculators. This code is copied to RAM before execution as with all code that deals with non-read operations on the flash memory.

```
//eg. HW 2 {42, 1, 1, 1, 1, 0, 2, 0xF0, 0x80, 0xF0, 0x80)
struct ParmBlock {
  WORD size;      //size of Parms (20, 42 on AMS2)
  DWORD Parm[1];  //should be size/4
};

struct FL_table {
  void (*pf)( void );
  WORD flen;
};
```

Flash Memory Routines (flash.o)

This code is based in the first sector.

```
void _flash_int( void ); /* Trap #11 */
/* d3 = func number
0: write
1: erase
2: addcert
3: getcert
4: cgetvernum
if (flen) function is copied and executed in RAM.
d4 = size
a2 = addr1
a3 = addr2
*/

WORD FL_addCert(void *src, size_t size);
#define FL_addCert ASAP_ref(361, WORD (*const)(void *, size_t))
static WORD _addcert( void );

void FL_download(DWORD param);
#define FL_download ASAP_ref(362, void (*const)(DWORD))
/* reset to base code  (reinstate vector table and reset) (param ignored) */

void FL_erase(const void *addr);
static void _eraseR( void );

void FL_getCert(HANDLE *hDest, size_t *len, BOOL decrypt);
#define FL_getCert ASAP_ref(364, void (*const)(HANDLE *, size_t *, BOOL))
static void _getcert( void );

ParmBlock *FL_getHardwareParmBlock( void );
#define FL_getHardwareParmBlock ASAP_ref(363, ParmBlock *(*const)( void ))
/* returns ParmBlock from boot code if found, else DefParmBlock */

const ParmBlock DefParmBlock = { 20, 1, 0, 1, 1, 0 };

WORD FL_getVerNum( void );
#define FL_getVerNum ASAP_ref(365, WORD (*const)( void ))
/* calls cgetvernum */

void _writecert(const void *src, void *dest, size_t size);
/* _writecert asm func, copies writecertR to RAM and executes */
static void writecertR(const void *src, void *dest, size_t size);
/* this is similar to write, although doesn't enter write mode first!
this may be why certificates don't work under VTI */

void _movecert(void *src);
static void movecertR(void *src);
/* copies 64kb (0x8000 words) from src to 0x410000 and erases old one */

void void _reboot( void );
static void rebootR( void );
/* clears all extended ROM code (to 0s), and then resets the boot code */

void FL_write(const void *src, void *dest, size_t size);
#define FL_write ASAP_ref(369, void (*const)(const void *, void *, size_t))
static void _writeR( void );


DWORD FL_oldVectors[0x40];

union {
  WORD Val; /* = 0x4E72+1 (prevent optimisation?) */
  void Exec( void ); /* would this work? */
} rte;

union {
```

```
  WORD Val; /* = 0x4E74+1 */
  void Exec( void );
} rts;

WORD FL_access; /* arbitrary access point for (0x18???? and 0x1C????).w */

union { /* buffer for flash functions to execute from RAM */
  WORD Buf[0x40];
  void Func( void );
} FL_code;
```

Extended Memory Management (em.o)

This code handles writing to the user flash memory region. Which on HW1 calcs is in the range 0x590000-0x600000, on HW2 it starts at the end of the ROM code. Verification code lies here mainly, all writes have to go through the flash (FL) code.

```
enum EM_type { EM_ERASED = -1, EM_USED = -2, EM_FREE = -4 };

struct FlashBlock {
  WORD EM_type;
  WORD Handle_Link;     //Word Length | H_LOCKED
  WORD Data[1];         //Handle points here
};
```

```
typedef struct {
  void *Pos
  void *Start;
  HANDLE h;
EM_FILE;
```

```
void EM_abandon(HANDLE h);
#define EM_abandon ASAP_ref(347, void (*const)(HANDLE))
/* abandon a memory block allocated with handle entry h
Clears entry in heap table, and precedes EM block with (-4)
*/

void EM_blockErase(WORD offset);
#define EM_blockErase ASAP_ref(348, void (*const)(WORD))
/* Erase 68kb block at offset (sets to -1) throws error if out of range
Block is first checked with EM_blockVerifyErase, then FL_erase is called if necessary
*/

BOOL EM_blockValid (const void *flash);
/* if first word -1, or -2 && rest all other words in 64kb block are -1 */

BOOL EM_blockVerifyErase(WORD offset);
#define EM_blockVerifyErase ASAP_ref(349, BOOL (*const)(WORD))
/* Returns if block at offset in EM has been erased (-1)
*/

void EM_delete(EM_FILE *file);
#define EM_delete ASAP_ref(350, void (*const)(EM_FILE *))
/* like EM_abandon, sets word at file.Start to -4. */

void *EM_findEmptySlot(size_t size);
#define EM_findEmptySlot ASAP_ref(351, void *(*const)(size_t))

BOOL EM_GC(BOOL allowDialog);
#define EM_GC ASAP_ref(352, BOOL (*const)(BOOL))
/* Garbage Collect
returns whether or not GC occurred */

void EM_Init( void );

BOOL EM_moveSymFromExtMem(const EStackIndex *name, HSYM hSym);
#define EM_moveSymFromExtMem ASAP_ref(353, BOOL (*const)(const EStackIndex *, HSYM))
/* moves either Sym, referenced by name or hSym (name==NULL), to RAM
```

```
returns success
*/

BOOL EM_moveSymToExtMem(const EStackIndex *name, HSYM hSym);
#define EM_moveSymToExtMem ASAP_ref(354, BOOL (*const)(const EStackIndex *, HSYM))
/* moves either Sym, referenced by name or hSym (name==NULL), to ExtMem
returns success
*/

BOOL EM_open(EM_FILE *file, size_t size);
#define EM_open ASAP_ref(355, BOOL (*const)(EM_FILE *, size_t))
/* write first var of given size to empty slot, filling in file structure
Handle of empty slot is returned, header is at -4(value)
returns success
*/

void EM_put(EM_FILE *file, const void *src, size_t size);
#define EM_put ASAP_ref(356, void (*const)(EM_FILE *, const void *, size_t))
/* Writes src and updates file (moves to next location) */

void EM_survey(void *d1, void *d2);
#define EM_survey ASAP_ref(357, void (*const)(void *, void *))

HSYM EM_twinSymFromExtMem(const EStackIndex *name, HSYM hSym);
#define EM_twinSymFromExtMem ASAP_ref(358, HSYM (*const)(const EStackIndex *, HSYM))
/* SymAddTwin & copy to RAM
if name==NULL use hSym instead
*/

void EM_write(const void *src, void *dest, size_t size);
#define EM_write ASAP_ref(359, void (*const)(const void *, void *, size_t))
/* throws error if dest out of range */

void EM_writeToExtMem(const void *src, size_t size, BOOL *success ,void *dest);
#define EM_writeToExtMem ASAP_ref(360, void (*const)(const void *, size_t, BOOL *,void
*))
/* -2, & (locked) handle header is written before src
written to any empty slot -> dest. EM_GC may occur
*/

static void EM_cmd(EStackIndex esi, BOOL bArchive);
void cmd_archive(EStackIndex esi);
void cmd_unarchive(EStackIndex esi);

BYTE EM_twinDelete; //.0
```

Assembler support (exec.o)

```
typedef FLOAT BCD16;

void cmd_exec( void );
/* TI-BASIC processing of Exec STRING [,EXPR,...]
Parses directly from top_estack
*/

EStackIndex EX_getArg(short n);
#define EX_getArg ASAP_ref(190, EStackIndex (*const)(short))
/* return EStackIndex of arg n (0, 1, ...) */

void EX_patch(void* prog_start, EStackIndex esi);
#define EX_patch ASAP_ref(346, void (*const)(void*, EStackIndex))
/* relocate fixup program at base prog_start, relocs at esi (end of prog) */

short EX_getBCD(short n, BCD16 *bcd);
#define EX_getBCD ASAP_ref(191, short (*const)(short, BCD16 *))
/* get BCD (from EX_getArg(n)) and store to *bcd
return BOOL success
*/
```

```
void EX_stoBCD(unsigned char *name, BCD16 *bcd);
#define EX_stoBCD ASAP_ref(192, void (*const)(unsigned char *, BCD16 *))
/* store bcd to var name */

BCD16 bcdadd(BCD16 f1, BCD16 f2);
#define bcdadd ASAP_ref(182, BCD16 (*const)(BCD16, BCD16))
BCD16 bcdsub(BCD16 f1, BCD16 f2;
#define bcdsub ASAP_ref(183, BCD16 (*const)(BCD16, BCD16))
BCD16 bcdmul(BCD16 f1, BCD16 f2);
#define bcdmul ASAP_ref(184, BCD16 (*const)(BCD16, BCD16))
BCD16 bcddiv(BCD16 f1, BCD16 f2;
#define bcddiv ASAP_ref(185, BCD16 (*const)(BCD16, BCD16))
BCD16 bcdneg(BCD16 f);
#define bcdneg ASAP_ref(186, BCD16 (*const)(BCD16))
BOOL bcdcmp(BCD16 f1, BCD16 f2);
#define bcdcmp ASAP_ref(187, BOOL (*const)(BCD16, BCD16))
SLONG bcdlong(BCD16 f);
#define bcdlong ASAP_ref(188, SLONG (*const)(BCD16))
BCD16 bcdbcd(SLONG l);
#define bcdbcd ASAP_ref(189, BCD16 (*const)(SLONG))

const LONG ProcTableCount;
const LONG ProcTable[];
/* On AMS1 this has about 972 entries, on AMS2 there are 1463 - so it should be
interesting to see the new exec.inc */

Note: structures (like BCD16) are stored at (a6) - the return register
In parameters, they are copied directly as expected.
```

ASM format

```
packed struct ASM_VAR {
  BYTE Code[]; //prog base & start at $02! (after len word)
  NA_WORD 0;
  packed struct {
    NA_WORD fileOffset;
    NA_WORD destOffset;
   } Relocs [];
  token $F3;
};
```

```
//exec.o
void EX_Patch(void* prog_start, EStackIndex esi);
//basic.o
run_user_func(EStackIndex esi, BOOL start, BOOL *twinDelete)
{
  //...
  void (*base)( void );
  EStackIndex Oldtop = top_estack; //a3

  HSYM hsym;             // = VarRecall(...)
  SYM_ENTRY* funcSym;    // = DerefSym(hsym)
  HANDLE h;              // = funcSym->hVal
  EStackIndex funcEsi;   // = HToEsi(h)

  // ... get all vars for our function execute ...

  if(ESTACK(funcESI) == ASM_TAG) {
    if(HeapGetLock(h)) ER_throw(ER_INVALID_PROGRAM_REFERENCE);
    HeapLock(h);
    base = (void *()) DEREF(h)+2;
    EX_Patch( (void *)base, funcESI ); //relocate
    base();
    top_estack = Oldtop;
    HeapUnLock(h);
    if(DerefSym(hsym)->Flags & SF_TWIN) EM_twinDelete = TRUE;
```

```
    return FALSE;
  } else if (ESTACK(funcESI) == FUNC_TAG) {
     // tokenises if necessary, pops parameters to a new temporary folder, & sets return
flags at end
  } else
    ER_throw(ER_NOT_FUNC_OR_PRGM);
  //...
}
```

Notes:
You can obviously examine the estack as you wish in your program, all the paramaters are pushed as expected. There number of paramaters is variable so END_TAG marks the end.

When running this code we are under 2 error handlers (ER_catch), one in home, the other in the interpreter. It wasn't intended for ASM programs to bail out to the TIOS or to return values to the home screen, however, both can be acheived:

1. If you want to return a value to the OS the best way is to alter the instructions that will be interpreted after your program, i.e. change the pointer to the current instruction to what you want to return. (remember ENDSTACK_TAG)
2. An ER_Throw will return you to the system as expected, but your program will remain locked, and will throw "ER_INVALID_PROGRAM_REFERENCE" on subsequent calls. Unlock your program just before you do your ER_throw will solve this (just a matter of clearing a flag - no bad consequences). You should watch what routines you call when the program is not locked, that is why it is best to do this at the end.

Dynamic Memory Allocation (heap.o)

Usual situaution after reset:

Description	!PLUS	PLUS
Null (0xFFFFFFFF)	N/A	$0
Heap markers (size 1)	0-4	1-4
temporary SYMs	5-7	N/A
TASK HANDLES (AMS2) [allocated high]		
BITMAP ERD Saved Screen (Error Dialogs (erdialog.o))	$9	$5

### *estack (Units* 	
• TI-BASIC Interface
• Dialog Handler
 	
TI-BASIC Interface (units.o)
`init_unit_system`
`#define init_unit_system ASAP_ref(944, ASAP_UNDEF)`
`push_unit_system_list`
`#define push_unit_system_list ASAP_ref(947, ASAP_UNDEF)`
`setup_unit_system`
`#define setup_unit_system ASAP_ref(948, ASAP_UNDEF)`
`has_unit_base`
`#define has_unit_base ASAP_ref(943, ASAP_UNDEF)`
`is_units_term`
`#define is_units_term ASAP_ref(945, ASAP_UNDEF)`
`did_push_divide_units`
`#define did_push_divide_units ASAP_ref(942, ASAP_UNDEF)`
`push_auto_units_conversion`
`#define push_auto_units_conversion ASAP_ref(946, ASAP_UNDEF)`
 	
Dialog Handler (wcmd.o?)
Just some extra Instruction tokens and keypresses on the PLUS machines
Note this really shouldn'y belong here. This provides some extra functions on PLUS machines, but cmd_customon and off just call external routines.
 	
`void UnitsDialog( void );`
`/* called via diamond-p */`
 	
`WORD Units_callBack(WORD a, DWORD b);`
 	
`static void CheckUnits(BOOL checkbase);`
 	
`void cmd_custmon( void );`
`#define cmd_custmon ASAP_ref(822, void (*const)( void ))`
`void cmd_custmoff( void );`
`#define cmd_custmoff ASAP_ref(821, void (*const)( void ))`
 	
EOS - The Expression Stack)
	$8	$6

(TEXT) clipboard (Clipboard (clipboard.o))	$A	$7
SYM_TABLE folder (Symbol Tables (sym.o))	$B	$8
SYM_TABLE main (Symbol Tables (sym.o))	$C	$9
SysVarSym (Variable Interface (store.o))	$D	$A
TEXT HOME entry	$E	$B
BITMAP PrgmIO	$F	$C
Plots (see Plots (gstat.o))	$10	$D
STORE LIST tblInput1	$11	$E
STORE LIST tblInput2	$12	$F
BITMAP Graph1	$13	$10
BITMAP Graph2 (if present)	($14)	($11)
### *MENU_STATE Current Menu States (Menus* • Standard Routines • Custom Menus • Variable Dialogs Standard Routines (menu.o))		

```
typedef unsigned short HANDLE;
#define H_NULL (HANDLE)0
```

```
struct HeapBlock {
  BOOL bLocked :1;
  WORD Link :15;
};

void * HeapEnd( void );
#define HeapEnd ASAP_ref(161, void * (*const)( void ))

void * HeapAllocPtr( DWORD Hlen) );
#define HeapAllocPtr ASAP_ref(162, void * (*const)( DWORD ))
/* Allocates Hlen bytes, stores handle information in block
handle is stored in first word of block, address of block+2 is returned
*/

void HeapFreePtr( void * Ptr);
#define HeapFreePtr ASAP_ref(163, void (*const)( void * ))
/* Frees a handle allocated with HeapAllocPtr
*/

DWORD HeapAvail( void );
#define HeapAvail ASAP_ref(143, DWORD (*const)(void))
/* returns total bytes available on heap
*/

DWORD HeapMax( void );
#define HeapMax ASAP_ref(156, DWORD (*const)(void))
/* returns maximum block size that can be allocated
*/

void HeapReset( void );

WORD HeapSize( HANDLE Handle );
#define HeapSize ASAP_ref(158, DWORD (*const)(HANDLE))
/* Returns total bytes taken up by Handle
*/

void HeapFreeIndir( HANDLE *PtrHandle );
#define HeapFreeIndir ASAP_ref(152, void (*const)(HANDLE *))
/* if PtrHanle != H_NULL, Frees handle pointed to by PtrHandle
*/

void HeapFree( HANDLE Handle );
#define HeapFree ASAP_ref(151, void (*const)(HANDLE))
/* Frees memory block given handle
*/

HANDLE HeapAlloc( DWORD Hlen );
#define HeapAlloc ASAP_ref(144, HANDLE (*const)(DWORD))

BOOL HeapResizeESTACK( WORD NewSize );
/* returns TRUE if successful in resizing estack
*/

HANDLE HeapAllocESTACK( DWORD Hlen);
#define HeapAllocESTACK ASAP_ref(145, HANDLE (*const)(DWORD))
/* allocates memory block, reducing the size of the estack if necessary
*/

HANDLE HeapAllocHigh( DWORD Hlen );
#define HeapAllocHigh ASAP_ref(146, HANDLE (*const)(DWORD))
HANDLE HeapAllocThrow( DWORD Hlen );
#define HeapAllocThrow ASAP_ref(147, HANDLE (*const)(DWORD))
HANDLE HeapAllocHighThrow( DWORD Hlen );
#define HeapAllocHighThrow ASAP_ref(148, HANDLE (*const)(DWORD))

HANDLE HeapPtrToHandle( void * Ptr );
#define HeapPtrToHandle ASAP_ref(570, HANDLE (*const)( void *))
/* Searches for handle of block at Ptr
```

```
H_NULL is returned if not found
*/

HANDLE HeapRealloc( HANDLE Handle, DWORD NewHsize );
#define HeapRealloc ASAP_ref(157, HANDLE (*const)(HANDLE,DWORD))

void * HLock( HANDLE h );
#define HLock ASAP_ref(153, void * (*const)(HANDLE))
/* Lock and dereference a handle */

HANDLE HeapLock( HANDLE Handle );
#define HeapLock ASAP_ref(154, HANDLE (*const)(HANDLE))
/* Lock a handle */

BOOL HeapGetLock( HANDLE handle );
#define HeapGetLock ASAP_ref(155, BOOL (*const)(HANDLE))
/* Returns lock state of handle */

HANDLE HeapUnlock( HANDLE Handle );
#define HeapUnlock ASAP_ref(159, HANDLE (*const)(HANDLE))

HANDLE HeapMoveHigh( HANDLE h );
#define HeapMoveHigh ASAP_ref(160, HANDLE (*const)(HANDLE))
/* try to reallocate h in high memory
returns H_NULL if unsuccessful
*/

void * HeapDeref(HANDLE h);
#define HeapDeref ASAP_ref(150, void * (*const)(HANDLE))
/* Dereference a handle */

static DWORD _HeapAvail(BOOL bTotal);
/* Calculate free mem for HeapAvail and HeapMax
returns number of bytes found
*/

static DWORD AdjustedBlockSize( DWORD Hlen);
/* Given a requested size get size to actually allocate
Used internally to make sure block is even size, and min of 8 bytes
*/

static void * ReallocHigh( DWORD curWordSize, void *blockStart, WORD Flags );
static HeapReduce( void * blockStart );

void HeapCompress( void );
#define HeapCompress ASAP_ref(149, void (*const)(void))
/* perform memory compaction */

HANDLE HeapGetHandle(void const *);
#define HeapGetHandle ASAP_ref(569, HANDLE (*const)(void const *))
/* Get next available handle */

static BOOL MoveHigh( void *blockHdr, WORD Flags);

static HANDLE _HeapAlloc( DWORD Hlen, BOOL High );
/* processes all allocations */

void NeedStack( short size );
#define NeedStack ASAP_ref(164, void (*const)(short))
/* Check a stack frame of given size can be created
Throws ER_MEMORY and resets HeapAllocSysFlag otherwise
*/

short FreeHandles( void );
#define FreeHandles ASAP_ref(571, short (*const)(void))
/* returns number of free handles */

Now able to understand what all handles refer to:
Not definate handles (may change in certain situations)
```

```
Should all be accessed through memory values.
```

```
static WORD OldSizeESTACK;
static BYTE MarkerAllocFlags[5];
static WORD ResizeCountESTACK;
static BOOL bHeapInit;
static void* ?after_heap;
static void* ?top_heap;
static void* ?max_top_heap;
```

```
BYTE HeapAllocSysFlag;
LONG HeapTotalAvail;
void **Heap;
```

Error Handling

- Low Level Implementation
- Error Dialogs

Low Level Implementation (error.o)

The error handling mechanism in C is almost identical to that used in TI-BASIC. Although it is most probably implemented as a compiler extension (as with the floating point package), I have defined a few macros here which will have the same effect.

```
typedef WORD ER_CODE;

typedef struct ErrorStruct {
  DWORD Reg[11]; //d3-d7/a2-a7
  DWORD NG_Control;
  EStackIndex RetIndex; //?
  void* pc;
  struct ErrorStruct *Prev;
} ER_FRAME;

//0xAeee: A line emulator throws error eeee
//a68k does not cope with "dc.w #errCode" properly
//#define ER_throw(x) asm("dc.w %0" : : "g"(x|0xA000))
//Use slower function call instead
#define ER_throw(x) ER_throwVar(x)

#define TRY { ER_CODE errCode; ER_FRAME errFrame; if(!(errCode = ER_catch(&errFrame))) {
#define ONERR ER_success(); } else {
#define ENDTRY }}
#define PASS ER_throwVar(errCode)

#define ER_STOP          2
#define ER_DIMENSION   230
#define ER_MEMORY      670
#define ER_MEMORY_DML  810
//Add your own codes here!
```

```
void ER_throwVar(ER_CODE error);
#define ER_throwVar ASAP_ref(339, void (*const)(ER_CODE))
/* throw a non-const error code, see PASS */

WORD ER_catch(ER_FRAME *frame);
#define ER_catch ASAP_ref(340, WORD (*const)(ER_FRAME *))
/* setup an error handler, see TRY */

void ER_success( void );
#define ER_success ASAP_ref(341, void (*const)( void ))
/* removes error frame from linked list, see ONERR */

void ER_Init( void );
```

Data, see OS Hardware Interface

- OS Variables and Initialisation
- General OS Routines
- Queues (IO Buffering)
- Link Control
- System Timers
- Keyboard Handling

```
OS Variables and Initialisation (init.o)
```

Example

```
TRY
      //protected code
ONERR
      //error handler
      if(errCode == ER_BREAK)
            PASS; //pass on any unhandled errors to a higher level
ENDTRY
```

Error Dialogs (erdialog.o)

```
void ERD_alloc( void );
/*
ERD screen backup is usually a 164*60 BITMAP on the TI92+ and is created automatically by
WinOpen (WF_TRY_SAVE_SCR) when not used the handle is of size 3794 large enough for
240*126 (10 bytes left over)
*/

KEYCODE ERD_dialog(ER_CODE err, BOOL program);
#define ERD_dialog ASAP_ref(337, KEYCODE (*const)(ER_CODE, BOOL))
/*  Displays the error dialog for err
if program is set then the GOTO button is displayed instead of OK
*/

static void ERD_EventHandler(EVENT* ev);
/* event handler for dialog */

void ERD_process(ER_CODE err);
#define ERD_process ASAP_ref(338, void (*const)(ER_CODE))
/* Processes the error code in err
calls ERD_dialog appropriately, and responds (by starting apps)
*/

static HSYM ERD_getFunc(BOOL* home);
/* get the HSYM of the func which cause the error
home is TRUE if redirecting to home, FALSE to program
*/

static HANDLE hSaveScr;

HSYM Func;
BOOL bActive;

BITMAP SaveScr;


static void ERD_EventHandler(EVENT* ev)
{
  if (ev->Type == CM_KEYPRESS)
    switch( ev->Key.Code )
    {
    case KB_ENTER:
```

```
      ER_throw(1); // Exit from event loop: Button 1
    case KB_ESC:
      ER_throw(2); // Button 2
    case KB_OFF:
       EV_defaultHandler(ev); // Allow the machine to be switched off
    default:
       break;
    }
}

BOOL ERD_dialog(WORD err, BOOL gototype)
{
  EV_captureEvents(&ERD_handleEvent);
  //…
  TRY
    EV_eventLoop();
  ONERR
    switch(errCode)
    {
    case 1:
      //button 1
      break;
    case 2;
      //button 2
    default:
      break;
    }
  ENDTRY
}
```

ER_throw is not usually used in event handlers, however it can be used as expected. Values <8 are ignored so as in the above example can be used to as you like. ERD_process calls ERD_dialog under an error handler so that ErrorCode is the resulting button value.

Certificates and Authentication

Although the actual certificate data is protected (see Flash) the general protection method used may be of some interest. PGP techniques are used to authenticate (digitally sign) ROM images (and flash applications).

The principal ideas behind this process are as follows. The RSA algorithm uses two keys, one is public and one private. Data encrypted with one of these keys can only be decrypted with the other key. So given the procedure is secure, data encryped with the public key can only be decrypted with the private key, and valid data decrypted with the public key could only be produced by the private key.

Message digests create a unique number for given data. By sending a message digest encrypted with the private key, along with the original message, it is possible to check that the message came from the correct source. This is done by generating the message digest, and comparing it against the encryted version sent along with the message (decrypted with the public key). If they match then this "authenticates" the message.

- Big Number Maths
- Message Digests
- Certificates

Big Number Maths (bignum.o)

These functions implement RSA decryption used for authenticating the certificate. The code he is of no interest, and is best ignored.

General use is power17Mod, decryption with public key (n, 17) where n is a 512 bit (64 byte) key, given at the beginning of the certificate code, although others may (or may not) be used. n=p*q, (p-1)*(q-1) relatively prime to 17

```
/*
typedef struct {
  BYTE Len;
```

```
  BYTE Data[1];   //Should be []
} BN;
*/
typedef BYTE BN;
```

```
void BN_power17Mod(BN *dest, const BN *x, const BN *n);
#define BN_power17Mod ASAP_ref(290, void (*const)(BN *, const BN *, const BN *))

void BN_prodMod(BN *dest, const BN *b, const BN *n);
#define BN_prodMod ASAP_ref(292, void (*const)(BN *, const BN *, const BN *))
static void _BN_copy( void ); //{a1}->{a0}

void BN_powerMod(BN *dest, const BN *x, const BN *e, const BN *n);
#define BN_powerMod ASAP_ref(291, void (*const)(BN *, const BN *, const BN *, const BN
*))
```

```
void BN_power17Mod(BN *dest, const BN *x, const BN *n)
/* Decrypt = (x^17) mod n */
{
  BN tmp[1+128];

  BN_copy(tmp, x);
  dest[0]=dest[1]=1;
  BYTE t=17;

  for(;;) {
    if(t&1)
      BN_prodMod(dest, tmp, n);
    if((t >>= 1) == 0) break;
    BN_prodMod(tmp, tmp, n);
  }
}

void BN_powerMod(BN *dest, const BN *x, const BN *e, const BN *n)
{
  BN tmp[1+128];

  BN_copy(tmp, x);
  dest[0]=dest[1]=1;

  for(int len=e[0]; len>0; len--) {
    for(BYTE t = *e++, bit = 7; bit>0; bit--) {
      if(t&1)
        BN_prodMod(dest, tmp, n);
      if((t >>= 1) == 0 && len == 0) break;
      BN_prodMod(tmp, tmp, n);
    }
  }
}
```

Message Digests (md5.o)

This object implements the MD5 Message-Digest Algorithm optimised for 16 bit operation rather than 32. The code is based on "RSA Data Security, Inc. MD5 Message-Digest Algorithm". A good reference and this source can be found at [RFC1321].

```
/* UINT4 defines a four byte word */
typedef unsigned long int UINT4;

/* MD5 context. */
typedef struct {
  UINT4 state[4];                                   /* state (ABCD) */
  UINT4 count[2];        /* number of bits, modulo 2^64 (lsb first) */
  unsigned char buffer[64];                         /* input buffer */
} MD5_CTX;
```

```
static const unsigned char PADDING[64] = {
```

```
  0x80, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
};

void MD5Init (MD5_CTX *context);
#define MD5Init ASAP_ref(596, void (*const)(MD5_CTX *))
void MD5Update (MD5_CTX *context, unsigned char *input, unsigned int inputLen);
#define MD5Update ASAP_ref(597, void (*const)(MD5_CTX *, unsigned char *, unsigned int))
void MD5Final (unsigned char digest[16], MD5_CTX *context);
#define MD5Final ASAP_ref(598, void (*const)(unsigned char [16], MD5_CTX *))

void MD5Done (unsigned char desc_digest[17], MD5_CTX *context);
#define MD5Done ASAP_ref(599, void (*const)(unsigned char [17], MD5_CTX *))
/* calls MD5Final, stores length in BN form (length byte dest[0])
-The only extension to this package by TI.
*/

static ULONG ROTATE_LEFT (ULONG x, ULONG n);
static void MD5Transform (UINT4 state[4], unsigned char block[64]);
static void Encode (unsigned char *output, UINT4 *input, unsigned int len);
static void Decode (UINT4 *output, unsigned char *input, unsigned int len);
```

Certificates (cert.o)

The certificate packages up all the data required to perform authentification. The different components are split up into various fields, which can be accessed fairly easily, and validated by the routines available in this module.

Field search Ids use lower 4 bits to encode size of field, as the fields are stored sequentially. This length can be a variable size. Fields are opened with copen.

Fields may contain other fields, which may be opened with copensub.

The length of fields is identified as follows.

```
Field & 0xF :
  <= 0xC len
  0xD char len
  0xE short len
  0xF long len
```

When searching for an ID the length field is set to 0.

```
Layout of 0x412000
0x8000
      Certificate
      Product ID = form("%02lX-%lX-%lx-%lX", 0x8010, 0x80A0, 0x8020, 0x8030);
      0x8010      x-?-?-? (1)
      0x8020      ?-?-x-? (1)
      0x8030      ?-?-?-x (0x4F)
      0x80A0      ?-x-?-? (0)

      0x8040      Product Name: "Advanced Mathematics Software"
      0x0320      Product code ? (6 bytes)
      0x0200      Signature - encrypted MD5 of 0x320 field (including hdr)
      0x8070      0xCCCCCCCC, actual ROM
0x0200
      Signature of entire ROM
```

Note: 0x0320 & 0x0200 is some authenticated number. This 0x0200 is the encrypted digital signature of the 0x0320 field. (Although I have not found these to match properly!)

```
Guessed info on Certificate 0x410004
0x0300 (caddcert: only continue if 0x0200 field follows)
      0x100 long Cert Revision Number
(0x0200)?
      0x700
            0x710
```

0x0330
0xA10 BYTE **Serial Number**[6?] (used as "pass phrase") (-> 10 hex digits?)
0xA20 Cert Start (encrypted certificate, encrypted with MD5 of Serial Number?)
Decrypted -> WORD **Version Number**

Note: pass phrase means the MD5 of this field is used a decryption key for the Version Number.

```
typedef WORD C_ER_CODE;
struct field {
  int ID : 12;
  int len : 4;
};

enum CERT_FIND_LEN { CERT_BYTE_LEN=0xD, CERT_WORD_LEN=0xE, CERT_LONG_LEN=0xF };
  //else value stored there
enum CERT_FIND_ID {
  CF_BASE = 0x8000, //total rom size
  CF_PRODID1 = 0x8010,
  CF_PRODID3 = 0x8020,
  CF_PRODID3 = 0x8030,
  CF_PRODIDA = 0x80A0,
  CF_PRODNAME = 0x8040, //"Advanced Mathematics Software"
  CF_EOF = 0xFFF0,

  CF_BASE_CODE = 0x8070,
};

typedef struct {
  PTR_ALL Start, Pos, End;
  BOOL EOF;
} CFILE;

typedef struct {
  WORD Field;
  WORD HdrLen; //size of Field and optional length field
  size_t Len;
  PTR_ALL Data;
} CERT_FIELD;

C_ER_CODE caddcert(const void *cert, size_t size);
#define caddcert ASAP_ref(294, WORD (*const)(const void *, size_t))
/* returns certificate error code */

static BOOL cmemeq(const void *m1, size_t len1, const void *m2, size_t len2);
static WORD cmemcmp(const void *m1, size_t len1, const void *m2, size_t len2);
/* compares memory, after incrementing m1, m2 past any zero bytes */

void cdecrypt(BN *dest, BYTE *src, size_t size, BN *public);
#define cdecrypt ASAP_ref(295, void (*const)(BN *, BYTE *, size_t, BN *))
BOOL ceof(CFILE *context); //End Of File
#define ceof ASAP_ref(296, BOOL (*const)(CFILE *))
BOOL cfindcertfield(WORD field, WORD subfield=0, CERT_FIELD *dest);
#define cfindcertfield ASAP_ref(297, BOOL (*const)(WORD, WORD, CERT_FIELD *))
BOOL cfindfield(CFILE *context, WORD field, CERT_FIELD *dest);
#define cfindfield ASAP_ref(298, BOOL (*const)(CFILE *, WORD, CERT_FIELD *))
/* finds a matching field from context, stores in dest
returns success
*/

BYTE cgetc(CFILE *context);
#define cgetc ASAP_ref(299, BYTE (*const)(CFILE *))
/* get char from context */

void cgetcert(HANDLE *hDest, size_t *len, BOOL decrypt);
#define cgetcert ASAP_ref(300, void (*const)(HANDLE *, size_t *, BOOL))
/* This routine is run through FL_getCert, running with flash protection off
(I would be intereseted to see the result of this command (with decrypt = FALSE))
success if hDest != H_NULL
*/
```

```
BOOL cgetcertrevno( long *dest);
#define cgetcertrevno ASAP_ref(672, BOOL (*const)( long *))
/* read long from revno field
return success
*/

size_t cgetflen(CFILE *context, WORD field);
#define cgetflen ASAP_ref(301, size_t (*const)(CFILE *, WORD))
/* get the length of field
if variable lenth, it is read from context
*/

long cgetfnl(CERT_FIELD *field);
#define cgetfnl ASAP_ref(302, long (*const)(CERT_FIELD *))
/* Get non-aligned long from field
field can be of any length, gets as many bytes as available up to long
*/

long cgetnl(CFILE *context);
#define cgetnl ASAP_ref(303, long (*const)(CFILE *))
/* Get non-aligned long
as cgetfnl
*/

short cgetns(CFILE *context);
#define cgetns ASAP_ref(304, short (*const)(CFILE *))
/* Get non-aligned short */

void cgetsn(char* dest);
#define cgetsn ASAP_ref(673, void (*const)(char*))
/* get serial number from certificate */

WORD cgetvernum( void );
#define cgetvernum ASAP_ref(305, WORD (*const)( void ))
/* get encrypted version number from certificate, runs from FL_getVerNum */

void copen(CFILE *context, BYTE *data, size_t size);
#define copen ASAP_ref(306, void (*const)(CFILE *, BYTE *, size_t))
/* opens a context - initialises context */

void copensub(CFILE *context, CERT_FIELD *subfield);
#define copensub ASAP_ref(307, void (*const)(CFILE *, CERT_FIELD *))
/* open a subfield
used mainly to reset position to start of a group of items
general procedure is copen ; cfindfield ; { copensub ; cfindfield ; }[]
*/

//Write values into context
BOOL cputhdr(CFILE *context, WORD field, WORD len);
#define cputhdr ASAP_ref(308, BOOL (*const)(CFILE *, WORD, WORD))
void cputnl(CFILE *context, long l);
#define cputnl ASAP_ref(309, void (*const)(CFILE *, long))
void cputns(CFILE *context, short s);
#define cputns ASAP_ref(310, void (*const)(CFILE *, short))

BOOL cread(CFILE *context, CERT_FIELD *dest);
#define cread ASAP_ref(311, BOOL (*const)(CFILE *, CERT_FIELD *))
/* read a field from the current context
moves to the next field
returns TRUE on success
*/

ULONG ctell(CFILE *context);
#define ctell ASAP_ref(312, ULONG (*const)(CFILE *))
/* return current position from start of context */

BOOL cwrite(CFILE *context, CERT_FIELD *source);
#define cwrite ASAP_ref(313, BOOL (*const)(CFILE *, CERT_FIELD *))
```

```
/* writes contents of field into context */

static WORD getcerttype(WORD field);
/* gets the field type:
0xFFF0 -> 1
0x0300 -> 2
0x0320 -> 3
0x0330 -> 4
0x???? -> 0
*/

//static more...
```

Modes and Settings

- Settings Structures
- Mode Interface
- Units

Settings Structures (Settings.o)

This object provides the interface for setting and retrieving mode settings and names. It is used subsequently in mode.o. The actual layout listed here is a little vague, but since the structures are correct, the rest should be clear.

```
typedef struct {
  WORD Num
  WORD XR_Vals[1]; /* Should be [Num] */
} SETTING_VALS;

typedef struct {
  WORD XR_Name;
  WORD (*SetFunc)(WORD Val, BOOL SetSys);
  WORD (*GetFunc)( void );
  SETTING_VALS *Settings;
} SETTING;

typedef struct {
  WORD Num;
  SETTING Setting[1]; /* should be [Num] */
} SETTINGS;
```

```
const SETTINGS Mode;
const SETTINGS Graph;
const SETTINGS GR3D;
const SETTINGS TblSet;

const SETTING_VALS AllVals[];
```

Mode Interface (mode.o)

These routines handle the mode options dialog and "digesting" these options into the various OS variables. Mostly between "options" and "mode_options" in: Units

- TI-BASIC Interface
- Dialog Handler

TI-BASIC Interface (units.o)

```
init_unit_system
#define init_unit_system ASAP_ref(944, ASAP_UNDEF)
push_unit_system_list
```

```
#define push_unit_system_list ASAP_ref(947, ASAP_UNDEF)
setup_unit_system
#define setup_unit_system ASAP_ref(948, ASAP_UNDEF)
has_unit_base
#define has_unit_base ASAP_ref(943, ASAP_UNDEF)
is_units_term
#define is_units_term ASAP_ref(945, ASAP_UNDEF)
did_push_divide_units
#define did_push_divide_units ASAP_ref(942, ASAP_UNDEF)
push_auto_units_conversion
#define push_auto_units_conversion ASAP_ref(946, ASAP_UNDEF)
```

Dialog Handler (wcmd.o?)

Just some extra Instruction tokens and keypresses on the PLUS machines
Note this really shouldn'y belong here. This provides some extra functions on PLUS machines, but cmd_customon and off just call external routines.

```
void UnitsDialog( void );
/* called via diamond-p */

WORD Units_callBack(WORD a, DWORD b);

static void CheckUnits(BOOL checkbase);

void cmd_custmon( void );
#define cmd_custmon ASAP_ref(822, void (*const)( void ))
void cmd_custmoff( void );
#define cmd_custmoff ASAP_ref(821, void (*const)( void ))
```

EOS - The Expression Stack.

```
typedef struct {
  WORD CurrentFolder;
  WORD SplitScreen;
  WORD NumGraphs;
  WORD Graph1;
  WORD Graph2;
  WORD Split1App;
  WORD Split2App;
  WORD SplitRatio;
  WORD Angle;
  WORD ExactApprox;
  WORD Fix; /* Display Digits */
  WORD Exp; /* Exponential Form */
  WORD Vector;
  WORD Complex;
  WORD Pretty;
  WORD Base;
  WORD UnitSystem;
  WORD CustomUnits;
} MO_OPTIONS;

typedef struct {
  WORD Area;
  WORD Capacitance;
  WORD Charge;
  WORD Conductance;
  WORD ElectricCurrent;
  WORD Energy;
  WORD Force;
  WORD Frequency;
  WORD Inductance;
  WORD Length;
  WORD MagFieldStrength;
```

```
  WORD MagFluxDensity;
  WORD MagneticFlux;
  WORD Mass;
  WORD Potential;
  WORD Power;
  WORD Pressure;
  WORD Resistance;
  WORD Time;
  WORD Velocity;
  WORD ViscosityDynamic;
  WORD ViscosityKinematic;
  WORD Volume;
} MO_CUSTM_UNITS;

typdef struct {
  const char *Name;     /* mode setting name */
  SETTING_VALS *Popup;  /* ptr in 0settings.o, array of possible values (XR) */
  WORD Index;           /* Index in MO_OPTIONS */
MO_GETMODE;
```

```
WORD MO_callBack(WORD a, DWORD b);
/* Mode Dialog Callback */

void MO_currentOptions( void );
#define MO_currentOptions ASAP_ref(217, void (*const)( void ))
/* fill in options structure from current settings */

void MO_defaults( void );
#define MO_defaults ASAP_ref(218, void (*const)( void ))
/* reset options structure, calls MO_digestOptions: */

void MO_digestOptions( void );
#define MO_digestOptions ASAP_ref(219, void (*const)( void ))
/* updates system mode settings from options */

void MO_init( void );

BOOL MO_isMultigraphTask(TASK App);
#define MO_isMultigraphTask ASAP_ref(220, BOOL (*const)(TASK))
/* Returns true if Task is Yeq, Window, Graph, or Table */

void MO_modeDialog( void );
#define MO_modeDialog ASAP_ref(221, void (*const)( void ))
void MO_notifyModeChange( void );
#define MO_notifyModeChange ASAP_ref(222, void (*const)( void ))
/* send mode change notify message to all apps */

static MO_GETMODE GetMode[15];
static void push_getmode_all( void );

//These get the mode settings from mode_flags in NG.o, and other system locations
WORD MO_getAngle( void );
WORD MO_getComplex( void );
WORD MO_getExp( void );
WORD MO_getGraphMode1( void );
WORD MO_getGraphMode2( void );
WORD MO_getGraphCount( void );
WORD MO_getExactApprox( void );
WORD MO_getPretty( void );
WORD MO_getSplit1App( void ); /* EV_appA */
WORD MO_getSplit2App( void ); /* EV_appB */
WORD MO_getSplitRatio( void );
WORD MO_getSplitMode( void );
WORD MO_getVector( void );
WORD MO_getBase( void );

void MO_sendQuit(TASK TaskID, WORD Side);
#define MO_sendQuit ASAP_ref(223, void (*const)(TASK, WORD))
```

```
WORD setoption(WORD Val, WORD Index, BOOL SetSys);
/* If SetSys set, MO_currentOptions, and MO_digestOptions are called
Retruns the previous value, used in the next few functions
*/

//Update current mode settings using setoption above
WORD MO_setAngle(WORD Val, BOOL SetSys);
WORD MO_setComplex(WORD Val, BOOL SetSys);
WORD MO_setExp(WORD Val, BOOL SetSys);
WORD MO_setGraphMode1(WORD Val, BOOL SetSys);
WORD MO_setGraphMode2(WORD Val, BOOL SetSys);
WORD MO_setGraphCount(WORD Val, BOOL SetSys);
WORD MO_setExactApprox(WORD Val, BOOL SetSys);
WORD MO_setPretty(WORD Val, BOOL SetSys);
WORD MO_setSplit1App(WORD Val, BOOL SetSys); /* EV_appA */
WORD MO_setSplit2App(WORD Val, BOOL SetSys); /* EV_appB */
WORD MO_setSplitRatio(WORD Val, BOOL SetSys);
WORD MO_setSplitMode(WORD Val, BOOL SetSys);
WORD MO_setVector(WORD Val, BOOL SetSys);
WORD MO_setBase(WORD Val, BOOL SetSys);

//None of these use the BOOL parameter (and may be called differently)
WORD MO_setGR3Style(WORD Val, BOOL SetSys);
WORD MO_setAxes(WORD Val, BOOL SetSys);
WORD MO_setCoords(WORD Val, BOOL SetSys);

static void setGraphFlags(WORD or1, WORD or2); /* used in MO_setDEAxes */
WORD MO_setDEAxes(WORD Val, BOOL SetSys);
WORD MO_setDEFields(WORD Val, BOOL SetSys);
WORD MO_setGR3Grid(WORD Val, BOOL SetSys);
WORD MO_setLabels(WORD Val, BOOL SetSys);
WORD MO_setLeadingCursor(WORD Val, BOOL SetSys);
WORD MO_setGraphOrder(WORD Val, BOOL SetSys);
WORD MO_setSeqAxes(WORD Val, BOOL SetSys);
WORD MO_setDESolnMethod(WORD Val, BOOL SetSys);
WORD MO_setTEGraphToTable(WORD Val, BOOL SetSys);
WORD MO_setTEIndependent(WORD Val, BOOL SetSys);

static const char *StyleNames[7] =
  { "LINE", "DOT", "THICK", "ANIMATE", "PATH", "ABOVE", "BELOW", "SQUARE" };

void cmd_style(EStackIndex EqnNum, EStackIndex StyleString);
#define cmd_style ASAP_ref(914, void (*const)(EStackIndex, EStackIndex))

static void current_gr_modes( void );
/* fill in option from current graph modes, i.e. distinguish graphs 1&2 */

static void access_settings
  (SETTINGS *Settings, EStackIndex name, EStackIndex setting, BOOL set, BOOL mode);

static char *get_cstr(EStackIndex string);
/* throws error if string==NULL or not a valid string */

void push_getmode(EStackIndex modeNameString);
#define push_getmode ASAP_ref(792, void (*const)(EStackIndex))

void push_setgraph(EStackIndex modeNameString, EStackIndex settingString);
#define push_setgraph ASAP_ref(802, void (*const)(EStackIndex, EStackIndex))
void push_setmode(EStackIndex modeNameStringOrList, EStackIndex settingString);
#define push_setmode ASAP_ref(803, void (*const)(EStackIndex, EStackIndex))
void push_settable(EStackIndex modeNameString, EStackIndex settingString);
#define push_settable ASAP_ref(804, void (*const)(EStackIndex, EStackIndex))

static char *convert_modestring(char *dest, const char *src);

static void CustomUnitsDialog( void );
WORD UnitDlgCallBack(WORD a, DWORD b);
static void CustomUnitsDefaults( void );
```

```
//Unit initialisation
?
?
?
?

void UnitsInit( void );

void push_getconfg( void );

static HANDLE hFolderPopup;

WORD MO_splitScreen;
WORD MO_graphCount;
WORD MO_graphMode1;
WORD MO_graphMode2;
WORD MO_fix;
WORD MO_exp;
WORD MO_vector;
WORD MO_flags;
WORD MO_units;
WORD MO_base;

//Structures used as Dialog Popup buffers
MO_OPTIONS option;
MO_CUSTM_UNITS units;


DIALOG ModeDialog;
DIALOG CustomUnitDialog;
```

Units

- TI-BASIC Interface
- Dialog Handler

TI-BASIC Interface (units.o)

```
init_unit_system
#define init_unit_system ASAP_ref(944, ASAP_UNDEF)
push_unit_system_list
#define push_unit_system_list ASAP_ref(947, ASAP_UNDEF)
setup_unit_system
#define setup_unit_system ASAP_ref(948, ASAP_UNDEF)
has_unit_base
#define has_unit_base ASAP_ref(943, ASAP_UNDEF)
is_units_term
#define is_units_term ASAP_ref(945, ASAP_UNDEF)
did_push_divide_units
#define did_push_divide_units ASAP_ref(942, ASAP_UNDEF)
push_auto_units_conversion
#define push_auto_units_conversion ASAP_ref(946, ASAP_UNDEF)
```

Dialog Handler (wcmd.o?)

Just some extra Instruction tokens and keypresses on the PLUS machines
Note this really shouldn'y belong here. This provides some extra functions on PLUS machines, but cmd_customon and off just call external routines.

```
void UnitsDialog( void );
/* called via diamond-p */

WORD Units_callBack(WORD a, DWORD b);

static void CheckUnits(BOOL checkbase);

void cmd_custmon( void );
```

```
#define cmd_custmon ASAP_ref(822, void (*const)( void ))
void cmd_custmoff( void );
#define cmd_custmoff ASAP_ref(821, void (*const)( void ))
```

EOS - The Expression Stack (estack)

All statements are tokenised before being executed (interpreted). Instructions are reduced to (byte sized) quanta and parsed into Reverse Polish Notation (RPN). This is a common technique in interpreted languagues.

Expressions, functions, etc. are all stored in RPN form, to allow for efficient operation of the EOS.

Examples

	Expression	RPN
1	a + b	A B +
2	(a + b) * c	A B + C *

The actual processing is done via a stack, called the estack. The position in the stack is given by top_estack. Pushing a value appends it to the estack and increments top_estack.

When a expression is interpreted, expressions are reduced, and executed as far as possible. Whatever remains on the stack Is the result, this may then be stored or processed later.

When a file is interpreted the end of the variable is found an it is processed as a separate expression stack. It is then processed recursively, producing a simplified version on the real estack.

Expressions are therefore interpreted from the top (high memory) recursively.

Implementation:
A variable may consist of multiple expressions, these are separated by several special quanta: NEXT_TAG, NEXTLINE_TAG. The last expression is marked with ENDSTACK_TAG.

Many internal (and all external functions) have a variable nmber of parameters, the last expression is marked with another special quantum END_TAG.

The format of user defined functions would be listed in the manual as:
USERFUNC(VAR[,EXPR, …])

Example

Expression	Stack
f(a, b)	ENDSTACK_TAG, END_TAG, B_TAG, A_TAG, F_TAG, USERFUNC_TAG

- Fundamental Routines
- Basic Push Operations
- Further estack Operations
- Vector Operations
- Matrix Operations
- TI-BASIC Variables
- Variables

Fundamental Routines (nestack.o?)

```
typedef BYTE TAG;

typedef BYTE * EStackIndex;
typedef DWORD EStackDisplacement;

/* Used particularly when resizing lists etc. Indexes are converted of displacements
before reallocation, and then generally restored */
```

```
typedef union {
  EStackIndex i;
  EStackDisplacement o;
} ESTACK_AND_OFFSET;

/* General deref of esi (may not cast) */
#define ESTACK(x) *((BYTE *)(x))
```

Push_operations

```
void push_quantum(BYTE tag);
#define push_quantum ASAP_ref(750, void (*const)(BYTE))

void push_quantum_pair(BYTE tag1, BYTE tag2);
#define push_quantum_pair ASAP_ref(751, void (*const)(BYTE, BYTE))
/* pushes tag1 then tag2
*/

void push_between(EStackIndex esi1, EStackIndex esi2);
#define push_between ASAP_ref(744, void (*const)(EStackIndex, EStackIndex))
/* pushes bytes between esi1 and esi2
*/

void delete_between(EStackIndex esi1, EStackIndex esi2);
#define delete_between ASAP_ref(707, void (*const)(EStackIndex, EStackIndex))
/* deletes bytes between esi1 and esi2
moves memory past esi2 to esi1
*/

WORD deleted_between(EStackIndex esi1, EStackIndex esi2);
#define deleted_between ASAP_ref(708, WORD (*const)(EStackIndex, EStackIndex))
/* same as delete_between, returns number of bytes deleted
*/

void delete_expression(EStackIndex esi);
#define delete_expression ASAP_ref(709, void (*const)(EStackIndex))
/* deletes the expression at esi
*/

WORD deleted_expression(EStackIndex esi);
#define deleted_expression ASAP_ref(710, WORD (*const)(EStackIndex))
/* same as delete_expression, returns number of bytes removed
*/

void move_between_to_top(EStackIndex esi1, EStackIndex esi2);
#define move_between_to_top ASAP_ref(741, void (*const)(EStackIndex, EStackIndex))
/* moves bytes between esi1 and esi2 to top of estack
*/

WORD moved_between_to_top(EStackIndex esi1, EStackIndex esi2);
#define moved_between_to_top ASAP_ref(742, WORD (*const)(EStackIndex, EStackIndex))
/* same as move_between_to_top, returns number of bytes moved
*/

EStackIndex next_expression_index(EStackIndex esi);
#define next_expression_index ASAP_ref(266, EStackIndex (*const)(EStackIndex))
/* finds the end of the expression at esi
calls index_after_match_endtag for Func & Prgm tags, so that inline functions are counted
as entire expressions rather than of their components
*/

static void _push_equation(EStackIndex esi);
void push_equation(EStackIndex esi);
/* Push Solver Equation? */

void push_expression(EStackIndex esi);
/* pushes the expression at esi
*/
```

```
BYTE min_quantum(BYTE tag1, BYTE tag2);
#define min_quantum ASAP_ref(740, BYTE (*const)(BYTE, BYTE))
/* returns min(tag1, tag2)
*/

void push_next_arb_int( void );
#define push_next_arb_int ASAP_ref(747, void (*const)( void ))
/* pushes @nxxx
increments count, and pushes ARB_INT_TAG, xxx
*/

void push_next_arb_real( void );
#define push_next_arb_real ASAP_ref(748, void (*const)( void ))
/* pushes @xxx
as push_next_arb_int, see csolve
*/

void push_expr_quantum(EStackIndex esi, BYTE tag);
#define push_expr_quantum ASAP_ref(745, void (*const)(EStackIndex, BYTE))
/* pushes expression at esi, followed by tag
*/

void push_expr2_quantum(EStackIndex esi1, EStackIndex esi2, BYTE tag);
#define push_expr2_quantum ASAP_ref(746, void (*const)(EStackIndex, EStackIndex, BYTE))
/* pushes expressions esi1, esi2, tag
*/

BOOL is_free_of_tag(EStackIndex esi, BYTE tag);
#define is_free_of_tag ASAP_ref(724, BOOL (*const)(EStackIndex, BYTE))
/* returns if expression at esi is free of tag
*/

void push_Init( void );
/* clears all flags, does not create the estack
*/

void estack_Init( void );
/* allocates at initialises estack
*/

void reset_estack_size(WORD newsize);
#define reset_estack_size ASAP_ref(755, void (*const)(WORD))
/* Reallocates estack
*/

void check_estack_size(WORD size);
#define check_estack_size ASAP_ref(706, WORD (*const)(WORD))
/* Throws ER_MEMORY if not enough enough memory on estack for size bytes */

void reset_control_flags( void );
#define reset_control_flags ASAP_ref(756, void (*const)( void ))

const char *find_error_message(ER_CODE error);
#define find_error_message ASAP_ref(705,const char *(*const)(WORD))

ER_CODE get_nearest_error_code(ER_CODE error);
void clear_error_context( void );

FLOAT init_precision, init_errorbound;    /* these are just initialised to FPZERO */
EStackIndex estack_min_index, estack_max_index;
EStackIndex top_estack;
#define top_estack ASAP_ref(265, EStackIndex (*const))

HANDLE estack_handle;
#define estack_handle (*(HANDLE*)(&top_estack+1))
/* Just as an example, be careful with casting! */

EStackIndex bottom_estack;
```

```
BYTE NG_return_flag;
HANDLE NG_code_handle;
EStackIndex NG_first_instr, NG_cur_instr;

BYTE ERR_return_flag;
HANDLE ERR_code_handle;
WORD ERR_offset;
ER_CODE ERR_errorcode;

char *top_parser_index;
char parser_cur_char;

FLOAT ?;          /* to do with calculus? */
FLOAT precision;  /* precision used by push_Float_to_rat (push_round_Float) */
FLOAT errorbound; /* determines some kind of error bound? */

BYTE arb_real_count, arb_int_count, internal_var_count;

WORD estack_size;
DWORD control;

DWORD NG_Control;
EStackIndex top_prev, parameter_in_tok_text,  ?, paramlist?oldtop;
BOOL bComplex, bFactor, bConstraint, bDiff;

#ifdef PLUS
BOOL de1?, de2?;
#endif

EStackIndex ?;
```

Basic Push Operations (basic.o)

```
// All referenced through pointers in RAM
const TAG int_zero[2] = { 0, POSINT_TAG };
const TAG int_one[3] = { 1, 1, POSINT_TAG };
const TAG int_neg_one[3] = { 1, 1, NEGINT_TAG};
const TAG int_two[3] = { 2, 1, POSINT_TAG };
const TAG int_neg_two[3] = { 2, 1, NEGINT_TAG };

const TAG frac_neg_zero[4] = { 1, 1, 0, NEGFRAC_TAG };
const TAG frac_half[4] = { 2, 1, 1, 1, POSFRAC_TAG };

const TAG int_ten[3] = { 10, 1, POSINT_TAG };
const TAG int_298[3] = { 0x2A, 1, 2, POSINT_TAG };

const TAG int_maxlong[7] = { 0, 0, 0, 0, 1, 5, POSINT_TAG };
const TAG int_1e14[8] = { 0, 0x40, 0x7A, 0x10, 0xF3, 0xFA, 6, POSINT_TAG };
const TAG int_449[4] = { 0xC1, 1, 2, POSINT_TAG };

const TAG infinity = INFINITY_TAG;
const TAG neginfinity = NEGINFINITY_TAG;
const TAG undef = UNDEF_TAG;

const TAG matrix[4] = { END_TAG, END_TAG, LIST_TAG, LIST_TAG };
```

```
void init_tag_pointers( void );
```

```
static void pushint1( void );
static void pushint0( void );
static void pushintN1( void );
```

```
void push1( void );
void push0( void );
```

```
void pushfrac0( void );
```

```
void pushN1( void );
void pushhalf( void );
void pushNhalf( void );
void push2( void );

void push1_typed(TAG type);
/* if type type is BCD_TAG push float */
void push0_typed(TAG type);
void pushN1_typed(TAG type);

DWORD estack_unsint_to_long(EStackIndex esi);
/* where *esi is the number of bytes of preceeding number */

DWORD estack_int_to_long(EStackIndex esi);
/* get the signed value from pointer to INT */

void pushint_ulong(ULONG ulong);

//...


TAG *INT_ZERO, *INT_ONE, *INT_NEGONE, *INT_TWO, *INT_NEGTWO;
TAG *FRAC_ZERO, *FRAC_HALF;
TAG *INT_TEN, *INT_298, *INT_MAXLONG, *INT_1E14, *INT_449;
TAG *INFINITY, *N_INFINITY, *UNDEFINED, *EMPTY_MATRIX, *EMPTY_LIST;
```

Further estack Operations (n???.o)

This may well be part of the above module, although it always used to be separate. This is just noted that there is no forced alignment between the two routines (when there usually is).

Note tail operations are those dealing with many elements, i.e. lists etc. Generally functions with tail in there name will perform an operation on all the elements in such an array.

```
void map_unary_over_comparison(void (*f)(EStackIndex next), EStackIndex cmp);
#define map_unary_over_comparison ASAP_ref(739, ASAP_UNDEF)
/* calls f for both comparison terms, and pushes the comparison tag *cmp */

BOOL is_simple_tag(EStackIndex esi);
/* VAR or ARB */

BOOL is_advanced_tag(BYTE tag);
#define is_advanced_tag ASAP_ref(720, BOOL (*const)(BYTE))
/* matrix, localvar, dash, fdash, ... */

BOOL is_complex_number(EStackIndex esi);
#define is_complex_number ASAP_ref(722, BOOL (*const)(EStackIndex))

BOOL is_complex0(EStackIndex esi);
#define is_complex0 ASAP_ref(723, BOOL (*const)(EStackIndex))
/* complex & ==0 */

//(lead_term)+(reductum) [(reductum)=(second_term)+...]
EStackIndex lead_term_index(EStackIndex esi);
#define lead_term_index ASAP_ref(737, EStackIndex (*const)(EStackIndex))
/* get lead term (skip past ADD_TAG) */
EStackIndex reductum_index(EStackIndex esi);
#define reductum_index ASAP_ref(752, EStackIndex (*const)(EStackIndex))
/* get the other term in ADD operation (0 if not ADD_TAG) */

//(lead_factor)*(remaining_factors) [(remaining factors)=(second factor)...]
EStackIndex lead_factor_index(EStackIndex esi);
#define lead_factor_index ASAP_ref(736, EStackIndex (*const)(EStackIndex))
/* get lead factor (skip past MUL_TAG) */
```

```
EStackIndex remaining_factors_index(EStackIndex esi);
#define remaining_factors_index ASAP_ref(753, EStackIndex (*const)(EStackIndex))
/* get esi of other terms (first expr in MUL operation, 1 if not MUL) */

//(factor_base)^(factor_exponent)
EStackIndex factor_base_index(EStackIndex esi);
#define factor_base_index ASAP_ref(713, EStackIndex (*const)(EStackIndex))
/* get base term at esi (skip past POW_TAG) */
EStackIndex factor_exponent_index(EStackIndex esi);
#define factor_exponent_index ASAP_ref(714, EStackIndex (*const)(EStackIndex))
/* get exponent of term at esi (first expr in POW operation, 1 if not pow) */

EStackIndex lead_base_index(EStackIndex esi);
#define lead_base_index ASAP_ref(734, EStackIndex (*const)(EStackIndex))
/* factor base of lead factor */
EStackIndex lead_exponent_index(EStackIndex esi);
#define lead_exponent_index ASAP_ref(735, EStackIndex (*const)(EStackIndex))
/* factor exponent of lead factor */
EStackIndex index_of_lead_base_of_lead_term(EStackIndex esi);
#define index_of_lead_base_of_lead_term ASAP_ref(718, EStackIndex (*const)(EStackIndex))
/* lead base of lead_term */

EStackIndex numeric_factor_index(EStackIndex esi);
#define numeric_factor_index ASAP_ref(743, EStackIndex (*const)(EStackIndex))
/* searches factors for numeric term (if not found returns 1 */
EStackIndex index_numeric_term(EStackIndex EStackIndex (*const)(EStackIndex));
#define index_numeric_term ASAP_ref(717, EStackIndex (*const)(EStackIndex))
/* as for above, find constant value */

EStackIndex main_gen_var_index(EStackIndex esi);
#define main_gen_var_index ASAP_ref(738, EStackIndex (*const)(EStackIndex))
/* last encountered var (after skipping, ADD, MUL, POW, next_expression_index */
EStackIndex index_main_var(EStackIndex esi);
#define index_main_var ASAP_ref(719, EStackIndex (*const)(EStackIndex))
/* index of first encountered var */

static BOOL compare_vars(EStackIndex esi1, EStackIndex esi2);
static BOOL compare?(EStackIndex esi1, EStackIndex esi2);
static BOOL compare_expressions_aux(EStackIndex esi1, EStackIndex esi2);

BOOL are_expressions_identical(EStackIndex esi1, EStackIndex esi2);
#define are_expressions_identical ASAP_ref(703, BOOL (*const)(EStackIndex, EStackIndex))
BOOL compare_expressions(EStackIndex esi1, EStackIndex esi2);
#define compare_expressions ASAP_ref(704, BOOL (*const)(EStackIndex, EStackIndex))

BOOL is_monomial(EStackIndex esi);
#define is_monomial ASAP_ref(729, BOOL (*const)(EStackIndex))
BOOL is_monomial_in_kernel(EStackIndex esi);
#define is_monomial_in_kernel ASAP_ref(730, BOOL (*const)(EStackIndex))
BOOL is_independent_of_de_seq_vars(EStackIndex esi);
#define is_independent_of_de_seq_vars ASAP_ref(726, BOOL (*const)(EStackIndex))
BOOL is_tail_indepenent_of(EStackIndex list, EStackIndex var);
#define is_tail_independent_of ASAP_ref(733, BOOL (*const)(EStackIndex, EStackIndex))
BOOL is_independent_of(EStackIndex main, EStackIndex var);
#define is_independent_of ASAP_ref(725, BOOL (*const)(EStackIndex, EStackIndex))
BOOL is_independent_of_tail(EStackIndex main, EStackIndex var);
#define is_independent_of_tail ASAP_ref(727, BOOL (*const)(EStackIndex, EStackIndex))
BOOL is_independent_of_elements(EStackIndex main, EStackIndex var);
#define is_independent_of_elements ASAP_ref(728, BOOL (*const)(EStackIndex, EStackIndex))
BOOL is_narrowly_independent_of(EStackIndex main, EStackIndex var);
#define is_narrowly_independent_of ASAP_ref(731, BOOL (*const)(EStackIndex, EStackIndex))

BOOL is_antisymmetric(EStackIndex, EStackIndex);
#define is_antisymmetric ASAP_ref(721, BOOL (*const)(EStackIndex, EStackIndex))
BOOL is_symmetric(EStackIndex, EStackIndex);
#define is_symmetric ASAP_ref(732, BOOL (*const)(EStackIndex, EStackIndex))

EStackIndex re_index(EStackIndex esi);
```

```
#define re_index ASAP_ref(754, EStackIndex (*const)(EStackIndex))
EStackIndex im_index(EStackIndex esi);
#define im_index ASAP_ref(716, EStackIndex (*const)(EStackIndex))

void push_next_internal_var(BYTE tag);
#define push_next_internal_var ASAP_ref(749, void (*const)(BYTE))
/* pushes: 0, tag+1, 0, internal_var_count++
tag only ever 0 or 1
*/

void push_attach(EStackIndex esi, EStackIndex not, EStackIndex attach);
/* I am not sure about this function, it is used in calculus operations, and seems to
push "attach" along with every expression term in "esi", except "not"
This is probably used internally, "push_next_internal_var" seems to perform similarly
strange operations.
*/

int estack_to_ushort(EStackIndex esi, unsigned short *s);
#define estack_to_ushort ASAP_ref(712, ASAP_UNDEF)
/* returns -1 if some info lost, 0 error, 1 OK */
int estack_to_short(EStackIndex esi, short *s);
#define estack_to_short ASAP_ref(711, ASAP_UNDEF)

DWORD GetValue(EStackIndex esi, DWORD low, DWORD high);
#define GetValue ASAP_ref(715, ASAP_UNDEF)
/* Get (sign dependent on bounds) short value from estack, throw error if out of bounds
*/

void push_string(const char *s);
/* push ti-basic string */
```

Vector Operations (vectors.o)

```
WORD push_offset_array(EStackIndex elements, EStackIndex dest);
#define push_offset_array ASAP_ref(964, WORD (*const)(EStackIndex, EStackIndex))
/* (increments esi to even address) pushes (WORD) array of offsets of elements to estack
dest is top address of elements, returns element count
*/

WORD remaining_element_count(EStackIndex start);
#define remaining_element_count ASAP_ref(963, WORD (*const)(EStackIndex))
/* returns number of elements remaining (in END_TAG terminated expression array) */

void tag_element_divide(EStackIndex esi);
BOOL is_matrix_tail(EStackIndex esi);
/* called from is_matrix after first tag is checked */

BOOL is_matrix(EStackIndex esi);
#define is_matrix ASAP_ref(951, BOOL (*const)(EStackIndex))

BOOL is_valid_smap_aggregate(EStackIndex esi);
#define is_valid_smap_aggregate ASAP_ref(953, BOOL (*const)(EStackIndex))

void map_tail(void (*f)(EStackIndex elem), EStackIndex start);
#define map_tail ASAP_ref(955, void (*const)(void (*)(EStackIndex), EStackIndex))
/* calls f for evey element, in reverse order */

map_?
map_?
map_?
BOOL map_matrix_tail?
BOOL map_tail_Int(BOOL (*f)(EStackIndex elem, int n), EStackIndex start, int n);
#define map_tail_Int ASAP_ref(956, BOOL (*const)(BOOL (*)(EStackIndex, int), EStackIndex,
int))
/* Result is | all values returned by f, each function passed n (const) */

void map_tail_ESI_Int(void (*f)(EStackIndex elem, EStackIndex esi, int n), EStackIndex
start, EStackIndex esi, int n);
/* call f with paramataters element esi, esi and int n */
```

```
void map_tail_2ESI(void (*f)(EStackIndex elem, EStackIndex esi1, EStackIndex esi2),
EStackIndex start, EStackIndex esi1, EStackIndex esi2);

void map_tail_3ESI(void (*f)(EStackIndex elem, EStackIndex esi1, EStackIndex esi2,
EStackIndex esi3), EStackIndex start, EStackIndex esi1, EStackIndex esi2, EStackIndex
esi3);

void map_tail_4ESI(void (*f)(EStackIndex elem, EStackIndex esi1, EStackIndex esi2,
EStackIndex esi3, EStackIndex esi4), EStackIndex start, EStackIndex esi1, EStackIndex
esi2, EStackIndex esi3, EStackIndex esi4);

void push_list_plus(EStackIndex esi1, EStackIndex esi2);
#define push_list_plus ASAP_ref(957, void (*const)(EStackIndex, EStackIndex))

void push_matrix_product(EStackIndex esi1, EStackIndex esi2);
#define push_matrix_product ASAP_ref(965, void (*const)(EStackIndex, EStackIndex))

void push_list_times(EStackIndex esi1, EStackIndex esi2)
#define push_list_times ASAP_ref(958, void (*const)(EStackIndex, EStackIndex))

void push_eigvc(EStackIndex esi); /* table entry -> one param (no intermediate func)*/
void push_eigvl(EStackIndex esi);
void push_transpose(EStackIndex esi);
void push_transpose_aux(EStackIndex esi, BOOL conj); /* conj = TRUE */
#define push_transpose_aux ASAP_ref(961, void (*const)(EStackIndex, BOOL))

void tail_ESI(void (*f)(EStackIndex elem, EStackIndex esi), EStackIndext start,
EStackIndex esi); /* correct name? */

BOOL is_square_matrix(EStackIndex esi);
#define is_square_matrix ASAP_ref(952, BOOL (*const)(EStackIndex))

__cross_product
_cross_product
tag_cross_product

BOOL all_tail(BOOL (*f)(EStackIndex element), EStackIndex start);
#define all_tail ASAP_ref(949, BOOL (*const)(BOOL (*)(EStackIndex), EStackIndex))
/* call f for all elements up to tail (e.g. for lists etc.)
returns last value of f:
f returns FALSE to stop further processing.
*/

BOOL any_tail(BOOL (*f)(EStackIndex element), EStackIndex start);
#define any_tail ASAP_ref(950, BOOL (*const)(BOOL (*)(EStackIndex), EStackIndex))
/* call f for all elements up to tail
returns last value of f:
f returns TRUE to stop processing
*/

void push_reversed_tail(EStackIndex esi);
#define push_reversed_tail ASAP_ref(959, void (*const)(EStackIndex))
/* pushes elements up to tail onto the stack in reversed order
used for functions, e.g. parms are pushed like:
  parm1, parm2, ..., END_TAG
converted to:
   ..., END_TAG, ..., parm2, parm1 (note they still remain before END_TAG)
*/

EStackIndex last_element_index(EStackIndex esi);
#define last_element_index ASAP_ref(954, EStackIndex (*const)(EStackIndex))

push_sq_matrix_to_whole_number
#define push_sq_matrix_to_whole_number ASAP_ref(960, ASAP_UNDEF)

?
void tag_subscript(EStackIndex esi);
push_zero_partial_column
```

```
#define push_zero_partial_column ASAP_ref(962, ASAP_UNDEF)
```

Matrix Operations (matrix.o)

Not started yet

```
init_list_indices
#define init_list_indices ASAP_ref(698, ASAP_UNDEF)
get_list_indices
#define get_list_indices ASAP_ref(696, ASAP_UNDEF)
init_matrix_indices
#define init_matrix_indices ASAP_ref(699, ASAP_UNDEF)
get_matrix_indices
#define get_matrix_indices ASAP_ref(697, ASAP_UNDEF)
push_float_qr_fact
#define push_float_qr_fact ASAP_ref(700, ASAP_UNDEF)
push_symbolic_qr_fact
#define push_symbolic_qr_fact ASAP_ref(702, ASAP_UNDEF)
push_lu_fact
#define push_lu_fact ASAP_ref(701, ASAP_UNDEF)
```

TI-BASIC Variables (basicvars.o)

```
BOOL did_push_var_val(EStackIndex var);
#define did_push_var_val ASAP_ref(685, BOOL (*const)(EStackIndex))
BOOL does_push_fetch(EStackIndex var);
#define does_push_fetch ASAP_ref(686, BOOL (*const)(EStackIndex))
BOOL assign_between(EStackIndex var, EStackIndex low, EStackIndex high);
#define assign_between ASAP_ref(684, BOOL (*const)(EStackIndex, EStackIndex,
EStackIndex))
/* calls VarStore, slightly odd method used */
HANDLE push_ans_entry(EStackIndex esi);
#define push_ans_entry ASAP_ref(688, HANDLE (*const)(EStackIndex))
void push_indir_name(EStackIndex esi);
#define push_indir_name ASAP_ref(690, void (*const)(EStackIndex))
/* deals with indirection tag (*esi must be a string etc.) */

//paramater functions these throw errors if variables do not satisfy conditions
```

//see Execution

- Simplification and Instructions
- Interpreter
- Instruction Handling

```
Simplification and Instructions (program.o)
void parmfunc_user_var(EStackIndex esi);
/* any var but system vars */
void parmfunc_var(EStackIndex esi);
/* any var type */

EStackIndex index_after_match_endtag(EStackIndex esi, BYTE endtag);
#define index_after_match_endtag ASAP_ref(689, EStackIndex (*const)(EStackIndex, BYTE))
/* Finds the end of a multi-statement Instruction block, matching endtag (e.g. EndPrgm)
Called from next_expression_index, with usual entry values PRGM_ITAG, FUNC_ITAG
It is recursive, and handles all the End* statements.
*/

void store_func_def(EStackIndex esi);
#define store_func_def ASAP_ref(692, void (*const)(EStackIndex))
/* Varstore Flags 0xDA */

void to_boolean(EStackIndex esi);
/* used in run_user_func to convert previous expression to boolean statement if present
*/
BOOL run_user_func(EStackIndex esi, BOOL start, BOOL *twinDelete);
/* called from push_user_func, twinDelete is set from this func if no error occurs */
```

```
void push_user_func(EStackIndex esi, BOOL start);
#define push_user_func ASAP_ref(691, void (*const)(EStackIndex, BOOL))
/* deal with user_func tag
start=TRUE when called from push_simplify_statements, FALSE if from another func
 */

void store_to_subscripted_element(const EStackIndex esi, const EStackIndex data);
#define store_to_subscripted_element ASAP_ref(693, void (*const)(const EStackIndex, const
EStackIndex))
/* interprets element access, and store elements, throws errors */
ER_CODE delete_list_element(const EStackIndex var, WORD element);
#define delete_list_element ASAP_ref(687, ER_CODE (*const)(const EStackIndex, WORD))
```

Variables

- Basic Types
- Type Recognition
- Symbol Tables
- Variable Interface

Basic Types

Numbers:

TAG	Type	Description
$1F	INT	Positive integer (Binary)
$20	INT	Negative integer (Binary)
$21	FRAC	Positive fraction (Binary)
$22	FRAC	Negative fraction (Binary)
$23	TAG_FLOAT	Floating point number (can be +ve or -ve)

```
packed struct UINT {
  BYTE bytes of integer [n]; //little endian
  BYTE n //number of bytes used to store integer
};
```

```
packed struct INT {
  UINT number;
  token +ve ? $1F : $20 ;
};
//Note: 0 is just $001F
```

```
packed struct FRAC {
  UINT denominator;
  UINT numerator;
  token +ve ? $21 : $22;
};
```

```
//Text:
//A standard array of characters
packed struct TEXT{
  char buf[];
  BYTE 0;
};
```

```
//How names are stored
packed struct NAME {
  BYTE 0;
  TEXT characters;
};
```

```
packed union VAR {
  NAME string;
```

```
  token char;
};
```

```
//STR data type "..."
packed struct STR_VAR {
  NAME string;
  token $2D;
};
```

```
//picture data, ceiling(width/8) bytes for each row of pixel
packed struct PIC_VAR {
  BITMAP bitmap;
  token $DF;
};
```

```
struct RET {
  token $E9;
  //../
}
```

```
all stored variables are of form:
struct STORE {
  WORD size;
  BYTE data[size];
}
```

So STORE in a definition means that data is preceeded by a size WORD.

```
typdef struct {
  struct { //WORD
    bool sign : 1; //1=negative
    int biased_exponent : 15; //$4000 (exponent=biased_exponent-$4000)
  } ext;
  BYTE Mantissa[7]; //(packed BCD ?.??????????????, ?=4 bits=1 digit)
  token $23;
} TAG_FLOAT;
```

Type Recognition (varlink.o)

Here is how VAR link recognises files

Name	description	File (0x)	SAVETYPE
EXPR	expression	..	EQ_TAG=0x87
STR	string	..2D	STRING_TAG=0x2D
LIST	list	..D9	LIST_TAG=0xD9
MAT	matrix	..D9D9	MATRIX_TAG=0xDB
PRGM	program	..19E4E5..DC	PRGM_ITAG=0x19
FUNC	function	..DC	FUNC_ITAG=0x17
DATA	data	..DD	DATA_TAG=0xDD
GDB	Graph Database	..DE	GDB_TAG=0xDE
PIC	Picture	..DF	PIC_TAG=0xDF
TEXT	Text	..E0	TEXT_TAG=0xE0
FIG	Geometry Figure	..E1	FIG_TAG=0xE1
MAC	Geometry Macro	..E2	MAC_TAG=0xE2
ASM	PLUS Assembler file	..F3	ASM_TAG=0xF3
OTH	PLUS Other	..F8	OTH_TAG=0xF8

Note "…DB" is also detected as a Matrix in files, although is not actually valid, as it is never used in variables.

Symbol Tables (sym.o)

Variables are usually stored from an expression, therefore most of these routines take their input in this form, i.e. as pointer to the terminating zero of a zero starting and terminated string. I will refer to these as sym's (or Tok values). When refering to standard C strings, Str is used.

```
#define $(n) n+sizeof(n)

//Apologies for the pattern:
const char *foldertable = (char*){0,0x7F,0}; /* diamond */
#define FOLDER_TABLE $(foldertable)
const char *mainfolder = "\0main";
#define MAIN_FOLDER $(mainfolder)
const char *datafolder = "\09998";
#define DATA_FOLDER $(datafolder)
const char *geomfolder = "\09999";
#define GEOM_FOLDER $(geomfolder)
const char *Backslash = "\";
```

```
//OPEN=HIDDEN
#define SETUP_FAST_DEREF register scratchHeap = Heap;
#define DEREF(h) Heap[h]
#define FDEREF(h) scratchHeap[h]
#define DEREF_SYM(hsym) DerefSym(hsym)
#define FDEREF_SYM(hsym) (FDEREF[hsym.h] + hsym.?disp)

enum FolderOpFlags { FOP_UNLOCK=0, FOP_LOCK=1, FOP_ALL_FOLDERS=0x80 };

enum CompatFlags { CF_NONE=0, CF_CONVERT=1, CF_ENHANCED=2, CF_NEW=3 };
enum SymFlags { SF_STATVAR=0x0004, SF_LOCKED=0x0008, SF_OPEN=0x0010,
            SF_CHECKED=0x0020, SF_OVERWRITTEN=0x0040, SF_FOLDER=0x0080,
            SF_ARCHIVED=0x0200, SF_TWIN=0x0400, SF_LOCAL=0x4000,
            SF_BUSY=0x8000 };
```

```
typedef struct SymStruct {
  char Name[8];   /* SymName (lowercase, null padded) */
#ifdef PLUS
  WORD Compat;
#endif
  WORD Flags;     /* SymFlags */
  HANDLE hVal;
} SYM_ENTRY;

typedef struct {
  HANDLE h;       /* handle of folder */
  WORD d;         /* displacement of symbol */
} HSYM;

typedef struct {
  SYM_ENTRY *First;
  SYM_ENTRY *Last;
  SYM_ENTRY *Cur;
} SYM_FIND;
```

```
HSYMS are used as a safe and efficient method for passing variable references around the
system. So that there are no probems after
garbage collection.
```

```
typedef struct {
  WORD alloc; //maximum number of entries before handle has to be resized
  WORD used;
  SYM_ENTRY entry[used]; // alphabetical order
} SYM_TABLE;
```

```
EStackIndex TokenizeSymName(const char *src, WORD flags);
#define TokenizeSymName ASAP_ref(128, EStackIndex (*const)(const char *, WORD))
/*
```

```
flags:
.0: don't append current folder if none specified
.1: allow system variable (0 - er throw)
.2: pass on errors
*/


BOOL SymInit( void );
/* Returns FALSE if unable to allocate tables */

EStackIndex *FolderAddTemp(void);
#define FolderAddTemp ASAP_ref(115, EStackIndex *(*const)(void))
/* returns SymName of new folder */

void FolderDelTemp(void);
#define FolderDelTemp ASAP_ref(116, void (*const)(void))

EStackIndex *TempFolderName(WORD TempNum);
#define TempFolderName ASAP_ref(118, EStackIndex *(*const)(WORD))
/* returns SymName of current folder (used in Folder*Temp) */

void FolderDelAllTemp(WORD start);
#define FolderDelAllTemp ASAP_ref(117, void (*const)(WORD))
/* Delete all temp folders from start to 0x2000 */

HSYM AddSymToFolder(const EStackIndex SymName, const EStackIndex FolderName);
#define AddSymToFolder ASAP_ref(112, HSYM (*const)(const EStackIndex, const EStackIndex))
/* Adds sym to folder, if already exists it is overwritten (i.e. HeapFree called) */

HSYM FindSymInFolder(const EStackIndex SymName, const EStackIndex FolderName);
#define FindSymInFolder ASAP_ref(113, HSYM (*const)(const EStackIndex, const
EStackIndex))

BOOL FolderCurTemp(const EStackIndex SymName);
#define FolderCurTemp ASAP_ref(114, BOOL (*const)(const EStackIndex))
/* Sets DefTemp handle to new temp folder with given name
returns success
*/

HSYM SymAdd(const EStackIndex SymName);
#define SymAdd ASAP_ref(92, HSYM (*const)(const EStackIndex))
/* calls SymAdd_aux: adds symbol "folder\var" */

HSYM SymAddMain(const EStackIndex SymName);
#define SymAddMain ASAP_ref(93, HSYM (*const)(const EStackIndex))

HSYM SymAddTwin(const EStackIndex SymName);
#define SymAddTwin ASAP_ref(639, HSYM (*const)(const EStackIndex))

HSYM SymFind(const EStackIndex SymName);
#define SymFind ASAP_ref(96, HSYM (*const)(const EStackIndex))
/* calls SymFindPtr; MakeHsym */

HSYM SymFindMain(const EStackIndex SymName);
#define SymFindMain ASAP_ref(97, HSYM (*const)(const EStackIndex))

HSYM SymFindHome(const EStackIndex SymName);
#define SymFindHome ASAP_ref(98, HSYM (*const)(const EStackIndex))

BOOL SymDel(const EStackIndex SymName);
#define SymDel ASAP_ref(94, BOOL (*const)(const EStackIndex))

BOOL HSymDel(HSYM hsym);
#define HSymDel ASAP_ref(95, BOOL (*const)(HSYM))
/* Delete symbol referenced by hsym */

BOOL SymDelTwin(SYM_ENTRY *FirstSymEnry);
#define SymDelTwin ASAP_ref(640, BOOL (*const)(SYM_ENTRY *))
/* free's the handle SymPtr->hVal, resets Second Twin's flag, then removes the first */
```

```
BOOL SymMove(const EStackIndex SrcName, const EStackIndex DestName);
#define SymMove ASAP_ref(99, BOOL (*const)(const EStackIndex, const EStackIndex))
/* create Dest, then Del Src */

WORD FolderFind(const EStackIndex SymName);
#define FolderFind ASAP_ref(103, WORD (*const)(const EStackIndex))
/* Returns enum { MAIN_FOLDER=2, FOLDER_TABLE=3, NOT_FOUND=4, BAD_FOLDER=5 }; */

BOOL FolderCur(const EStackIndex SymName, BOOL nonSys);
#define FolderCur ASAP_ref(101, BOOL (*const)(const EStackIndex, BOOL))
/* Sets current folder, if nonSys set SymFindNext is called until on non-system var */

void FolderGetCur(char *buf0);
#define FolderGetCur ASAP_ref(104, void (*const)(char *))

HANDLE FolderAdd(const EStackIndex FolderName);
#define FolderAdd ASAP_ref(100, HANDLE (*const)(const EStackIndex))

BOOL FolderDel(EStackIndex SymName, WORD Flags);
#define FolderDel ASAP_ref(102, BOOL (*const)(EStackIndex, WORD))
/* if Flags==TRUE, delete is forced even if main folder (0x8000 - CheckGraphRef) */


SYM_ENTRY *SymFindFirst(const EStackIndex SymName, WORD Flags);
#define SymFindFirst ASAP_ref(108, SYM_ENTRY *(*const)(const EStackIndex, WORD))
SYM_ENTRY *SymFindNext(void);
#define SymFindNext ASAP_ref(109, SYM_ENTRY *(*const)(void))
SYM_ENTRY *SymFindPrev(void);
#define SymFindPrev ASAP_ref(110, SYM_ENTRY *(*const)(void))
char *SymFindFoldername(void);
#define SymFindFoldername ASAP_ref(111, char *(*const)(void))
/* Returns current Find Folder Name (main if not specified) */

SYM_ENTRY *DerefSym(HSYM hsym);
#define DerefSym ASAP_ref(121, SYM_ENTRY *(*const)(HSYM))
/* standard conversion of hsym to SYM_ENTRY * */

int FolderOp(const EStackIndex SymName, WORD Flags);
#define FolderOp ASAP_ref(105, int (*const)(const EStackIndex, WORD))
/* Locks or Unlocks given folder table, or all of them */

BOOL FolderRename(const EStackIndex src, const EStackIndex dest);
#define FolderRename ASAP_ref(106, BOOL (*const)(const EStackIndex, const EStackIndex))
WORD FolderCount(SYM_ENTRY *FolderEntry);
#define FolderCount ASAP_ref(107, WORD (*const)(SYM_ENTRY *))

EStackIndex StrToTokN(const char *src, EStackIndex dest);
#define StrToTokN ASAP_ref(123, EStackIndex (*const)(const char *, EStackIndex))
/* Tokenises single name to end of buffer */
BOOL TokToStrN(char *dest, const EStackIndex src);
#define TokToStrN ASAP_ref(124, BOOL (*const)(char *, const EStackIndex))
/* Detokenises single variable name source, to dest */

int SymCmp(const char *s1, const char * s2);
#define SymCmp ASAP_ref(129, int (*const)(const char *, const char *))
void SymCpy(char *dest, const char *src);
#define SymCpy ASAP_ref(130, void (*const)(char *, const char *))
void SymCpy0(char *dest, const char *src);
#define SymCpy0 ASAP_ref(131, void (*const)(char *, const char *))

BOOL ValidateSymName(const char *s);
#define ValidateSymName ASAP_ref(132, BOOL (*const)(const char *))
/* make sure characters are in correct range etc. */

BOOL CheckGraphRef(const SYM_ENTRY *SymEntry);
#define CheckGraphRef ASAP_ref(125, BOOL (*const)(const SYM_ENTRY *))

void CheckLinkLockFlag(const SYM_ENTRY *FuncSymEntry);
```

```
#define CheckLinkLockFlag ASAP_ref(127, void (*const)(const SYM_ENTRY *))
/* Sets/Clears LinkLock flag for function according to SF_LOCKED & SF_ARCHIVED */

void ClearUserDef(HANDLE hFuncVar);
#define ClearUserDef ASAP_ref(126, void (*const)(HANDLE))
/* Clears the function flags */

BOOL ParseSymName(const EStackIndex SymName);
#define ParseSymName ASAP_ref(120, BOOL (*const)(const EStackIndex))
/* Parses name Symname, stores appropriate Folder and Var sym names */

HSYM MakeHsym(HANDLE FolderHandle, const SYM_ENTRY *SymPtr);
#define MakeHsym ASAP_ref(642, HSYM (*const)(HANDLE, const SYM_ENTRY *))
/* Return an HSYM formed from SymPtr in FolderHandle */

BOOL HSYMtoName(HSYM hsym, char *buf20);
#define HSYMtoName ASAP_ref(122, BOOL (*const)(HSYM, char *))
/* store Name of hsym to buffer in form "folder\var" */

SYM_ENTRY *SymFindPtr(const EStackIndex SymName, WORD Flags);
#define SymFindPtr ASAP_ref(643, SYM_ENTRY *(*const)(const EStackIndex, WORD))

void LoadSymFromFindHandle( void );
#define LoadSymFromFindHandle ASAP_ref(641, void (*const)( void ))
/* calls LoadSymFromHandle for FindHandle, called before SymDelTwin etc. */

static void FindLoadSym( void );
/* Initialise SymFind vars */

static void LoadSymFromHandle(HANDLE hTable);
/* Initialise Sym vars for specified Table */

static void LoadSymDel(const SYM_ENTRY *SymEntry);
static SYM_ENTRY *LoadSymAdd(const char *Name);
static SYM_ENTRY *LoadSymFind(HANDLE hTable, const char *Name);

SYM_ENTRY *SymAdd_aux(const EStackIndex SymName, WORD Flags);

BOOL IsMainFolderStr(const char *s);
#define IsMainFolderStr ASAP_ref(119, BOOL (*const)(const char *))

static BOOL IsFolderTblStr(const char *s);

static void cleanup_aux(BOOL init);
void cleanup( void );

static void checkSysVar( void );

static void default_val(HANDLE h);
/* i.e. if h is current folder handle, restore it to "main". if DefTempHandle, set to 0
*/

static BOOL FindFlags(const SYM_ENTRY *Folder, WORD FlagMask)
/* Returns TRUE if any files in folder have a bit in FlagMask set */

static char ParseVarStr[9]; /* if =0 VarTok = $(ParseStr), FolderTok = $(FolderCurStr) */
static char ParseStr[9];    /* else VarTok = $(ParseVarStr), FolderTok = $(ParseStr) */
static char _TempFolderName[6];
static HANDLE InitTempHandle;
static char aTokToStr[9];
static LONG LoadCurSymNum;
static WORD LoadMaxSymNum;

static SYM_ENTRY *LoadCurSym, *LoadLastSym, *LoadFirstSym;
static WORD FindFlags;

WORD SymTempFolCount;
EStackIndex ParseFolderTok;
EStackIndex ParseVarTok;
```

```
HANDLE FindFolderHandle;      /* Actual Find Folder */
HANDLE FolderTblHandle;
HANDLE FolderMainHandle;
HANDLE DefTempHandle;
HANDLE CurFolderHandle;       /* Current OS Folder handle */

ER_CODE SymError;
BOOL bCleanupSysVars;
HANDLE FindTblHandle;         /* Folder handle, FolderTblHandle if none */
char CurFolderStr[9];         /* Current OS Folder name */
SYM_FIND Find, FolderFind;    /* Find used for Vars & Folders, copied to FolderFind */
```

When archived programs are executed EM_twinSymFromExtMem is called, which:

1. copies the variable to a new handle in RAM
2. copies the sym table entry (to one slot before) to point to this new handle (flags SF_TWIN)
3. sets EM_bTwinDelete

If the program executes without error everything is cleaned up when the command has finished at the end of the event loop.

each token has an associated compatibilty flag
only see 0,1,3 in SYM_ENTRY, for sending purposes:
0 sends immediatedly
1 converts and sends
3 will not send

//2 is an enhanced token and may or may not work...a compatibilty number of 1 or 3 is determined from its context

Variable Interface (store.o)

Variable storing and recalling. It is much easier to use VarStore and and associated routines than symbol tables (above), VarStore for example will take a variable name and data from an EStackIndex allocate a memory block for the data, and add the relevant symbol table entries for you. For specific info see end of section.

```
#ifdef PLUS
struct SysVarSym {
  WORD Size;
  BYTE Any[60];
  SYM_ENTRY sysVar;
  SYM_ENTRY tblInput;
  SYM_ENTRY unit;
};
#else
struct SysVarSym {
  WORD Size; //10
  TAG_FLOAT Val;
  SYM_ENTRY sysVar;
  SYM_ENTRY tblInput;
};
#endif
```

```
enum MemberStuctId { StatVar=1, GraphVars=2, GraphZoomVar=3, SysFloat=4, GenVar=6,
Table=7, FldPic?=8};
struct SysTokenStruct {
  BYTE MemberStructID;
  BYTE ModeFlags;
  BYTE Index;
  BYTE Compat;
  char *Name;
};
```

```
enum StoreFlags {
  STOF_ESI=0x4000,
  STOF_ELEM_STORE=0x4001,
```

```
  STOF_ELEM_DEL=0x4002,
  STOF_HESI=0x4003,
  STOF_CREATE_FOLDER=0x1000, //for SymFindPtr
};

const char * sysdata = "\0sysdata";
#define SYSDATA_VAR $(sysdata)
const char * regeq = "\0regeq";
#define REGEQ_VAR $(regeq)
const char * regcoef = "\0regcoef";
#define REGCOEF_VAR $(regcoef)
const char * fldpic = "\0fldpic";
#define FLDPIC_VAR $(fldpic)

static const EStackIndex const VarTok [3] = //GenVar (Data/StatVar Variables)
  { SYSDATA_VAR, REGEQ_VAR, REGCOEF_VAR };
static const char * const xt = "xt";
static const char * const nxt = "nxt";

static const DWORD t=0; //?
static const char * const x = "x";

const SysTokenStruct SysTokenTable[0x72];

HSYM VarRecall(EStackIndex Var, WORD deFlags);
#define VarRecall ASAP_ref(133, HSYM (*const)(EStackIndex, WORD))

HSYM VarStore(EStackIndex Var, WORD Flags, ...);
#define VarStore ASAP_ref(134, HSYM (*const)(EStackIndex, WORD, ...))

FLOAT GetFloat(EStackIndex esi);
/* gets a float from esi (calling NG_approxESI if immediate)
throws "Domain error" if can't resolve float
*/

static void SelectGR3Func(WORD num);
/* Used by VarStore to select GR3 func (and deselect others) */

void VarGraphRefBitsClear( void );
#define VarGraphRefBitsClear ASAP_ref(616, void (*const)( void ))

BOOL QSysProtected(BYTE tag);
#define QSysProtected ASAP_ref(136, BOOL (*const)(BYTE))
/* checks whether the end tag is a protected type */

BOOL CheckSysFunc(const char *var, WORD *offset);
#define CheckSysFunc ASAP_ref(137, BOOL (*const)(const char *, WORD *))
/* stores found offset */

BOOL CheckReservedName(EstackIndex esi);
#define CheckReservedName ASAP_ref(139, BOOL (*const)(EstackIndex))
/* Calls TokToStrN, CheckSysFunc */

BOOL SymSysVar(const char *var);
#define SymSysVar ASAP_ref(140, BOOL (*const)(const char *))
/* Calls CheckSysFunc, compares sysdata, regcoef */

void VarInit( void );
#define VarInit ASAP_ref(617, void (*const)( void ))

void ResetSymFlags( WORD Flags);
#define ResetSymFlags ASAP_ref(142, void (*const)(WORD))
/* Clears Flags in all vars & varsym */

HSYM VarStoreLink(EStackIndex esi, HANDLE *DestVar, WORD *Status);
#define VarStoreLink ASAP_ref(135, HSYM (*const)(EStackIndex, HANDLE *, WORD *))
/* Status may be set to 0 if error occurs */

//Sets the VarSym structure to given values
```

```
static HSYM RclSysFloat(FLOAT f);
static HSYM RclSysVar(HANDLE h, WORD symnum);
static HSYM RclUnit(WORD unitnum);

//Get address of system variables from tags:
static FLOAT *GetTableSetVar(BYTE tag); //tblStart, _DELTA_tbl
static FLOAT *GetSysVar(BYTE tag);

static void VarGraphRefBitsSet( void );

WORD GetSysGraphRef(WORD num);
#define GetSysGraphRef ASAP_ref(138, WORD (*const)(WORD))
/* get GraphRef from graph num: 0 active, 1 other */

static WORD ListDim(EStackIndex esi, WORD *size);
/* returns num elements, sets size to byte size */

HSYM ValidateStore(HSYM Var, BYTE Flags);
#define ValidateStore ASAP_ref(141, HSYM (*const)(HSYM, BYTE))

//($5DBC,$609C,$788C)
static FLOAT ok;
static FLOAT errornum;

static HANDLE hSysVarSym;
static BYTE GraphRefBits;

BOOL bGraphESI;
```

VarStore operations

VarStore may appear to perform trivial operations, but it does automate many common tasks. It provides a consistent interface to all of the system variables.

VarRecall returns an HSYM to any variable requested even system variables.
VarStore performas many storing operations, from basic file saving, to list/matrix element access

VarStore(EStackIndex Var, WORD Flags, …);

Enum val	Function parameters
Store pop value from estack	VarStore(ESI, 0x4001, size = 0, ESI);
Store pop esi to element	VarStore(ESI, 0x4001, [size], ESI, col, row);
Delete element from var (pos)?	VarStore(ESI, 0x4002);
Store copy of STORE var	VarStore(ESI, 0x4003, [size], h);
Standard variable Tag	VarStore(ESI, tag, size, *start);

Delete is is pretty specific to the variable referred to, for example when tblInput is the variable “table_win_vars” is used to get the current element number, which is then removed.

VarRecall - SysVarSym:

System variables are not stored consistently, however, they are always referred to exactly as normal variables, and generally
passed through HSYMS.
To allow this a special structure is exported from store.o consisiting of a buffer and self-referring SYM_ENTRIES. If VarRecall
is given the name of a system variable, it finds the address and copies the variable into the SysVarSym buffer. An HSYM to the
relevelent SYM_ENTRY is then returned.
On the ti92 is is only used for float variables, these are converted to 14dp. On the plus, units are also sent via this structure.

More freely, it is observered in the following form

```
#ifdef PLUS
#define hVARSYM $D
#else
#define hVARSYM $A
#endif

struct SysVarSym {
#ifdef PLUS
  STORE EXPR tmp; //60 bytes reserved ($3E)
#else
  STORE TAG_FLOAT tmp;
#endif
  SYM_ENTRY sysVar={var_name, CF_OLD, SF_LOCKED, hVARSYM}; //($C,$3E)
  SYM_ENTRY seTblInput={"tblInput", CF_OLD, SF_LOCKED, htblInput};
#ifdef PLUS
  SYM_ENTRY unit={"", CF_NEW, SF_LOCKED, hVARSYM};
#endif
};
```

Conversions and Execution

- Interface to Parser and Interpreter
- Backwards Compatibility
- Execution
- Parser
- Display and Detokenization

Interface to Parser and Interpreter (NG.O)

```
//NG_Control:
//NG_Approx=0x0004 //2
//NG_Exact=0x0008  //3

//NG_return:
//2: Func
//1: Prgm

HANDLE NG_RPNToText(HANDLE hRPN, WORD width, BOOL torf);
#define NG_RPNToText ASAP_ref(603, HANDLE (*const)(HANDLE, WORD, BOOL))
/* Detokenises hRPN,
Returns HANDLE of allocated TEXT, throws an ER_MEMORY if it fails
Calls: dispexr.o; display_statements
*/

BOOL DecodeFloat(const char *source, FLOAT *dest);
/* Converts text representation of string to float
Returns success
Calls: push_parse_text
*/

void NG_approxESI(EStackIndex esi);
#define NG_approxESI ASAP_ref(604, void (*const)(EStackIndex))
/* execute tokenised statement at esi
sets flags to approximate
*/

void NG_rationalESI(EStackIndex esi);
#define NG_rationalESI ASAP_ref(607, void (*const)(EStackIndex))
*/ execute tokenised statement at esi
sets flags to exact
*/

void NG_execute(HANDLE hTok, BOOL approx);
#define NG_execute ASAP_ref(605, void (*const)(HANDLE, BOOL))
/* Execute tokenisted statement
```

```
This routine is called after NG_RPNToText from the home screen
calls push_simplify_statements
*/

BOOL NG_tokenize(HANDLE hText, ER_CODE *errCode, WORD *errOffset);
#define NG_tokenize ASAP_ref(608, BOOL (*const)(HANDLE, WORD *, WORD *))
/* Tokenises hText to estack
Returns success, if no error errCode contains multi state;
if there is an error errCode is set and errOffset containes offset in text where error
occurred, otherwise undefined
Calls: parser; push_parse_text
(Home screen calls HS_popEStack to pop result to handle)
*/

BOOL NG_all_vars_defined(EStackIndex esi);

void NG_graphESI(EStackIndex esi, HANDLE h);
#define NG_graphESI ASAP_ref(606, void (*const)(EStackIndex, HANDLE))
/* Execute function at esi and push the result
tokenises, executes, checks vars with NG_all_vars_defined
*/

void NG_setup_graph_fun(HSYM funcText, BYTE funcNum, HANDLE *h);
#define NG_setup_graph_fun ASAP_ref(609, void (*const)(HSYM, BYTE, HANDLE *))
void NG_cleanup_graph_fun(HSYM funcNum, HANDLE tmp);
#define NG_cleanup_graph_fun ASAP_ref(610, void (*const)(HSYM, HANDLE))

void push_END_TAG( void );
#define push_END_TAG ASAP_ref(611, void (*const)( void ))
/* push_quantum(END_TAG);
*/
void push_LIST_TAG( void );
#define push_LIST_TAG ASAP_ref(612, void (*const)( void ))
/* push_quantum(LIST_TAG);
*/

WORD SE_offset;


//MO_flags:
//MO_pretty=0x0001;
```

Backwards Compatibility (Ncompat.o?)

Conversion of variables from the TI92+/TI89 to the TI92. Sym tables and the token tables contain a compatibilty flag, which is used by these routines to determine whether or not a variable can be transferred from the PLUS calculators to the non PLUS version. The incompatibilites are mainly do to local variable references are handled differently on the PLUS, there are some new tokens, and some are extended (these are the different compatiblity codes – see Symbol Tables (sym.o))

```
static ESI convert_local_vars(ESI,ESI);

void convert_to_TI_92(HANDLE h);
#define convert_to_TI_92 ASAP_ref(600, void (*const)(HANDLE))

static convert_local_vars_multi

gen_version
#define gen_version ASAP_ref(601, ASAP_UNDEF)
tokenize_if_TI_92_or_text
#define tokenize_if_TI_92_or_text ASAP_ref(613, ASAP_UNDEF)
is_executable
#define is_executable ASAP_ref(602, ASAP_UNDEF)
```

Execution

- Simplification and Instructions
- Interpreter
- Instruction Handling

Simplification and Instructions (program.o)

Somewhat of a strange mix you might think, but this is only the given name. It really just calls the interpreter to do most of the work, and merely deals with the instruction tokens.
Program or instruction tokens are available in TI-BASIC programs only. The CmdFunc entries are usually named cmd_*, and the paramaters are usually the EStackIndex's of the paramaters listed in the manual, returning void.
ParmFunc is used to validate the paramaters before CmdFunc is called.

push_simplify_statements calls interpret_statements (immediately below).

```
struct InstrTokenStruct {
  BYTE Compat;
  BYTE Parms;
  BYTE DefParms;
  // alignment
  void (*ParmFunc)(EStackIndex esi);
  void (*CmdFunc)(…);
  char *Name;
};
```

```
void push_simplify_statements(EStackIndex esi);
```

```
const InstrTokenStruct InstructionTable[0x98];
```

Interpreter (interpret.o)

PushFunc here is like CmdFunc in program.o above, entries are usually names push_*.

```
struct TokenStruct {
  BYTE Type;
  BYTE Compat;
  BYTE Parms;
  BYTE DefParms;
  void (*PushFunc)(EStackIndex esi);
  char *Name;
};

struct InterpretStruct {
  EStackIndex OldTop;
  void (*PushFunc)(EStackIndex esi);
};
```

```
#ifdef PLUS //all handled in interpret_statement on previous versions
//Entries from the jump tables Token prefixed with _:
//non direct (non-data) links are not prefixed as such
void _arclen(EStackIndex esi);
void _nint(EStackIndex esi);
void _store(EStackIndex esi);
void _exp_primed(EStackIndex esi);
void _func_primed(EStackIndex esi);
void _with(EStackIndex esi);
void _exact(EStackIndex esi);
void _exp2list(EStackIndex esi);
void _det(EStackIndex esi);
void _ref(EStackIndex esi);
void _rref(EStackIndex esi);
void _simult(EStackIndex esi);
void _system_unknown_F5(EStackIndex esi);
```

```
void _system_unknown_F6(EStackIndex esi);
void _system_unknown_F7(EStackIndex esi);
void _randpoly(EStackIndex esi);
void _expand(EStackIndex esi);
void factor(EStackIndex esi);
void _factor(EStackIndex esi);
void _cfactor(EStackIndex esi);
void _integrate(EStackIndex esi);
void _diff(EStackIndex esi);
void _avgrc(EStackIndex esi);
void _nderiv(EStackIndex esi);
void _taylor(EStackIndex esi);
void _limit(EStackIndex esi);
void _DMS(EStackIndex esi);
void _PN(EStackIndex esi);
void _PN_expr(EStackIndex esi);
void _complex(EStackIndex esi);
void _seq(EStackIndex esi);
void _rand(EStackIndex esi);
void _list(EStackIndex esi);
void _user_func(EStackIndex esi);
void inv_trig(EStackIndex esi1, EStackIndex old);
void _var(EStackIndex esi);
void _rational //int, or fraction
void _float(EStackIndex esi);
void _exp(EStackIndex esi);
void _im(EStackIndex esi);
void simple_power(EStackIndex esi); //non element-wise
```

```
void _ext_tag(EStackIndex esi);
/* Process extended tokens (ExtTokens) 0xE3 */
#endif
```

```
void interpret_statements(EStackIndex esi);
void solve_statement?(EStackIndex esi);
```

```
#ifdef PLUS
const TokenStruct Tokens[0xFA];
const TokenSturct ExtTokens[0x36];
#endif
```

Instruction Handling

These are not yet documented – Various routines from the Instruction table.

- basicops
- hmatrix
- instructions

basicops.o

```
push_degrees
#define push_degrees ASAP_ref(788, ASAP_UNDEF)
push_format
#define push_format ASAP_ref(789, ASAP_UNDEF)
push_instring
#define push_instring ASAP_ref(794, ASAP_UNDEF)
push_part
#define push_part ASAP_ref(796, ASAP_UNDEF)
push_str_to_expr
#define push_str_to_expr ASAP_ref(805, ASAP_UNDEF)
push_string
#define push_string ASAP_ref(806, ASAP_UNDEF)
push_to_cylin
#define push_to_cylin ASAP_ref(808, ASAP_UNDEF)
```

```
did_push_to_polar
#define did_push_to_polar ASAP_ref(787, ASAP_UNDEF)
push_to_sphere
#define push_to_sphere ASAP_ref(809, ASAP_UNDEF)
cmd_clrerr
#define cmd_clrerr ASAP_ref(814, ASAP_UNDEF)
cmd_cycle
#define cmd_cycle ASAP_ref(824, ASAP_UNDEF)
cmd_else
#define cmd_else ASAP_ref(837, ASAP_UNDEF)
cmd_endfor
#define cmd_endfor ASAP_ref(838, ASAP_UNDEF)
cmd_endloop
#define cmd_endloop ASAP_ref(839, ASAP_UNDEF)
cmd_endtry
#define cmd_endtry ASAP_ref(840, ASAP_UNDEF)
cmd_endwhile
#define cmd_endwhile ASAP_ref(841, ASAP_UNDEF)
cmd_exit
#define cmd_exit ASAP_ref(842, ASAP_UNDEF)
cmd_for
#define cmd_for ASAP_ref(847, ASAP_UNDEF)
cmd_goto
#define cmd_goto ASAP_ref(850, ASAP_UNDEF)
cmd_if
#define cmd_if ASAP_ref(852, ASAP_UNDEF)
cmd_ifthen
#define cmd_ifthen ASAP_ref(853, ASAP_UNDEF)
cmd_local
#define cmd_local ASAP_ref(862, ASAP_UNDEF)
cmd_passerr
#define cmd_passerr ASAP_ref(874, ASAP_UNDEF)
cmd_return
#define cmd_return ASAP_ref(901, ASAP_UNDEF)
cmd_try
#define cmd_try ASAP_ref(919, ASAP_UNDEF)
cmd_while
#define cmd_while ASAP_ref(922, ASAP_UNDEF)
```

hmatrix.o

```
push_mrow_aux
#define push_mrow_aux ASAP_ref(795, ASAP_UNDEF)
push_rand
#define push_rand ASAP_ref(799, ASAP_UNDEF)
push_randpoly
#define push_randpoly ASAP_ref(800, ASAP_UNDEF)
cmd_cubicreg
#define cmd_cubicreg ASAP_ref(820, ASAP_UNDEF)
cmd_expreg
#define cmd_expreg ASAP_ref(843, ASAP_UNDEF)
cmd_fill
#define cmd_fill ASAP_ref(844, ASAP_UNDEF)
cmd_lnreg
#define cmd_lnreg ASAP_ref(861, ASAP_UNDEF)
cmd_linreg
#define cmd_linreg ASAP_ref(860, ASAP_UNDEF)
cmd_medmed
#define cmd_medmed ASAP_ref(865, ASAP_UNDEF)
cmd_onevar
#define cmd_onevar ASAP_ref(872, ASAP_UNDEF)
cmd_twovar
#define cmd_twovar ASAP_ref(920, ASAP_UNDEF)
cmd_powerreg
#define cmd_powerreg ASAP_ref(879, ASAP_UNDEF)
cmd_quadreg
#define cmd_quadreg ASAP_ref(894, ASAP_UNDEF)
cmd_quartreg
#define cmd_quartreg ASAP_ref(895, ASAP_UNDEF)
```

```
cmd_sinreg
#define cmd_sinreg ASAP_ref(908, ASAP_UNDEF)
cmd_logistic
#define cmd_logistic ASAP_ref(864, ASAP_UNDEF)
cmd_randseed
#define cmd_randseed ASAP_ref(896, ASAP_UNDEF)
cmd_sorta
#define cmd_sorta ASAP_ref(910, ASAP_UNDEF)
cmd_sortd
#define cmd_sortd ASAP_ref(911, ASAP_UNDEF)
```

instructions.o

```
GraphActivate
#define GraphActivate ASAP_ref(508, ASAP_UNDEF)
push_gettype
#define push_gettype ASAP_ref(793, ASAP_UNDEF)
push_pttest
#define push_pttest ASAP_ref(797, ASAP_UNDEF)
push_pxltest
#define push_pxltest ASAP_ref(798, ASAP_UNDEF)
push_getfold
#define push_getfold ASAP_ref(791, ASAP_UNDEF)
push_setfold
#define push_setfold ASAP_ref(801, ASAP_UNDEF)
cmd_andpic
#define cmd_andpic ASAP_ref(810, ASAP_UNDEF)
cmd_circle
#define cmd_circle ASAP_ref(812, ASAP_UNDEF)
cmd_copyvar
#define cmd_copyvar ASAP_ref(819, ASAP_UNDEF)
cmd_custom
#define cmd_custom ASAP_ref(823, ASAP_UNDEF)
cmd_delfold
#define cmd_delfold ASAP_ref(826, ASAP_UNDEF)
cmd_delvar
#define cmd_delvar ASAP_ref(827, ASAP_UNDEF)
cmd_dialog
#define cmd_dialog ASAP_ref(828, ASAP_UNDEF)
cmd_drawfunc
#define cmd_drawfunc ASAP_ref(833, ASAP_UNDEF)
cmd_drawinv
#define cmd_drawinv ASAP_ref(834, ASAP_UNDEF)
cmd_drawparm
#define cmd_drawparm ASAP_ref(835, ASAP_UNDEF)
cmd_drawpol
#define cmd_drawpol ASAP_ref(836, ASAP_UNDEF)
cmd_cyclepic
#define cmd_cyclepic ASAP_ref(825, ASAP_UNDEF)
cmd_line
#define cmd_line ASAP_ref(856, ASAP_UNDEF)
cmd_linehorz
#define cmd_linehorz ASAP_ref(857, ASAP_UNDEF)
cmd_linetan
#define cmd_linetan ASAP_ref(858, ASAP_UNDEF)
cmd_linevert
#define cmd_linevert ASAP_ref(859, ASAP_UNDEF)
cmd_lock
#define cmd_lock ASAP_ref(863, ASAP_UNDEF)
cmd_movevar
#define cmd_movevar ASAP_ref(866, ASAP_UNDEF)
cmd_newfold
#define cmd_newfold ASAP_ref(868, ASAP_UNDEF)
cmd_newpic
#define cmd_newpic ASAP_ref(869, ASAP_UNDEF)
cmd_newplot
#define cmd_newplot ASAP_ref(870, ASAP_UNDEF)
cmd_plotsoff
#define cmd_plotsoff ASAP_ref(876, ASAP_UNDEF)
```

cmd_plotson
#define cmd_plotson ASAP_ref(877, ASAP_UNDEF)
cmd_popup
#define cmd_popup ASAP_ref(878, ASAP_UNDEF)
cmd_ptchg
#define cmd_ptchg ASAP_ref(882, ASAP_UNDEF)
cmd_ptoff
#define cmd_ptoff ASAP_ref(883, ASAP_UNDEF)
cmd_pton
#define cmd_pton ASAP_ref(884, ASAP_UNDEF)
cmd_pttext
#define cmd_pttext ASAP_ref(885, ASAP_UNDEF)
cmd_pxlcircle
#define cmd_pxlcircle ASAP_ref(887, ASAP_UNDEF)
cmd_pxlchg
#define cmd_pxlchg ASAP_ref(886, ASAP_UNDEF)
cmd_pxlhorz
#define cmd_pxlhorz ASAP_ref(888, ASAP_UNDEF)
cmd_pxlline
#define cmd_pxlline ASAP_ref(889, ASAP_UNDEF)
cmd_pxloff
#define cmd_pxloff ASAP_ref(890, ASAP_UNDEF)
cmd_pxlon
#define cmd_pxlon ASAP_ref(891, ASAP_UNDEF)
cmd_pxltext
#define cmd_pxltext ASAP_ref(892, ASAP_UNDEF)
cmd_pxlvert
#define cmd_pxlvert ASAP_ref(893, ASAP_UNDEF)
cmd_rclpic
#define cmd_rclpic ASAP_ref(898, ASAP_UNDEF)
cmd_rename
#define cmd_rename ASAP_ref(899, ASAP_UNDEF)
cmd_request
#define cmd_request ASAP_ref(900, ASAP_UNDEF)
cmd_rplcpic
#define cmd_rplcpic ASAP_ref(902, ASAP_UNDEF)
cmd_shade
#define cmd_shade ASAP_ref(906, ASAP_UNDEF)
cmd_showstat
#define cmd_showstat ASAP_ref(907, ASAP_UNDEF)
cmd_slpline
#define cmd_slpline ASAP_ref(909, ASAP_UNDEF)
cmd_stopic
#define cmd_stopic ASAP_ref(913, ASAP_UNDEF)
gr_stopic
#define gr_stopic ASAP_ref(503, ASAP_UNDEF)
cmd_text
#define cmd_text ASAP_ref(916, ASAP_UNDEF)
cmd_toolbar
#define cmd_toolbar ASAP_ref(917, ASAP_UNDEF)
cmd_trace
#define cmd_trace ASAP_ref(918, ASAP_UNDEF)
cmd_unlock
#define cmd_unlock ASAP_ref(921, ASAP_UNDEF)
cmd_xorpic
#define cmd_xorpic ASAP_ref(923, ASAP_UNDEF)
cmd_zoombox
#define cmd_zoombox ASAP_ref(924, ASAP_UNDEF)
cmd_zoomdata
#define cmd_zoomdata ASAP_ref(925, ASAP_UNDEF)
cmd_zoomdec
#define cmd_zoomdec ASAP_ref(926, ASAP_UNDEF)
cmd_zoomfit
#define cmd_zoomfit ASAP_ref(927, ASAP_UNDEF)
cmd_zoomin
#define cmd_zoomin ASAP_ref(928, ASAP_UNDEF)
cmd_zoomint
#define cmd_zoomint ASAP_ref(929, ASAP_UNDEF)
cmd_zoomout

```
#define cmd_zoomout ASAP_ref(930, ASAP_UNDEF)
cmd_zoomprev
#define cmd_zoomprev ASAP_ref(931, ASAP_UNDEF)
cmd_zoomrcl
#define cmd_zoomrcl ASAP_ref(932, ASAP_UNDEF)
cmd_zoomsqr
#define cmd_zoomsqr ASAP_ref(933, ASAP_UNDEF)
cmd_zoomstd
#define cmd_zoomstd ASAP_ref(934, ASAP_UNDEF)
cmd_zoomsto
#define cmd_zoomsto ASAP_ref(935, ASAP_UNDEF)
cmd_zoomtrig
#define cmd_zoomtrig ASAP_ref(936, ASAP_UNDEF)
QSkipGraphErr
#define QSkipGraphErr ASAP_ref(487, ASAP_UNDEF)
```

Parser (parser.o)

Main entry point is push_parse_text, this tokenises text so that it can be executed.

```
void nonblank( void );
#define nonblank ASAP_ref(968, void (*const)( void ))
/* if on blank (whitespace) move to nonblank character */

void nextnonblank( void );
/* move to next non-blank
same as above but always moves on at least one char */

BOOL independent?(EStackIndex expr1, EStackIndex expr2);
BOOL QBinaryDigit(char c); //'0' || '1'
BOOL QHexDigit(char c);
void push_parse_integer(const char *src, WORD Radix);

WORD parse_array(TAG tag1, TAG tag2, ER_CODE errorCode);
/* Returns num elements, errorcode is thrown if tag1 && tag2 not matched */

WORD parse_vector(BYTE, BYTE, EStackIndex);
void push_parse_vector( void );
void push_func_call_strict(WORD elements); /* throws error if wrong number */
void push_func_call(BOOL fewArgThrow, BOOL manyArgThrow, BOOL AllowDefArgs);
push_multi?

void push_var(const char *start, const char *end);
#define push_var ASAP_ref(971, void (*const)(const char *, const char *))

void match_start_block(TAG start, BYTE type);
BOOL is_used_symbol(BYTE char);
BOOL is_bcd?( void );

int cmpstri(const char *str1, const char *str2);
#define cmpstri ASAP_ref(367, int (*const)(const char *, const char *))

void get_token_from_name( void ) /* does string table lookup */
void get_token_from_symbol( void );
BOOL get_var_name( BYTE start ); /* start = "_" */

ER_CODE next_token(BOOL errorthrow);
#define next_token ASAP_ref(967, ER_CODE (*const)(BOOL errorthrow))
/* move to the next token
returns the errorcode, if any, and optionally throws the errors
*/

static void push_string( void );
static void push_subscript( void );

BOOL is_pathname(const EStackIndex name);
#define is_pathname ASAP_ref(966, BOOL (*const)(const EStackIndex))
```

```
static WORD is_var_local(const EStackIndex varname);

static BOOL push_parse_main(BOOL);
static void push_parse_factor(BOOL);
static void push_parse_term(BOOL); //+,-,...
static void push_parse_not(BOOL);
static void push_parse_and(BOOL);
static void push_parse_or(BOOL);
static void push_parse_with(BOOL);
static void push_parse_conv(*)
BOOL push_parse_comment( void );
static void push_parse_local(EStackIndex esi);
static void push_parse_var(EStackIndex, BOOL compare);

static void push_parse_text_aux(BYTE block, BYTE pass, BYTE TypeTag, BOOL call?);
/* state on of P_STATE enum, TypeTag = 0x17 when parsing func etc.
PB_CUSTOM = 6,
PB_DIALOG = 8,
PB_TOOLBAR = 0x1D,
PB_TRY = 0x1F
PB_IF = 0x39
PB_ELSE = 0x3B,
*/

void fix_loop_displacements(EStackIndex esi);
#define fix_loop_displacements ASAP_ref(368, void (*const)(EStackIndex))

static BOOL parse_fix(EStackIndex esi);
/* Called after parsing to check expression
calls fix_loop_displacements if necessary
returns TRUE if a multi expression (multi, store, instruction tag present)
*/

BOOL push_parse_text(const char *text);
#define push_parse_text ASAP_ref(970, ER_CODE (*const)(const char *))
/* push parse (tokenize) text onto the estack
returns multi status from parse_fix
throws error (Syntax and others) on errors
*/

void push_parse_prgm_or_func_text(const char *text, EStackIndex esi, BOOL cmp);
#define push_parse_prgm_or_func_text ASAP_ref(969, (*const)(const char *, EStackIndex,
BOOL))
```

Display and Detokenization

Parsing of expressions into graphical (2D) or text (1D) forms.

- Pretty Printing
- Text Conversion

Pretty Printing (2dexpr.o)

```
Parse2DExpr sets up various blocks of this form, these are then displayed by Print2DExpr.

The whole group is parsed like a list of calls e.g.
  PP_TYPE y, x, yDescent, yAscent, Width[, CHAR|{ WORD, TEXT}]
```

01	PP_LTEXT WORD y, WORD x, WORD yD, WORD yA, WORD wid, WORD slen, TEXT str
02	PP_CHAR WORD y, WORD x, WORD yD, WORD yA, WORD wid, BYTE char
03	PP_HLINE WORD y, WORD x, WORD yD, WORD yA, WORD wid
04	PP_VLINE WORD y, WORD x, WORD yD, WORD yA, WORD wid
05	PP_ARRAY PPEXPR, …
06	PP_SQRT WORD y, WORD x, WORD yD, WORD yA, WORD wid
07	PP_INTEGRATE WORD y, WORD x, WORD yD, WORD yA, WORD wid
08	PP_OPEN_PAREN WORD y, WORD x, WORD yD, WORD yA, WORD wid
09	PP_CLOSE_PAREN WORD y, WORD x, WORD yD, WORD yA, WORD wid
0A	PP_SIGMA WORD y, WORD x, WORD yD, WORD yA, WORD wid

0B	PP_PI_PRODUCT WORD y, WORD x, WORD yD, WORD yA, WORD wid
0C	PP_OPEN_MAT WORD y, WORD x, WORD yD, WORD yA, WORD wid
0D	PP_CLOSE_MAT WORD y, WORD x, WORD yD, WORD yA, WORD wid
0E	PP_OPEN_LIST WORD y, WORD x, WORD yD, WORD yA, WORD wid
0F	PP_CLOSE_LIST WORD y, WORD x, WORD yD, WORD yA, WORD wid

```
enum PP_TYPE {
  PP_LTEXT=1,     /*literal text, length word */
  PP_CHAR=2,
  PP_HLINE=3,
  PP_VLINE=4,
  PP_ARRAY=5,     /* Array of items, ending in END_TAG=0xE5 */
  PP_SQRT=6,
  PP_INTEGRATE=7,
  PP_OPEN_PAREN=8,
  PP_CLOSE_PAREN=9,
  PP_SIGMA=10,
  PP_PI_PRODUCT=11,
  PP_OPEN_MAT=12,
  PP_CLOSE_MAT=13,
  PP_OPEN_LIST=14,
  PP_CLOSE_LIST=15
};

typedef byte PP_TYPE;

//2D_Flags
enum { PF_NO_DIV2RAT=0x8, PF_FULLPREC=1 };
```

```
static const BYTE scDMS[3] = {0xB0, 0x27, 0x22}; //degrees, minutes, seconds
static const char sLE[2] = {0x9C};
static const char sGE[2] = {0x9E};
static const char sNE[2] = {0x9D};
static const char sDot[2] = {0xB7};
static const char sPN[2] = {0xB1};
static const char sTo[2] = {0x12};

static const char * const OP_names[23]= {
  "|", "xor", "or", "and", "<", sLE, "=", sGE, ">", sNE,
  "+", ".+", "-", ".-", sDot, ".*", "/", "./", "^", ".^", "&", sPN, sTo
};

static const BYTE OP_widths[23] =
  { 4, 6, 6, 6, 3, 3, 3, 3, 3, 3, 3, 6, 3, 6, 0, 6, 3, 6, 3, 6, 3, 3, 3 };

static const BYTE OP_flags[23] = { //not used:
  0x80, 0x82, 0x82, 0x83, 0x8A, 0x8A, 0x8A, 0x8A, 0x8A, 0x8A, 0x8A, 0x8A,
  0x8A, 0x8A, 0x8D, 0x8D, 0x8D, 0x8D, 0x8D, 0x8D, 0x83, 0x8A, 0x81
};
```

```
void PP_Init( void );
```

```
EStackIndex Parse2DExpr(EStackIndex esi, BOOL torf);
#define Parse2DExpr ASAP_ref(74, EStackIndex (*const)(EStackIndex, BOOL))
/* Parse expression at esi, so it can be displayed with Print2DEexpr
(this just splits up expression into blocks with rel. position and size info)
Note: this is split up for efficiency, so can display again without having to calculate
the positions again */

void Print2DExpr(EStackIndex esi, WINDOW *w, SWORD x, SWORD y);
#define Print2DExpr ASAP_ref(76, void (*const)(EStackIndex, WINDOW *, SWORD, SWORD))
/* Print out Parametrised expression */

static EStackIndex Print2DExpr_aux(EStackIndex esi, SWORD x, WIN_COORDS y);
/* recursive */
EStackIndex Parse2DMultiExpr(HANDLE h, BOOL torf);
#define Parse2DMultiExpr ASAP_ref(75, EStackIndex (*const)(HANDLE, BOOL))
/* Parse multi statement expression in h so that it can be displayed with Print2DExpr
*/
```

```
void Parms2D(EStackIndex esi, SWORD* Width, SWORD* yB, SWORD* yA);
#define Parms2D ASAP_ref(77, void (*const)(EStackIndex, SWORD*, SWORD*, SWORD*))
/* Get pretty print paramaters as described above
*Width = total width of block, *yA = y above position, *yB = y below position */
```

```
static void push_no_mem( void );
static void get_matrix_dim(EStackIndex esi, SWORD *rows, SWORD* cols);
static PP_TYPE push_matrix(EStackIndex esi);
static BOOL is_list_op(BYTE tag);
static BOOL pushed_rat(EStackIndex esi, WORD tag);
static PP_TYPE push_number(EStackIndex esi, BYTE Radix);
static EStackIndex push_params
  (EStackIndex esi, SWORD *Width, SWORD *yB, SWORD *yA, WORD Params);
static EstackIndex push_var
  (EStackIndex esi, SWORD *Width, SWORD *ayB, SWORD *ayA, SWORD* Height);
static void push_info
  (SWORD width, SWORD yB, SWORD yA, SWORD x, SWORD y, BYTE type);
static void push_char_info(char c, SWORD x, SWORD y);
static SWORD push_name(const char *s, SWORD x, BOOL push_trail); //trail pos,str
static PP_TYPE push_arb(EStackIndex esi, BOOL int); //int or real

static PP_TYPE ParseRangeInParen(EStackIndex esi, WORD low, WORD high);
static PP_TYPE ParseLowerPrecedence(BOOL lower, EStackIndex esi);
static PP_TYPE ParseHigherPrecedence(BOOL higher, EStackIndex esi);
static PP_TYPE ParseTokenInParen(EStackIndex esi);
static PP_TYPE ParseToken(EStackIndex esi);

static void push_width_throw(SWORD x);
static void push0_word( void );
static void push_word(SWORD w);
static void putns(short* p, SWORD v); //put non-aligned short (word)
static SWORD getns(const short* p); //get non-aligned short
static void CheckMemoryThrow(EStackIndex esi); //check esi isn't too high
static EStackIndex push_integer(EStackIndex esi, BOOL bSigned, BYTE Radix);
static const char *get_key_ptr(EStackIndex esi); //Get Name (detokenise)
static EStackIndex GetNext(EStackIndex esi); //next info block
```

```
static WORD tw;
static SWORD yA,yB;
static SWORD tsw;
static SWORD y,x;
static WINDOW* wDest;
static BYTE tb, ExprFont;
/* These vars are mostly temporary, and not particular specific */
```

```
BOOL PP_bString;
WORD PP_Flags;
/* These are also used in 1D Expressions (see Text Conversion (dispexpr.o)) */
```

```
//This is all done via the estack, Words are in little endian
struct PrintStruct {
  packed union {
    struct {
      TEXT Text;
      NA_WORD Len;
    } String;
    char c;
    void Nothing;
  };
  WORD Width,
  NA_WORD yAscent;
  NA_WORD yDescent;
  NA_WORD x;
  NA_WORD y;
  BYTE PP_TYPE;
} PP_PARMS;
```

Text Conversion (dispexpr.o)

Converts RPN expression to TEXT. Called from display_statements called from NG_RPNToText. Parse1Dexpr is more generally used.

```
//print: output to end of buffer, reallocating block as required
static void print_char(char c);
static void print_str(EStackIndex esi);
static void print_str_spaced(EStackIndex esi);
static void print_integer(EStackIndex esi, BYTE base);
static void print_number(EStackIndex esi, BYTE base);
static EStackIndex print_expr_paren(EStackIndex esi);
static EStackIndex print_LowerPrecedence(BOOL lower, EStackIndex esi);
static EStackIndex print_HigherPrecedence(BOOL higher, EStackIndex esi);
static EStackIndex print_RangeInParen(EStackIndex esi, BYTE low, BYTE high);
static EStackIndex print_params(EStackIndex esi);

const char *get_key_ptr(BYTE tag1, BYTE tag2);
#define get_key_ptr ASAP_ref(695, const char (*const)(BYTE, BYTE))
/* gets name from tags, tag2 is only used for extended tags etc. */

EStackIndex index_below_display_expression_aux(EStackIndex esi);
#define index_below_display_expression_aux ASAP_ref(694, EStackIndex
(*const)(EStackIndex))
/* main (recursive) routine for detokenizing (shouldn't need to use) */

HANDLE display_statements(EStackIndex esi, WORD width, BOOL torf);
#define display_statements ASAP_ref(78, HANDLE (*const)(EStackIndex, WORD, BOOL))
/* Detokenizes expression at esi, return handle of TEXT
torf specifies floating point output precision XR_FloatFullPrec if true
*/

HANDLE Parse1DExpr(EStackIndex esi, BOOL torf, WORD width);
#define Parse1DExpr ASAP_ref(79, HANDLE (*const)(EStackIndex, BOOL, WORD))
/* Does the same as display_statements (paramater order differed)
returns handle to text */

static WORD Flags;
static WORD BufLen;
static HANDLE hText;
static char* CurPos;
static BOOL torf;
static DWORD Width;
```

System Events and Dialogs

TaskID:

>0	app num
-1	Running app
-2	current app
-3	override CM_PAINT block

Apps can register menus with the system event handler, these have lists of options along with ev command codes which the event handler in the app should respond to, F? keys are checked if pressed menu is processed and found EV_code send to handler.

```
event in range:
      <  0x500    XR_String is sent as message
      >= 0x500    custom commands (dialogs etc.)
      >= 0x700    EV commands
```

System events automatically have correct keypresses sent to them (diam-?) others have to be responded to in EV handling code. cos sin tan etc. are all sent via sendstring ($723) and do not have to be processed as keypresses. off etc all handled by the event handler.

System Messages

Stored in (event.Type)

Code	Extra	Description	Default
700		Idle (idle() called after message sent) [e.g. spinning of 3D graph]	
701		Init (app) [called from EV_centralDispatcher after reset]	
702	r	Start Task (in rect r) [event->StartType check here]	
703		Activate [set command state] [update register menus]	startMenu()
704		Focus [get Mode options, startApp]	
705		UnFocus (switch)	
706		DeActivate [Hide menus]	endMenu()
707		End Task (WinHide, Cleanup) ()	
708		CM_START_CURRENT start current	
70B		Reset DEFAULTS (from MEM Dialog)	
710	key	Keycode	Handle system keys
720		Menu: Cut	
721		Menu: Copy	
722		Menu: Paste	
723	PasteText	Paste static (constant) text (pointer)	EV_pasteString = event -> extra.pasteText;
724	hPasteText	Paste dynamic text (HANDLE)	HeapFree(event -> extra.hPasteText);
725		DEL (backspace)	
726		CLR (button)	
727		Menu: Clear	
728		Menu: Find	
730		Insert	
740		Flash cursor (.5 sec timer)	
750		Store (keypress)	handleStoreKey()
751		recall var	handleRclKey(TRUE);
760	w	CM_WPAINT (paint window) *w to paint (apps should check to see if *w belongs to the app and redraw it if it is). See EV_paintWindows. These are not sent to the capturing hook, unless app == -3 (force redraw) WinOpen stores the current app in w->TaskId	
770		Menu: Open…	

771		Menu: Save Copy As…	
772		Menu: New…	
773		Menu: Format…	
774		Menu: About…	
780		Notify Mode Change (see MO_notifyModeChange)	
781		Notify Switch Graph (sent to AP_GRAPH, AP_WINDOW, AP_TABLE, AP_EQU)	
7C0		Geom open Data	

Event Mechanism (event.o)

```
#ifdef AMS2
typedef HANDLE TASK;
#else

typedef WORD TASK;

enum AppTaskIDs {
  AP_ALGEBRA = 0,
  AP_EQU = 1,
  AP_WINDOW = 2,
  AP_GRAPH = 3,
  AP_TABLE = 4,
  AP_DATA = 5,
  AP_PROGRAM = 6,
  AP_GEOM = 7,
  AP_TEXT = 8,
  AP_SOLVER = 9,
  AP_SLFTST = 10, //<F5>+<DIAMOND>+<(>+<S>
};

#endif

Enum StandardTaskIDs {
  AP_NULL = -3,
  AP_RUNNING = -2,
  AP_CURRENT = -1,
};

enum StartType {
  AP_START_CURRENT = 0,
  AP_START_NEW = 0x10,
  AP_START_OPEN = 0x20,
  AP_START_ERROR = 0x30,
};

typedef struct EventStruct {
  WORD Type;
  WORD RunningApp;
  WORD Side;
  WORD StatusFlags;

  union {
    WINDOW *w;
    WIN_RECT *r;
    char* pasteText;
    HANDLE hPasteText;
    struct {
      WORD Mod;
      KEYCODE Code;
    } Key;
  } extra;
  BYTE StartType;
} EVENT;
```

```
void (* static const appTable[11])(EVENT*) = {
  AP_algebra,
  AP_equ,
  AP_window,
  AP_graph,
  AP_table,
  AP_data,
  AP_program,
  AP_geom,
  AP_text,
  AP_solver
  AP_slftst,
};
```

```
void (*old)(EVENT*) EV_captureEvents(void (*handler)(EVENT*));
#define EV_captureEvents ASAP_ref(198, void (*)(EVENT*) (*const)(void (*)(EVENT*)))
/* sets EV_capture hook, overrides any current message handler, used in custom windows
About, Catalog, Error Message, etc. See Error Dialogs (erdialog.o)
*/
```

```
void EV_centralDispatcher( void );
#define EV_centralDispatcher ASAP_ref(342, void (*const)( void ))
/* initialises and despatches and continually calls EV_eventLoop, under an Error Handler
(TRY).
As the name suggests this is the main controlling loop of the entires OS.
if an error is caught it is despatched to ERD_Process
ERD_Process calls ERD_Dialog under an error handler, and switches applicationas etc.
Sends $701 to all apps, startTask(AP_ALGEBRA)
$706, $703
*/
```

```
void EV_clearPasteString( void );
#define EV_clearPasteString ASAP_ref(199, void (*const)( void ))
/* Clears EV paste buffer, used primarily for cos etc.
Frees hPasteBuffer, and clears pointer pasteString
```

```
void EV_defaultHandler(EVENT *);
#define EV_defaultHandler ASAP_ref(343, void (*const)(EVENT *))
/* as expected called to process any unhandled messages in handlers
Note: sending a string to the default handler will set a static pointer to that data. The
event loop will then send the string as individual keypresses to that app.
For a buffer is exported from this code which allows pasting large amounts of text.
*/
```

```
void EV_eventLoop( void );
#define EV_eventLoop ASAP_ref(344, void (*const)( void ))
/* checks timers, ports etc, and despatches main messages via sendevent
it also checks for errors.
EV_ErrorCode can be set to any error code which calls ERD_Dialog.
There is no error handler in place, so ER_Throw passes control the the previous level or
handler. In terms of normal operation this is handled by EV_centralDespatcher which
restarts the loop…However in custom event handling is is used to return to the calling
code.
*/
```

```
WORD EV_getc(WORD busy, EVENT *event);
#define EV_getc ASAP_ref(200, WORD (*const)(WORD, EVENT *))
/* wait for keypress, fill event structure
busy is a ST_ACTIVE enumeration
returns the keycode of keypress
if there is no keypress and the cursor timer expires, this event is sent instead
*/
```

```
const WIN_RECT EV_splitRects[14];
/* [MO_splitScreen][Side]
*/
```

```
WIN_RECT *EV_getSplitRect(WORD Side);
#define EV_getSplitRect ASAP_ref(201, WIN_RECT *(*const)(WORD))
/* get screen rect for side
*/

void EV_notifySwitchGraph( void );
#define EV_notifySwitchGraph ASAP_ref(202, void (*const)( void ))
/* notify appropriate applications of graph switch
sends CN_SWITCH_GRAPH to graph, window, table, yeq
*/

BOOL EV_paintOneWindow( void );
#define EV_paintOneWindow ASAP_ref(203, BOOL (*const)( void ))
/* repaints topmost window
draws last WF_DIRTY flag set (clears flag). CM_WPAINT is sent to any valid task
returns TRUE if window was painted, FALSE if none to paint
*/

void EV_paintWindows( void );
#define EV_paintWindows ASAP_ref(204, void (*const)( void ))
/* paints all windows
if EV_paint&1==0
*/

void EV_registerMenu(HANDLE hMenuState);
#define EV_registerMenu ASAP_ref(345, void (*const)(HANDLE))
/* attaches a Menu State to an application (in the APP_MENUS)
*/

BOOL EV_restorePainting(BOOL blockPaint);
#define EV_restorePainting ASAP_ref(205, BOOL (*const)(BOOL))
/* sets paint flags 0x1
returns previous value
*/

void EV_sendEvent(TASK TaskID, EVENT *);
#define EV_sendEvent ASAP_ref(206, void (*const)(TASK, EVENT *))
/* SendEvent to TaskID from current side
*/

void EV_sendEventSide(TASK TaskID, EVENT *, WORD Side);
#define EV_sendEventSide ASAP_ref(207, void (*const)(TASK, EVENT *, WORD))
/* send event to TaskID from Side
*/

void EV_sendString(WORD XR_String);
#define EV_sendString ASAP_ref(208, void (*const)(WORD))
/* send xr_string to running app
*/

//Menu State Commands

void EV_setCmdCheck(WORD cmd, BOOL check);
#define EV_setCmdCheck ASAP_ref(209, void (*const)(WORD, BOOL))
/* check/uncheck menu cmd in current app
*/

void EV_setCmdState(WORD cmd, BOOL state);
#define EV_setCmdState ASAP_ref(210, void (*const)(WORD, BOOL))
/* enable/disable menu cmd in current app
*/

void EV_setFKeyState(WORD num, BOOL state, BOOL redraw);
#define EV_setFKeyState ASAP_ref(211, void (*const)(WORD, BOOL, BOOL))
/* sets (and optionally redraws) the state of menu num
menu num is 0 indexed
optionally redraws the menus as well
*/
```

```
void EV_startApp(TASK TaskID, WORD StartType);
#define EV_startApp ASAP_ref(212, void (*const)(TASK, WORD))
/* start app from any state
*/

void EV_startSide(TASK *saveTaskID, TASK TaskID, WORD Side);
#define EV_startSide ASAP_ref(213, void (*const)(TASK *, TASK, WORD))
/* start side, *=TaskId
Sends: $702, $703, $704 to current side
*/

void EV_startTask(WORD StartType);
#define EV_startTask ASAP_ref(214, void (*const)(WORD))
/* sends CM_STARTTASK to running app
*/

void EV_suspendPainting( void );
#define EV_suspendPainting ASAP_ref(215, void (*const)( void ))
/* sets EV_paint&1, stops window painting
*/

void EV_switch( void );
#define EV_switch ASAP_ref(216, void (*const)( void ))
/* perform switch side
*/

static void startMenu( void );
/* registers and starts the menu for running app
*/

static void endMenu( void );
/* ends custom and normal menu
*/

static void handleStoreKey( void );
static void CharDlg( void );
static void handleMenuKey( void );
static void MathDlg( void );
static void AppsMenu( void );
static void OffStartHome( void );
static void handleQuitKey( void );

void push_switch(EStackIndex esi);
#define push_switch ASAP_ref(807, void (*const)(EStackIndex))

static void sendCurMessage(WORD eventType);

BOOL QstandardState( void );
/* { return ( EV_currentApp==0 && MO_splitScreen==0 && EV_capture ); }
```

Using EV_hook you can capture all messages and process them as you like, as an example you can hook the APPS key and display a different menu (see my web page)...You can then start your own event driven application via EV_captureEvents.

```
struct AppMenuStruct {
  MENU* Menu;
  HANDLE hMenuState;
} APP_MENUS[11]; //one spare slot

void (*EV_capture)(EVENT*);
/* Pointer to capture hook
See EV_captureEvents
*/

void (*EV_hook)(EVENT*);
#define EV_hook ASAP_ref(675, void (* (*const))(EVENT*))
/* hooks events like EV_captureEvents, however, the application handler is still called
afterwards this is probably meant for debugging purposes as it is never used
```

```
*/

WORD EV_runningApp;
WORD EV_currentApp;
WORD EV_appA;
WORD EV_appB;
WORD EV_side;

ER_CODE EV_errorCode;
/* System Error Code indicator
EV_eventLoop will process this error, calling ERD_process
*/

WORD EV_paint;

char* EV_pasteString;
/* Pointer to current paste character
EV_defaultHandler will send any unprocessed CM_STRING messages a keypress at a time, this
pointer is used to step though the string
*/

HANDLE EV_hPasteBuffer;
/* Buffer for pasting text via CM_STRING
Not used internally, handle is freed by EV_clearPasteString, so this buffer may be used
as a buffer for pasting text.
*/
```

About Dialog (about.o)

```
void ABT_dialog( void );
#define ABT_dialog ASAP_ref(269, void (*const)( void ))

static void AB_handleEvent(EVENT* ev);
static void AB_close( void );

static void AB_prodinfo( void ):
/* Draws all the information lines in about dialog from following routines: */

void AB_prodid( char *dest );
#define AB_prodid ASAP_ref(669, void (*const)( void ))
/* Writes in product ID as described above */

void AB_prodname( char *dest );
#define AB_prodname ASAP_ref(670, void (*const)( void ))
/* Writes producte name AMS to dest */

void AB_serno( char *dest );
#define AB_serno ASAP_ref(671, void (*const)( void ))
/* Build serial number form("#%s %04X", insert_space_in_centre(cgetsn), FL_getVerNum) */

static WORD old_paint;
static WORD old_capture;
static WINDOW wAbout;
```

Catalog Dialog (cat.o)

```
type struct {      /* All XRefs */
  WORD Name;       /* Display Name */
  WORD PasteName; /* Pasted version (same, or may contain extra spaces */
  WORD Parms;      /* Parm info */
} CAT_INFO;

static const WIN_RECT ScrollRect;
```

```
void CAT_dialog( void );
#define CAT_dialog ASAP_ref(293, void (*const)( void ))

static void CAT_handleEvent(EVENT* ev);
static void close( void );
static void copy( void );
static void showParms( void );
static void redrawline( void );
static void scrollDown( void );
static void pageDown( void );
static void toStart( void );
static void toEnd( void );
static void scrollUp( void );
static void pageUp( void );
static void jumpRedraw( void );

static const Entries[0x157]; /* size may vary */
static const NumEntries = 0x157;
static const Index[26];

static WORD old_paint;
static WORD line, old_line;
static WORD old_capture;
static WINDOW wCatalog;
```

Misc. Simple Fundamentals

- Status Line Control
- Clipboard
- Cursor
- Character Code Table
- String Table (xrefs)

Status Line Control (status.o)

```
enum ST_ANGLE {RAD=0, DEG=1};
enum ST_BATT{ OK=0, LOW=1, REPLACE=2 };
enum ST_ACTIVITY {IDLE=0, BUSY=1, PAUSE=2, NORMAL=3};
enum ST_GRAPH {FUNC=0, PAR=2, POL=3, SEQ=4, 3D=5, DE=6};
enum ST_MODKEY {NONE=0, 2ND=1, SHIFT=2, DIAMOND=4, LOCK=8};
enum ST_ALPHA {SH_A_LOCK=1, A_LOCK=2};
enum ST_PREC { AUTO=0, EXACT=1, APPROX=2 };

typedef struct StatusFlagStruct {
  BOOL ReadOnly : 1; //0x80000
  ST_BATT Batt : 2; //0x60000
  BOOL ModUpdate : 1; //0x10000
  BOOL Erase : 1; //0x8000
  ST_BUSY Busy : 2; //0x6000
  ST_GRAPH Graph : 4; //0x1E00
  ST_PREC Precision : 2; //0x180
  ST_ANGLE Angle : 1; //0x40
  ST_ALPHA Lock : 2; //0x30 (lock?alpha) (0x38 alpha or hand)
  ST_MODKEY Modifiers : 4; //0xF (0x7)
} ST_FLAGS;
```

```
void ST_angle (WORD mode);
#define ST_angle ASAP_ref(224, void (*const)(WORD))
void ST_batt (WORD mode);
#define ST_batt ASAP_ref(225, void (*const)(WORD))
void ST_busy (WORD mode);
#define ST_busy ASAP_ref(226, void (*const)(WORD))
BOOL ST_eraseHelp (void);
#define ST_eraseHelp ASAP_ref(227, BOOL (*const)(void))
```

```
void ST_folder (char *name);
#define ST_folder ASAP_ref(228, void (*const)(char *))
void ST_graph (WORD mode);
#define ST_graph ASAP_ref(229, void (*const)(WORD))
void ST_helpMsg (char *msg);
#define ST_helpMsg ASAP_ref(230, void (*const)(char *))
void ST_Init( void );

void ST_modKey (WORD flags);
#define ST_modKey ASAP_ref(231, void (*const)(WORD))
void ST_precision (WORD mode);
#define ST_precision ASAP_ref(232, void (*const)(WORD))

void ST_readOnly (BOOL lock);
#define ST_readOnly ASAP_ref(233, void (*const)(BOOL))
/* Update ST_Flags for displaying the lock symbol
Note this appears when you open a locked variable in an editor.
Set from TE_indicateReadOnly
*/

void ST_stack (WORD index, WORD total);
#define ST_stack ASAP_ref(234, void (*const)(WORD, WORD))
/* Internalises information for home screen stack numbers
displaying history pair info, "i/t"
*/

void ST_refDsp (WORD msg_no);
#define ST_refDsp ASAP_ref(235, void (*const)(WORD))
/* Looks up msg_no in XR_TABLE, number is automatically mapped to ST range.
```

1	"TYPE OR USE \0x15\0x16\0x17\0x18 + [ENTER]=OK AND [ESC]=CANCEL"
2	"USE \0x15 AND \0x16 TO OPEN CHOICES"
3	"USE \0x15\0x16\0x17\0x18 + [ENTER]=OK AND [ESC]=CANCEL"
4	"TYPE + [ENTER]=OK AND [ESC]=CANCEL"
5	"USE \0x15\0x16\0x17\0x18 OR TYPE + [ESC]=CANCEL"
6	"USE \0x15\0x16\0x17\0x18 + [ENTER]=OK AND [ESC]=CANCEL, OR DRAG"
7	"DATA PLACED IN VARIABLE SYSDATA"
8	"DATA PLACED IN HOME SCREEN HISTORY"
9	"[ENTER]=OK AND [ESC]=CANCEL"
10	""
11	"USE \0x15\0x16 + [ENTER]=OK AND [ESC]=CANCEL"
12	"USE \0x17\0x18 + [ENTER]=OK AND [ESC]=CANCEL"
13	"USE [2ND] [KEYS] OR [ESC]=CANCEL"

```
*/

static void drawAngle( void );
static void drawBatt( void );
static void drawBusy( void );
static void drawLock( void );
static void drawFolder( void );
static void drawGraph( void );
static void drawModKey( void );
static void drawPrecision( void );
static void drawStack( void );
static void ST_Clear( void );
static void ST_Fill(const WIN_RECT *r, BYTE Attr);
static void ST_StrX(WIN_COORDS x, char* str);

static WORD HistoryTotal;
static WORD HisoryCurrent;
static char FolderText0[9];
static WINDOW wStatus;

ST_FLAGS ST_flags;
```

Clipboard (clipboard.o)

TIOS only uses clipboard for text, but capable of other types (ignored).

```
void CB_Init( void );

BOOL CB_replaceTEXT(TEXT text, size_t len, BOOL strip_CR);
#define CB_replaceTEXT ASAP_ref(193, BOOL (*const)(TEXT, size_t, BOOL))
TEXT CB_fetchTEXT(HANDLE *hText, size_t *len);
#define CB_fetchTEXT ASAP_ref(194, TEXT (*const)(HANDLE *, size_t *))

static char DataType[4]; //'TEXT'
static size_t DataLen;
static HANDLE hClipboard;
```

Cursor (cursor.o)

A very small object which maintains state of the Cursor and its timer.

```
void CU_init( void );

void CU_restore(BOOL state);
#define CU_restore ASAP_ref(195, void (*const)(BOOL))
BOOL CU_start( void );
#define CU_start ASAP_ref(196, BOOL (*const)( void ))
void CU_stop( void );
#define CU_stop ASAP_ref(197, void (*const)( void ))

BOOL CU_active;

TIMER ID4
```

Character Code Table (chartypes.o)

The character table returned by id 8 is frequently used in the ROM in mapping characters, e.g. mapping upper case to lower case characters etc. This justifies its appearance here.

```
enum CharTypes {
  Symbol=0x40
  Number=0x4C,
  Capital=0x5A, //including accents
  Minus=0x48,
  Lower=0x59, //including accents
  Greek=0x78,
  Pi=0x60,
};

const BYTE CharTbl[256];
```

String Table (xrefs) (XRef.o)

This is where many of the commonly used strings are stored. Either referred to directly, or via a lookup table, XR_stringPtr.

It is not recommended to make use of these id's as they are not consistent across ROM versions, and are mainly used interally for efficiency. For example dialog structures and EV_sendString use them.

Note on AMS2 practically all text references are done though this module (Requiring a DWORD jump table of offsets rather than a WORD jump table).

```
const char *XR_stringPtr(WORD id);
#define XR_stringPtr ASAP_ref(659, const char *(*const)(WORD))
```

```
Items commonly referred to directly in table:
const char *XRA_NoMemory = {0xAB, 0xA0, 0xBB, 0};
/* Note enough memory to display line: <<...>> */
const char *XRA_FloatFullPrec = "%^.14G";
/* TI format, maximum of 14 significant digits, compact form */
const char *XRA_XB_FloatFullPrec = "%^z.14G";
/* TI format, no trailing chars, max 14 sf, compact form - used in dialogs */
```

Drawing and Windows

- Graphics Library
- Windows

Graphics Library (glib.o)

```
enum Attrs { A_REVERSE=0, A_NORMAL=1, A_XOR, A_SHADED, A_REPLACE, A_OR,
           A_AND, A_THICK1, A_SHADE_V, A_SHADE_H, A_SHADE_NS, A_SHADE_PS };
enum BoxAttrs { B_NORMAL=0x10, B_ROUNDED=0x20, B_DOUBLE=0x40, B_CUT=0x80 };
enum Fonts { F_4x6, F_6x8, F_8x10 };

#define ValidateRect(r) assert(r.xy.x0 <= MAX_X && r.xy.x1 <= MAX_X &&  \
r.xy.y0 <= MAX_Y && r.xy.y1 <= MAX_Y && r.xy.x0 <= r.xy.x1 && r.xy.y0 <= r.xy.y1 )
#define ValidateRectP(r) assert(r->xy.x0 <= MAX_X && r->xy.x1 <= MAX_X &&  \
r->xy.y0 <= MAX_Y && r->xy.y1 <= MAX_Y && r->xy.x0 <= r->xy.x1 && r->xy.y0 <= r->xy.y1 )
#define ValidateX(x) assert(x >= 0 && x <= MAX_X)
#define ValidateY(y) assert(y >= 0 && y <= MAX_Y)
#define ValidateFont(f) assert(f <= F_8x10)
typedef unsigned char SCR_COORDS;
#define BITMAP_HDR_SIZE   4

typedef union {
struct {
  SCR_COORDS x0, y0;
  SCR_COORDS x1, y1;
} xy;
unsigned long l;
} SCR_RECT;

#if defined(_92)
#define MAX_X 239
#define MAX_Y 127

#elif defined(_89)
#define MAX_X 159
#define MAX_Y 99
#endif

typedef signed short WIN_COORDS;

typedef struct {unsigned short i[16];} ICON;
typedef unsigned short *pICON;

typedef struct {
  WORD NumRows;
  WORD NumCols;
  BYTE Data[1];   /* should be [] */
} BITMAP;

typedef struct {
  BYTE Attr;
  SCR_COORDS x0, y0, x1, y1;
} MULTI_ENTRY;

typedef struct {
  BYTE NumLines;
  MULTI_ENTRY Data[1];  /* should be [] */
} MULTI_LINES;
```

```
typedef struct {
  WIN_COORDS x0, y0;
  WIN_COORDS x1, y1;
} WIN_RECT;

typedef struct {
  WIN_COORDS x0, y0;
} WIN_POINT;

struct ScrStateStruct {
  void* Port;
  WORD xMax, yMax;
  BYTE Font;
  WORD Attr;
  WORD x, y;
  SCR_RECT Clip;
} SCR_STATE;
```

```
static const BYTE ClipMasks[0x3C];
```

```
void DrawClipChar(WORD x, WORD y, char c, const SCR_RECT *Clip, WORD Attr);
#define DrawClipChar ASAP_ref(401, void (*const)(WORD, WORD, char, const SCR_RECT *,
WORD))
void DrawClipLine(const WIN_RECT *Line, const SCR_RECT *Clip, WORD Attr);
#define DrawClipLine ASAP_ref(403, void (*const)(const WIN_RECT *, const SCR_RECT *,
WORD))

void DrawClipPix(WORD x, WORD y);
#define DrawClipPix ASAP_ref(404, void (*const)(WORD, WORD))
void DrawClipEllipse
  (WORD x, WORD y, WORD a, WORD b, const SCR_RECT *Clip, WORD Attr);
#define DrawClipEllipse ASAP_ref(402, void (*const)(WORD, WORD, WORD, WORD, const
SCR_RECT *, WORD))

void DrawClipRect(const WIN_RECT *r, const SCR_RECT *Clip, WORD Attr);
#define DrawClipRect ASAP_ref(405, void (*const)(const WIN_RECT *, const SCR_RECT *,
WORD))
void DrawMultiLines(WORD x, WORD y, const MULTI_LINES *Multi);
#define DrawMultiLines ASAP_ref(406, void (*const)(WORD, WORD, const MULTI_LINES *))

int DrawStrWidth(const char*, BYTE Font);
#define DrawStrWidth ASAP_ref(407, int (*const)(const char*, BYTE))

BYTE FontSetSys(BYTE Font);
#define FontSetSys ASAP_ref(399, BYTE (*const)(BYTE))
BYTE FontGetSys(void);
#define FontGetSys ASAP_ref(398, BYTE (*const)(void))
WORD FontCharWidth(char c);
#define FontCharWidth ASAP_ref(400, WORD (*const)(char))
int GetPix(WORD x, WORD y);
#define GetPix ASAP_ref(415, int (*const)(WORD, WORD))
void LineTo(WORD x, WORD y);
#define LineTo ASAP_ref(412, void (*const)(WORD, WORD))
void MoveTo(WORD x, WORD y);
#define MoveTo ASAP_ref(413, void (*const)(WORD, WORD))
void PortSet(void *buf, SCR_COORDS Wid, SCR_COORDS Hgt);
#define PortSet ASAP_ref(418, void (*const)(void *, SCR_COORDS, SCR_COORDS))
void PortRestore( void );
#define PortRestore ASAP_ref(419, void (*const)( void ))
void RestoreScrState(const SCR_STATE *scr);
#define RestoreScrState ASAP_ref(417, void (*const)(const SCR_STATE *))
void SaveScrState(SCR_STATE *scr);
#define SaveScrState ASAP_ref(416, void (*const)(SCR_STATE *))

static SCR_RECT *ClipRect(SCR_RECT *r, const SCR_RECT *Clip);
void ScrRectFill(const SCR_RECT *r, const SCR_RECT *Clip, WORD Attr);
```

```
#define ScrRectFill ASAP_ref(393, void (*const)(const SCR_RECT *, const SCR_RECT *,
WORD))

void BitmapGet(const SCR_RECT *r, BITMAP *Bitmap);
#define BitmapGet ASAP_ref(389, void (*const)(const SCR_RECT *, BITMAP *))
void BitmapInit(const SCR_RECT *r, BITMAP *Bitmap);
#define BitmapInit ASAP_ref(390, void (*const)(const SCR_RECT *, BITMAP *))
int ScrRectOverlap(const SCR_RECT *r1, const SCR_RECT *r2, SCR_RECT *dest);
#define ScrRectOverlap ASAP_ref(394, int (*const)(const SCR_RECT *, const SCR_RECT *,
SCR_RECT *))
void BitmapPut(WORD x, WORD y, BITMAP *Bitmap, const SCR_RECT *Clip, WORD Attr);
#define BitmapPut ASAP_ref(391, void (*const)(WORD, WORD, BITMAP *, const SCR_RECT *,
WORD))
void ScrRectScroll
  (const SCR_RECT *r, const SCR_RECT *Clip, SWORD NumRows, WORD Attr);
#define ScrRectScroll ASAP_ref(395, void (*const)(const SCR_RECT *, const SCR_RECT *,
SWORD, WORD))
void ScrRectShift
  (const SCR_RECT *r, const SCR_RECT *Clip, SWORD NumCols, WORD Attr);
#define ScrRectShift ASAP_ref(396, void (*const)(const SCR_RECT *, const SCR_RECT *,
SWORD, WORD))
WORD BitmapSize(const SCR_RECT *r);
#define BitmapSize ASAP_ref(392, WORD (*const)(const SCR_RECT *))
int QScrRectOverlap(const SCR_RECT *r1, const SCR_RECT *r2);
#define QScrRectOverlap ASAP_ref(397, int (*const)(const SCR_RECT *, const SCR_RECT *))
void ScreenClear( void );
#define ScreenClear ASAP_ref(414, void (*const)( void ))
SCR_RECT DefScrRect;

void GL_Init( void );

int SetCurAttr(WORD Attr);
#define SetCurAttr ASAP_ref(410, int (*const)(WORD))
void SetCurClip(const SCR_RECT *Clip);
#define SetCurClip ASAP_ref(411, void (*const)(const SCR_RECT *))
void DrawChar(WORD x, WORD y, char c, WORD Attr);
#define DrawChar ASAP_ref(420, void (*const)(WORD, WORD, char, WORD))
void DrawFkey(WORD x, WORD y, WORD num, WORD Attr);
#define DrawFkey ASAP_ref(421, void (*const)(WORD, WORD, WORD, WORD))
void DrawIcon(WORD x, WORD y, const pICON Icon, WORD Attr);
#define DrawIcon ASAP_ref(422, void (*const)(WORD, WORD, const pICON, WORD))
void DrawLine(WORD x0, WORD y0, WORD x1, WORD y1, WORD Attr);
#define DrawLine ASAP_ref(423, void (*const)(WORD, WORD, WORD, WORD, WORD))
void DrawPix(WORD x, WORD y, WORD Attr);
#define DrawPix ASAP_ref(424, void (*const)(WORD, WORD, WORD))
void DrawStr(WORD x, WORD y, const char *Str, WORD Attr);
#define DrawStr ASAP_ref(425, void (*const)(WORD, WORD, const char *, WORD))
void FillTriangle
  (WORD x0, WORD y0, WORD x1, WORD y1, WORD x2, WORD y2, const SCR_RECT *clip, WORD
Attr);
#define FillTriangle ASAP_ref(408, void (*const)(WORD, WORD, WORD, WORD, WORD, WORD,
const SCR_RECT *, WORD))
void FillLines2
  (const WIN_RECT *low_line, const WIN_RECT *hi_line, const SCR_RECT *Clip, WORD Attr);
#define FillLines2 ASAP_ref(409, void (*const)(const WIN_RECT *, const WIN_RECT *, const
SCR_RECT *, WORD))

static BYTE *GetPtrPixel(WORD x, WORD y);
static void DrawEllipsePix
  (WORD dx, WORD dy, WORD x, WORD y, const SCR_RECT *Clip, WORD Attr);
/* Draws (4) ellipse pixels for (dx,dy) relative to ellipse centre (x,y) */

static void DrawClippedChar
  (char c, WORD x, WORD y, WORD Attr, WORD dx, WORD Width, WORD dy, WORD Height);
/* Draws clipped character, with prepared clipping extents */

static void ClipLine(WIN_RECT *Line, const SCR_RECT *Clip);
static void ClipRegion(WORD *dest, const SCR_RECT *Clip, WORD x, WORD y );
/* store to dest flags indicating where (x,y) with respect to Clip? (ClipLine)*/
```

```
SCR_STATE SysScr; //current state
WORD CharHgt;
SCR_RECT ScrRect;
#define ScrRect ASAP_ref(47, SCR_RECT (*const))

WORD SysFontX, SysFontY;

BYTE GL_ShadeMods[2];
WORD GL_ShadeOffset;
WORD GL_ScrScanSize;
```

Windows

winrect deals with borders

- win
- winrect

win.o

```
typedef signed short WIN_COORDS;

#define BITMAP_HDR_SIZE   4
typedef struct
{
  WORD NumRows;
  WORD NumCols;
  BYTE Data[1];   /* should be [] */
} BITMAP;

typedef struct {
  WIN_COORDS x0, y0;
  WIN_COORDS x1, y1;
} WIN_RECT;

typedef struct {
  WIN_COORDS x0, y0;
} WIN_POINT;

/*  Defines / Structures  */

#define WinEnd(w)
#define WinSetCursor(w,x,y) (w)->CursorX=x; (w)->CursorY=y
#define WinShow(w) (w)->Flags |= WF_VISIBLE

enum WinFlags { WF_SYS_ALLOC=0x0001, WF_STEAL_MEM=0x0002,
            WF_DONT_REALLOC=0x0004, WF_ROUNDEDBORDER=0x0008,
            WF_SAVE_SCR=0x0010, WF_DUP_SCR=0x0020, WF_TTY=0x0040,
            WF_ACTIVE=0x0080, WF_NOBORDER=0x0100, WF_NOBOLD=0x0200,
            WF_DUP_ON=0x0400, WF_VIRTUAL=0x0800, WF_TITLE=0x1000,
            WF_DIRTY=0x2000, WF_TRY_SAVE_SCR=0x4010, WF_VISIBLE=0x8000 };

typedef struct WindowStruct {
  WORD Flags;            /* 00 WinFlags */
  BYTE CurFont;          /* 02 Current font */
  BYTE CurAttr;          /* 03 Current text/line attribute */
  BYTE Background;       /* 04 Background attribute */
  SINT TaskId;           /* 06 Task ID of owner */
  WIN_COORDS CurX, CurY; /* 08 Current X,Y position (window coordinates) */
  WIN_COORDS CursorX, CursorY; /* 0C Cursor X,Y position */
  SCR_RECT Client;       /* 10 Client region of window (excludes border) */
  SCR_RECT Window;       /* 14 Window region (entire window inc border) */
  SCR_RECT Clip;         /* 18 Current clipping region */
  SCR_RECT Port;         /* 1C Port region for DUP_SCR */
  HANDLE DupScr;         /* 20 Duplicated or saved screen area */
  struct WindowStruct *Next;   /* 22 Windows kept in linked list */
  char *Title;           /* 26 Optional title */
```

```
} WINDOW;

//Unimplemented functions/macros:
void ValidateWin( WINDOW *w ); //not defined, probably very messy
void WinIcon( WINDOW *w, WIN_COORDS x0, WIN_COORDS y0, const WORD *Icon )
/*****************************************************************************
 WinIcon - this is not defined, and may look something like this:
           there is no Attr parameter to this function!
*/
{
  if(w->Flags & WF_DUP_ON) {
    PortSet(DEREF(w->DupScr), w->Port.x1, w->Port.y1);
    DrawIcon(x0, y0, Icon, A_NORMAL);
    PortRestore ();
  }
  if(w->Flags & WF_VIRTUAL)
    DrawIcon(x0, y0, Icon, A_NORMAL);
}

static const WINDOW DeskTop0;

WIN_COORDS WinWidth( WINDOW *w );
#define WinWidth ASAP_ref(657, WIN_COORDS (*const)( WINDOW * ))
WIN_COORDS WinHeight( WINDOW *w );
#define WinHeight ASAP_ref(658, WIN_COORDS (*const)( WINDOW * ))


void WinActivate( WINDOW *w );
#define WinActivate ASAP_ref(1, void (*const)(WINDOW *))
BYTE WinAttr( WINDOW *w, BYTE Attr );
#define WinAttr ASAP_ref(2,BYTE (*const)(WINDOW *,BYTE))
void WinBackupToScr( WINDOW * w);
#define WinBackupToScr ASAP_ref(3, void (*const)(WINDOW *))
void WinBackground( WINDOW *w, BYTE Attr );
#define WinBackground ASAP_ref(4, void (*const)(WINDOW *,BYTE))
void WinBegin( WINDOW *w );
#define WinBegin ASAP_ref(5, void (*const)(WINDOW *))
BOOL WinBitmapGet( WINDOW *w, const WIN_RECT *WinRect, BITMAP *Bitmap );
#define WinBitmapGet ASAP_ref(6, BOOL (*const)(WINDOW *,const WIN_RECT *,BITMAP *))
void WinBitmapPut( WINDOW *w, SWORD x0, SWORD y0, BITMAP *Bitmap, WORD Attr);
#define WinBitmapPut ASAP_ref(7, void (*const)(WINDOW *,SWORD,SWORD,BITMAP *,WORD))
WORD WinBitmapSize( WINDOW *w, const WIN_RECT *WinRect );
#define WinBitmapSize ASAP_ref(8, WORD (*const)(WINDOW *,const WIN_RECT *))
void WinChar( WINDOW *w, char c );
#define WinChar ASAP_ref(10, void (*const)(WINDOW *,char))
void WinCharXY(WINDOW *w, WIN_COORDS x, WIN_COORDS y, char c, short Count );
#define WinCharXY ASAP_ref(9, void (*const)(WINDOW *, WIN_COORDS, WIN_COORDS, char,
short))
void WinClose( WINDOW *w );
#define WinClose ASAP_ref(11, void (*const)(WINDOW *))
void WinClr( WINDOW *w );
#define WinClr ASAP_ref(12, void (*const)(WINDOW *))
void WinDeactivate( WINDOW *w );
#define WinDeactivate ASAP_ref(13, void (*const)(WINDOW *))
void WinDupStat( WINDOW *w, BOOL Stat );
#define WinDupStat ASAP_ref(14, BOOL (*const)(WINDOW *,BOOL))
void WinEllipse( WINDOW *w, WIN_COORDS x0, WIN_COORDS y0, WIN_COORDS a0, WIN_COORDS b0 );
#define WinEllipse ASAP_ref(15, void (*const)(WINDOW *, WIN_COORDS, WIN_COORDS,
WIN_COORDS, WIN_COORDS))
void WinFill( WINDOW *w, const WIN_RECT *r, BYTE Attr );
#define WinFill ASAP_ref(16, void (*const)(WINDOW *,const WIN_RECT *,BYTE))
void WinFillTriangle( WINDOW *w, WIN_COORDS ax, WIN_COORDS ay, WIN_COORDS bx,
  WIN_COORDS by, WIN_COORDS cx, WIN_COORDS cy, BYTE Attr );
#define WinFillTriangle ASAP_ref(18, void (*const)(WINDOW *, WIN_COORDS, WIN_COORDS,
WIN_COORDS, WIN_COORDS, WIN_COORDS, WIN_COORDS, BYTE))
void WinFont( WINDOW *w, BYTE Font );
#define WinFont ASAP_ref(19, void (*const)(WINDOW *,BYTE))
void WinHide( WINDOW *w );
#define WinHide ASAP_ref(21, void (*const)(WINDOW *))
```

```
void WinHome( WINDOW *w );
#define WinHome ASAP_ref(22, void (*const)(WINDOW *))

void WinInit( void );

void WinLine( WINDOW *w, const WIN_RECT *Line );
#define WinLine ASAP_ref(23, void (*const)(WINDOW *,const WIN_RECT *))
void WinLineNC( WINDOW *w, const WIN_RECT *Line );
#define WinLineNC ASAP_ref(24, void (*const)(WINDOW *,const WIN_RECT *))
void WinLineTo( WINDOW *w, WIN_COORDS x0, WIN_COORDS y0 );
#define WinLineTo ASAP_ref(25, void (*const)(WINDOW *,WIN_COORDS,WIN_COORDS))
void WinLineRel( WINDOW *w, WIN_COORDS x0, WIN_COORDS y0 );
#define WinLineRel ASAP_ref(26, void (*const)(WINDOW *,WIN_COORDS,WIN_COORDS))
void WinMoveTo( WINDOW *w, WIN_COORDS x0, WIN_COORDS y0 );
#define WinMoveTo ASAP_ref(28, void (*const)(WINDOW *,WIN_COORDS,WIN_COORDS))
void WinGetCursor(WINDOW *w,WIN_COORDS*x,WIN_COORDS*y);
#define WinGetCursor ASAP_ref(20, void (* const 418)(WINDOW *,WIN_COORDS *,WIN_COORDS *))
void WinFillLines2( WINDOW *w, const WIN_RECT *l0, const WIN_RECT *l1, BYTE Attr );
#define WinFillLines2 ASAP_ref(17, void (*const)(WINDOW *,const WIN_RECT *,const WIN_RECT
*,BYTE))
void WinMoveCursor( WINDOW *w, WIN_COORDS x, WIN_COORDS y );
#define WinMoveCursor ASAP_ref(27, void (*const)(WINDOW *, WIN_COORDS, WIN_COORDS))
void WinMoveRel( WINDOW *w, WIN_COORDS x0, WIN_COORDS y0 );
#define WinMoveRel ASAP_ref(29, void (*const)(WINDOW *,WIN_COORDS,WIN_COORDS))
BOOL WinOpen( WINDOW *w, const WIN_RECT *r, WORD Flags, ... );
#define WinOpen ASAP_ref(30, BOOL (*const)(WINDOW *,const WIN_RECT *,WORD,...))
BOOL WinReOpen( WINDOW *w, const WIN_RECT *wr, WORD Flags, ... );
#define WinReOpen ASAP_ref(34, BOOL (*const)(WINDOW *,const WIN_RECT *, WORD, ...))
WORD WinPixGet( WINDOW *w, WIN_COORDS x0, WIN_COORDS y0 );
#define WinPixGet ASAP_ref(31, WORD (*const)(WINDOW *,WIN_COORDS,WIN_COORDS))
void WinPixSet( WINDOW *w, WIN_COORDS x0, WIN_COORDS y0 );
#define WinPixSet ASAP_ref(32, void (*const)(WINDOW *,WIN_COORDS,WIN_COORDS))
void WinRect( WINDOW *w, const WIN_RECT *r, BYTE BoxAttr );
#define WinRect ASAP_ref(33, void (*const)(WINDOW *,const WIN_RECT *,BYTE))
void WinScrollV( WINDOW *w, const WIN_RECT *r, SWORD NumRows );
#define WinScrollV ASAP_ref(36, void (*const)(WINDOW *,const WIN_RECT *,SWORD))
void WinScrollH( WINDOW *w, const WIN_RECT *r, SWORD NumCols );
#define WinScrollH ASAP_ref(35, void (*const)(WINDOW *,const WIN_RECT *,SWORD))
void WinStrXY( WINDOW *w, WIN_COORDS x, WIN_COORDS y, char *Str );
#define WinStrXY ASAP_ref(38, void (*const)(WINDOW *,WIN_COORDS,WIN_COORDS,char *))
void WinStr( WINDOW *w, char *Str );
#define WinStr ASAP_ref(37, void (*const)(WINDOW *,char *))

static void ActivateWindow( WINDOW *w, BOOL bUpdate );
static void ClearPortRegion( WINDOW *w );
static void WinCharClip( WINDOW *w, char c );
static WINDOW *WinFind( WINDOW *w );
static void WinNewLine( WINDOW *w);
static void WinRemove( WINDOW *w, BOOL bRedrawAll);
static void SetShadeOffset( BYTE Mod1, BYTE Mod2, BYTE mode);

//external
WINDOW *FirstWindow;
#define FirstWindow ASAP_ref(0, WINDOW* (*const))
WINDOW DeskTop;
```

winrect.o

```
static const SCR_RECT ScrRect0;

void DrawWinBorder(WINDOW *w, const SCR_RECT *scr);
#define DrawWinBorder ASAP_ref(39, void (*const)(WINDOW *,const SCR_RECT *))
WIN_RECT *RectWinToWin(const SCR_RECT* rs, WIN_RECT *rw); //offset rw by rs
#define RectWinToWin ASAP_ref(41, WIN_RECT * (*const)(const SCR_RECT *,WIN_RECT *))
SCR_RECT *ScrToHome(SCR_RECT* rs); //move rs to TL corner
#define ScrToHome ASAP_ref(46, SCR_RECT * (*const)(SCR_RECT *))
SCR_RECT *RectWinToScr
  (const SCR_RECT *clip, const WIN_RECT *src, SCR_RECT *dest);
```

```
#define RectWinToScr ASAP_ref(42, SCR_RECT * (*const)(const SCR_RECT *,const WIN_RECT
*,SCR_RECT *))
WIN_RECT *ScrToWin(const SCR_RECT *r);
#define ScrToWin ASAP_ref(45, WIN_RECT * (*const)(const SCR_RECT *))
WIN_RECT *MakeWinRect(SWORD x0, SWORD y0, SWORD x1, SWORD y1);
#define MakeWinRect ASAP_ref(44, WIN_RECT * (*const)(SWORD,SWORD,SWORD,SWORD))
SCR_RECT *ScrRectDivide(const SCR_RECT *r, const SCR_RECT *, short *);
#define ScrRectDivide ASAP_ref(40, SCR_RECT * (*const)(const SCR_RECT *,const SCR_RECT
*,short *))
void UpdateWindows( const SCR_RECT *clip );
#define UpdateWindows ASAP_ref(43, void (*const)( const SCR_RECT * ))

static void UpdateMenu( void );

.bss static
static SCR_RECT _ScrRectDivide[4];
static WIN_RECT _MakeWinRect;
ststic WIN_RECT _ScrToWin;
```

User Input

- Text Editor
- Menus
- Dialogs

Text Editor (te.o)

```
typedef struct TextEditStruct {
  WINDOW *Parent; /* 00 Edit Window */
  WORD ReadOnly;  /* 04 Number of Bytes at start that are Read Only */
  WIN_RECT Rect;  /* 06 Edit Rect */
  WORD BufSize;   /* 0E Allocated currently */
  WORD CurSize;   /* 10 Current Length */
  WORD CursorOffset;    /* 12 Offset of Cursor */
  WORD StartOffset;     /* 14 ScrollX, position at which text is displayed */
  WORD PreChars;  /* 16 Number of characters to display before ":" */
  WORD CharWidth; /* 18 width in characters */
  WORD CharHeight;/* 1A number of characters high */
  WORD LineNum;   /* 1C Line number cursor is on 0...CharHeight-1 */
  WORD CursorX;   /* 1E Horz char position (in line or rel to StartOffset) */
  WORD Flags;     /* 20 TextFlags */
  union {         /* 22 TEXT Data */
    HANDLE h;           //TE_open
    const char* p;      //TE_openFixed
  } Text;
} TEXT_EDIT;

enum { SE_CODE=2 }; //Select marker
```

```
void TE_checkSlack(TEXT_EDIT *te);
#define TE_checkSlack ASAP_ref(166, void (*const)(TEXT_EDIT *))
/* if not much space left, realloc buffer */

void TE_close(TEXT_EDIT *te);
#define TE_close ASAP_ref(165, void (*const)(TEXT_EDIT *))
void TE_empty(TEXT_EDIT *te);
#define TE_empty ASAP_ref(167, void (*const)(TEXT_EDIT *))
BOOL TE_focus(TEXT_EDIT *te);
#define TE_focus ASAP_ref(168, BOOL (*const)(TEXT_EDIT *))
BOOL TE_handleEvent(TEXT_EDIT *te, EVENT *event);
#define TE_handleEvent ASAP_ref(169, BOOL (*const)(TEXT_EDIT *, EVENT *))
void TE_indicateReadOnly(TEXT_EDIT *te);
#define TE_indicateReadOnly ASAP_ref(170, void (*const)(TEXT_EDIT *))
/* set appropriate command states (cut, paste, del, clear) and status line */

void TextInit( void );
```

```
BOOL TE_isBlank(TEXT_EDIT *te);
#define TE_isBlank ASAP_ref(171, BOOL (*const)(TEXT_EDIT *))

BOOL TE_open
(TEXT_EDIT *te, WINDOW *w, WIN_RECT *r, HANDLE init, WORD ro, WORD offset, WORD Flags);
#define TE_open ASAP_ref(172, BOOL (*const)(TEXT_EDIT *, WINDOW *, WIN_RECT *, HANDLE,
WORD, WORD, WORD))
BOOL TE_openFixed
  (TEXT_EDIT *te, WINDOW *w, WIN_RECT *r, const char *text, WORD maxlen, WORD Flags);
#define TE_openFixed ASAP_ref(173, BOOL (*const)(TEXT_EDIT *, WINDOW *, WIN_RECT *, const
char *, WORD, WORD))

static open(TEXT_EDIT *te, WINDOW *w, WIN_RECT *r);

static void DimScrToChar(TEXT_EDIT *te);
/* Fills in width and height fields in te from current fond and coords */

void TE_pasteText(TEXT_EDIT *te, const TEXT text , size_t len);
#define TE_pasteText ASAP_ref(174, void (*const)(TEXT_EDIT *, const TEXT, size_t))

void TE_reopen(TEXT_EDIT *te, BOOL bFocus);
#define TE_reopen ASAP_ref(175, void (*const)(TEXT_EDIT *, BOOL))
void TE_reopenPlain(TEXT_EDIT *te, BOOL bFocus);
#define TE_reopenPlain ASAP_ref(176, void (*const)(TEXT_EDIT *, BOOL))

void TE_select(TEXT_EDIT *te, WORD Low, WORD High);
#define TE_select ASAP_ref(177, void (*const)(TEXT_EDIT *, WORD, WORD))
/* Selects the text Low-High, positions cursor if Low==High
this places char(2) at Low and High in the text buffer
*/

void TE_shrinkWrap(TEXT_EDIT *te);
#define TE_shrinkWrap ASAP_ref(178, void (*const)(TEXT_EDIT *))
/* Removes the select characters from the buffer
*/

BOOL TE_unfocus(TEXT_EDIT *te);
#define TE_unfocus ASAP_ref(179, BOOL (*const)(TEXT_EDIT *))
void TE_updateCommand(TEXT_EDIT *te, BYTE cmd);
#define TE_updateCommand ASAP_ref(180, void (*const)(TEXT_EDIT *, BYTE))


//more static functions (not particularly useful)

static WORD CurHSEntry;

char FindText[20]; //FindDlgBuf
```

Menus

- Standard Routines
- Custom Menus
- Variable Dialogs

Standard Routines (menu.o)

Menus are automated by a dynamic structure, which is interpreted at runtime. There are two types of menus Popups and standard (toolbar) Menus. Popups allow a subset of the features of Menus to be used.

Toolbar or top meus require dynamic storage for enabling/disabling and checking/unchecking menu options and, less frequently, changing the icons and selecting the top of the menu. These have separate drawing routines and need to be sent the keypresses so that they can open up when the appropriate function keys are pressed.

```
//MenuBegin(struct) or use MenuNew, MenuAdd, MenuBegin to build the menu
```

```
enum MENU_TYPE { MT_TEXT=0x8000, MT_XREF=0x9000, MT_ICON=0xA000 /* ((n*0x1000)|0x8000) */
                 MT_CASCADE=0x4000 }; /* top 4 bits */
/* If ICON offset not defined, it is a Redef type */

typedef union {
  struct {
    WORD ID;      /* MENU_TYPE | ID returned */
    WORD Val;     /* offset of ICON or TEXT, or XR_String value (NB top bit clear) */
    //WORD CascadeOffset; /* if( ID&&MT_CASCADE ) */
  } entry;
  DWORD l;        /* -1 is end terminator */
} MENU_ENTRY;

/* Note: The top bits of MT_TEXT, MT_XREF & MT_ICON are set because MENU_ENTRY is a
 * variable length structure (4 or 6 bytes) depending on whether a cascade is
 * defined. To correctly move between entry indexes, i.e. up & down, the following occurs
 * .down - if MT_CASCADE set move down 6 bytes, otherwise move down 4 bytes
 * .up - move back 4 bytes if bit is clear => on entry.Val, move back another 2 bytes
 */

typedef struct {
  WORD ID;              /* without the MENU_TYPE | */
  ICON Icon;
} MENU_REDEF;

typedef struct {
  WORD DisplayOffset;
  WORD Flags;
  WORD TotalItems;
  SCR_COORDS Width, Height;
  WORD MainItems;
  WORD DynSize;         /* for dynamic building of menus */
  WORD RedefOffset;
  WORD RedefItems;
  BYTE Data[1];         /* variable length structure should be [] */
} MENU;

typedef struct {
  MENU *Menu;           /* 0x0 Menu structure this belongs to */
  MENU_ENTRY *StartEntry;     /* 0x4 Current MENU_ENTRY open */
  SCR_RECT Rect;        /* 0x8 Top Bar */
  WORD Flags;           /* 0xC MenuFlags */
  WORD TopSelect;       /* 0xE Top number selected (see Geom) (MenuTopSelect) */
  WORD TopStatIndex;    /* 0x10 Top flags, Index in Data[] (0) (MenuTopStat) */
  WORD SubStatIndex;    /* 0x12 (MenuSubStat) */
  WORD CheckIndex;      /* 0x14 (MenuCheck) */
  WORD TopRedefIndex;   /* 0x16 Geometry (MenuTopRedef) */
  BYTE Data[1];         /* 0x18 flags - should be [] */
} MENU_STATE;
/* Stat and Check take 1 bit per item, RedefIndex's are each one WORD */
```

```
static const BYTE BITS[8] = { 1, 2, 4, 8, 0x10, 0x20, 0x40, 0x80 };
static const MULTI_LINES TopTab[16] = { 3,
  { A_REVERSE, -2, 0, 2, 0, },
  { A_NORMAL, -3, 0, 0, 3, },
  { A_NORMAL, 0, 3, 3, 0, },
};
#ifndef AMS2
static const MULTI_LINES BottomTab[16] = { 3, //Not used
  { A_REVERSE, -2, 0, 2, 0, },
  { A_NORMAL, -3, 0, 0, -3, },
  { A_NORMAL, 0, -3, 3, 0, },
};
#endif
```

```
void MenuInit( void );
```

```
void MenuUpdate( void );
#define MenuUpdate ASAP_ref(73, void (*const)( void ))

WORD MenuPopup(MENU *Menu, WORD x, WORD y, WORD ID);
#define MenuPopup ASAP_ref(59, WORD (*const)(MENU *, WORD, WORD, WORD))
/* Displays popup menu, and returns selected item
ID is starting item to select
*/

//(toolbar) menus which maintain the state in a handle: (activated by MenuKey)
void MenuSubStat(HANDLE State, WORD ID, BOOL Status);
#define MenuSubStat ASAP_ref(60, void (*const)(HANDLE, WORD, BOOL))
/* Set Enabled/Disabled flag of sub-entry with event ID */

void MenuTopStat(HANDLE State, WORD Top, BOOL Status);
#define MenuTopStat ASAP_ref(61, void (*const)(HANDLE, WORD, BOOL))
/* Set Enabled/Disabled flag of Top menu */

void MenuCheck(HANDLE State, WORD ID, WORD Check);
#define MenuCheck ASAP_ref(55, void (*const)(HANDLE, WORD, WORD))
/* Set Checked/Unchecked flag of sub-entry with event ID
Should be BOOL? although 2 is actually considered for some reason
*/

HANDLE MenuBegin(MENU *Menu, WORD x, WORD y, WORD Flags);
#define MenuBegin ASAP_ref(54, HANDLE (*const)(MENU *, WORD, WORD, WORD))
/* creates and returns state */

void MenuOn(HANDLE State);
#define MenuOn ASAP_ref(58, void (*const)(HANDLE))
/* draws dynamic menu (after MenuBegin etc) */

//changing the top states (see Geometry)
void MenuTopSelect(HANDLE State, WORD Top);
#define MenuTopSelect ASAP_ref(62, void (*const)(HANDLE, WORD))
/* Select (draws thick surround) Menu Top
-1 for none */

void MenuTopRedef(HANDLE State, WORD Top, WORD Index);
#define MenuTopRedef ASAP_ref(63, void (*const)(HANDLE, WORD, WORD))
/* Set the Top Menu icon to new Index */

WORD MenuGetTopRedef(HANDLE State, WORD Top);
#define MenuGetTopRedef ASAP_ref(64, WORD (*const)(HANDLE, WORD))
/* Get the Index of Top Menu num's icon */

void MenuEnd(HANDLE State);
#define MenuEnd ASAP_ref(56, void (*const)(HANDLE))
/* deallocate state etc. */

WORD MenuKey(HANDLE State, WORD KeyCode);
#define MenuKey ASAP_ref(57, WORD (*const)(HANDLE, WORD))
/* Send KeyCode to (state) menu, used to drop down appropriate menu when FKey pressed
return the value if item selected (-2 if bad KeyCode, 0 if esc) */

//Create MENU dynamically (used for custom menus, see below)
//Note: once created dynamically this structure can be used as a resource
HANDLE MenuAddText(HANDLE h, WORD Link, const char *Text, WORD ID, WORD Flags);
#define MenuAddText ASAP_ref(65, HANDLE (*const)(HANDLE, WORD, const char *, WORD, WORD))

HANDLE MenuAddIcon(HANDLE h, WORD Link, pICON Icon, WORD ID, WORD Flags);
#define MenuAddIcon ASAP_ref(66, HANDLE (*const)(HANDLE, WORD, pICON, WORD, WORD))

HANDLE MenuNew(WORD Flags, WORD Width, WORD Height);
#define MenuNew ASAP_ref(67, HANDLE (*const)(WORD, WORD, WORD))

//Create popup MENU, these are simplified menus, same structure though
HANDLE PopupAddText(HANDLE h, WORD Link, const char *Text, WORD ID);
#define PopupAddText ASAP_ref(68, HANDLE (*const)(HANDLE, WORD, const char *, WORD))
```

```
HANDLE PopupClear(HANDLE h);
#define PopupClear ASAP_ref(70, HANDLE (*const)(HANDLE))

HANDLE PopupNew(const char *Title, WORD Height);
#define PopupNew ASAP_ref(69, HANDLE (*const)(const char *, WORD))

WORD PopupDo(HANDLE h, WORD x, WORD y, WORD ID);
#define PopupDo ASAP_ref(71, WORD (*const)(HANDLE, WORD, WORD, WORD))
/* Display dynamically created menu
calls MenuPopup
*/

char *PopupText(HANDLE h, WORD ID);
#define PopupText ASAP_ref(72, char *(*const) (HANDLE, WORD))
/* returns name associated with ID */

static void *GetItemInfo(MENU *Menu, MENU_ITEM *Cur);
/* Return either MenuItemName or MENU_REDEF for Cur */

static MENU_ITEM *FindPopupTextItem(HANDLE h, WORD ID);
/* for PopupText */

static HANDLE PopupInit(HANDLE h, char *Title, WORD Height);
/* PopupClear & PopupNew use this */

static HANDLE MenuAdd(HANDLE h, WORD Link, const void *Ptr, WORD ID, WORD Flags);

static void PaintPopup(HANDLE State, MENU *Menu, MENU_ENTRY *Start, MENU_ENTRY *Cur,
MENU_ENTRY *Last, pIcon OptionalIcon);
/* Every time popup is opened or redrawn */

static void PaintItem(HANDLE State, MENU_ENTRY *Cur, const void *Item, BOOL Draw);
/* Draws Icon or Text from GetItemInfo */

static void PaintTop(HANDLE State, MENU_ENTRY *Start, WORD Attrs /* BOOL Draw */);
/* Draw entire Top toolbar */

static void MenuRect(SCR_RECT *Dest, MENU *Menu, WORD x, WORD y);
/* Fill in Dest rect from Menu at (x,y) */

WORD GetPopupOffset(MENU *Menu, WORD ID);
/* Find offset (number from top) of item with ID in menu structure */

static WORD GetIDOffset(MENU *Menu, WORD ID);
static WORD GetIDOffset_aux(MENU *Menu, WORD ID, BOOL noscan);
/* returns offset in a0 */

static WORD CountAllItems(MENU_ENTRY *Cascade, BOOL CountIconsTwice, WORD *CascadeCount);

static MENU_ENTRY *GetCascade(MENU *Menu, MENU_ENTRY *Cur);

static MENU_ENTRY *MoveDown(MENU_ENTRY *Cur);
/* Note at top this will wrap around to bottom (although this is never allowed on AMS1)
*/

static MENU_ENTRY *MoveUp(MENU_ENTRY *Cur);
/* As with MoveDown, at bottom will wrap to top */

//These checks to stop above 2 from wrapping on AMS1
static BOOL AtTop(MENU_ENTRY *Cur);
static BOOL AtBottom(MENU_ENTRY *Cur);

static WORD DefaultWidth(MENU *Menu, MENU_ENTRY *Cur);
/* Returns size or minimum default if "small" (used for top menus) */

static MENU_ENTRY *GetUptoWidth(WORD Width, MENU *Menu, MENU_ENTRY *Start);
/* Strange - seems to sum up widths unti Width is reached, returning MENU_ENTRY */
```

```
static MENU_ENTRY *LastItemFit(WORD Height, MENU_ENTRY *Start);
/* Returns the last entry that will fit within the given height */

char *MenuItemName(MENU *Menu, MENU_ENTRY *Cur);
/* Get the name of cur, either from structure or XR name */

static WORD CountBetweenItems(MENU_ENTRY *Start, MENU_ENTRY *Cur, BOOL CountIconsTwice);

static MENU_REDEF *GetRedef(MENU *Menu, WORD ID);
/* Do a binary search in Redef section of Menu for ID */

static MENU_ENTRY *MenuItemN(MENU_ENTRY *Start, WORD num);
/* used for finding offset of popup in menu structure (for FKeys etc.) */

static MENU_ENTRY *Find(MENU *Menu, MENU_ENTRY *Start, char c);
/* searches for entry starting with char 'c' */

MENU_ENTRY *MenuItemNSub(MENU *Menu, MENU_ENTRY *start, WORD num);
/* Return pointer to ItemN in (Sub) Menu, ignoring popup menus in count */

static WORD GetFKey(KEYCODE key);
/* Return FKey num or -2 if key out of (FKey) range */

static WORD FindWidth(MENU *Menu, MENU_ENTRY *Start);
/* Finds maximum width of all items from Start to end of entries */

static void InvertTop(HANDLE State, MENU_ENTRY *Start, MENU_ENTRY *Cur);
/* Used in deselecting & selecting */

static BYTE InvertSub(SCR_RECT r, MENU_ENTRY *Start, MENU_ENTRY *Cur, WORD yOffset);
/* Returns r.y1 */

static BYTE NumToChar(BYTE c);
/* isn't supposed to go above 8 */

static WORD GetJumpKey(KEYCODE key);
/* Return number from top, or -1 if out of range */

static BOOL GetTopStat(MENU_STATE *State, WORD Top);
/* returns the state of Top, non-zero if enabled etc.
(Not true BOOL returned, shifted according to position)
*/

static BOOL GetSubStat(HANDLE State, MENU_ENTRY *Cur, WORD *Offset);
/* if found Offset is set, otherwise 1 is retured (default - enabled)
If Offset isn't supplied, it is looked up from cur
(Not true BOOL returned, shifted according to position)
*/

static BOOL GetCheck(HANDLE State, MENU_ENTRY *Cur, WORD *Offset);
/* As GetSubStat, but returns default value of 0 (unchecked)
(Not true BOOL returned, shifted according to position)
*/

static WORD CountItems(MENU *Menu, WORD PopupOffset)

BOOL is_cascade(MENU_ENTRY *Cur);

static WORD DoPopup(HANDLE State, MENU *Menu, MENU_ENTRY *Cur, MENU_ENTRY *Start, WORD x,
WORD y, WORD StartOffset);
/* State or Menu & entry */

// Save system states before and after menu
static void MenuBackup( void );
static void MenuRestore( void );

static WORD *GetTopRedef(MENU_STATE *State, WORD num);

static WORD FindOffset(MENU *Menu, MENU_ENTRY *Start);
```

```
/* Recursivly search for global FindID in Menu, returning Index in cascade
-1 if not found
*/

static pICON TopDef;
static MENU_ENTRY *CurEntry;
static WORD x, y;
static WORD tmp;

WORD StatPos;
KEYCODE KeyPress;
WORD FindID;

WINDOW wMenu;
BOOL OldCursorActive;
SCR_STATE OldScrState;

WORD TopXPos[11];

BOOL DrawTopCascade;    /* Haven't seen this in use */
WORD FirstFKey;
WORD LastFKey;
WORD Cascades;    /* Recursion count */
```

```
struct MENU_ENTRY {
  WORD ID;  /* MENU_TYPE | ID returned */
  WORD Val; /* offset of ICON or TEXT, or XR_String value */
  packed union {  /* if cascade, we have an extra MENU_ENTRY offset: */
    WORD CascadeOffset;
  } extras;
};

struct MENU {
  WORD DisplayOffset;   /* Text or Icon */
  WORD Flags;
  WORD TotalItems;
  SCR_COORDS Width, Height;
  WORD MainItems;
  WORD DynSize;         /* 0 for const */
  WORD RedefOffset;     /* Redefinable icons */
  WORD RedefItems;
  DWORD separator=-1;

  MENU_ENTRY titles[];
  DWORD separator=-1;

  struct {
    MENU_ENTRY Entry;
    DWORD separator=-1;
  } cascade[];          /* defined later in structure */

  packed union {
    ICON Icon;
    TEXT Name;
  } Display[];
  MENU_REDEF RedefIcons[];
};
```

Custom Menus (custom.o)

```
void CustomBegin( void );
#define CustomBegin ASAP_ref(330, void (*const)( void ))

BOOL CustomMenuItem(WORD id);
#define CustomMenuItem ASAP_ref(331, BOOL (*const)(WORD))
/* Sends custom menu string to current app
id = result from MenuKey
```

```
*/

void CustomEnd( void );
#define CustomEnd ASAP_ref(332, void (*const)( void ))
void CustomFree( void );
#define CustomFree ASAP_ref(336, void (*const)( void ))
BOOL ReallocExprStruct(HANDLE struct, WORD , EStackIndex);
#define ReallocExprStruct ASAP_ref(333, BOOL (*const)(HANDLE, WORD , EStackIndex))
char *SearchExprStruct(HANDLE struct, WORD id);
#define SearchExprStruct ASAP_ref(334, char *(*const)(HANDLE, WORD))
void handleRclKey(BOOL fromevent);
#define handleRclKey ASAP_ref(335, void (*const)(BOOL))

HANDLE hCustomMenu, hCustomExprStruct, hCustomMenuState;
BOOL bRclDlgActive;
```

Dialogs and Standard Dialogs

- Dialogs
- Dialog Messages

Dialogs (dialog.o)

```
enum Buttons { NONE, SB_OK, SB_SAVE, SB_YES, SB_CANCEL, SB_NO, SB_GOTO };

//For code use, see variable structure definition later
typedef DialogHdrStruct {
  WORD EntryData;
  WORD Flags;
  SCR_COORDS Width, Height;
  DWORD (*CallBack)(WORD, DWORD);
  BYTE Data[1];          /* variable length structure should be [] */
} DIALOG;
```

```
static const char * const SB_Names[6] = {
  "Enter=OK",
  "Enter=SAVE",
  "Enter=YES",
  "ESC=CANCEL",
  "ESC=NO",
  "Enter=Goto"
};
```

```
WORD Dialog
  (const DIALOG *Dialog, WORD x, WORD y, char *text_buf, WORD *popup_buf);
#define Dialog ASAP_ref(48, WORD (*const)(const DIALOG *, WORD, WORD, char *, WORD *))
/* Display dialog structure
text_buf[]: for all text input boxes
popup_buf[]: for all popup menus
returns key used to clear dialog
*/

#ifndef _NOCALLBACK
#define _NOCALLBACK
DWORD NoCallBack(WORD a,DWORD b) {return 1;}
//#define NoCallBack ASAP_ref(49, DWORD (*const)(WORD,DWORD))
#endif
/* call has been replaced in jump table on AMS2 */

KEYCODE DialogDo(HANDLE h, WORD x, WORD y, char *text_buf, WORD *popup_buf);
#define DialogDo ASAP_ref(50, KEYCODE (*const)(HANDLE h, WORD, WORD, char *, WORD *))
/* returns keypress */

HANDLE DialogNew(WORD Wid, WORD Hgt, DWORD (*CallBack)(WORD, DWORD));
#define DialogNew ASAP_ref(52, HANDLE (*const)(WORD, WORD, DWORD (*)(WORD, DWORD)))
/* returns handle of allocated dialog */

HANDLE DialogAdd(HANDLE h, WORD Flags, BYTE x, BYTE y, BYTE Type, ...);
#define DialogAdd ASAP_ref(51, HANDLE (*const)(HANDLE, WORD, BYTE, BYTE, BYTE, ...))
```

```
/* return handle (h) or H_NULL on error (unsupported Type) */

void DrawStaticButton(WINDOW *w, WORD SB_num, WORD x);
#define DrawStaticButton ASAP_ref(53, void (*const)(WINDOW *, WORD, WORD))

#ifndef PLUS
static void DlgText(WORD *?, WORD XR_String);
#endif

static WIN_RECT *PopupDraw
  (DIALOG *Dialog, DLG_ENTRY *popup, WORD ID, WINDOW *w, WORD *buf, BOOL Str, WORD yAsc);
/* Handle Popup, calling MenuPopup
Dialog callback is called with (a=-2, ID), and (a=ID,)
*buf is the address in popup_buf
returns the WIN_RECT of the popup box (when closed)
*/

static KEYCODE EditDraw
  (DIALOG *Dialog, DLG_ENTRY *request, WORD ID, WINDOW *w, BYTE *buf, BOOL *Str, WORD
yAsc);
/* Handle Request, using TE session CurTE
returns the keypress used to end the session
*/

static void DialogEventHandler(EVENT *event);
/* EditDraw event handler */

.bss static
static WIN_RECT _PopupDraw;
static TEXT_EDIT *CurTE;
```

ID	Description	Union Member	DialogAdd Extra Parameters
0x1	Menu	menu	
0x2	Request (empty box)	request	TEXT, WORD offset, WORD maxlen, WORD Width
0x3	Scroll Info (long dialogs)	scroll	BYTE sWid, BYTE sHgt, BYTE first, BYTE t, BYTE p, BYTE t, BYTE
0x7	Text	text	TEXT
0x8	Title [and buttons]	buttons	TEXT, BYTE b1, BYTE b2
0xA	DropDown (static popup)	dropdown	TEXT, MENU *popup, WORD num
0xB	DropDown (dynamic popup callback)	dyn_drop	
0xC	Request (start with value)	request	TEXT, …
0xD	Buttons (when title not defined)	buttons	
0xE	DropDown (dynamic popup…code ver)		TEXT, HANDLE popup, WORD num

Can use either Dialog with the following structure, or manually add the items by calling DialogNew, DialogAdd[], and DialogDo.

```
struct DLG_ENTRY {
  struct {
    BYTE type;
    BYTE Num; //id.Num that is returned by dialog entry
  } id;
  SCR_COORDS x,y;
  union extras {
    struct {
      MENU* m;
    } menu;
    struct {
      WORD text_offset;
      WORD offset;
      WORD Maxlen;
      BYTE Width;
    } request;
    struct {
      BYTE ScrollWidth, ScrollHeight;
```

```
      BYTE first;
      BYTE total_items;
      BYTE page_items;
      BYTE total_items?;
      BYTE max?;
    } scroll;
    struct {
      WORD text_offset;
    } text;
    struct {
      WORD text_offset;
      BYTE b1, b2; // SB_Buttons value
    } buttons;
    struct {
      WORD text_offset;
      MENU* dropdown;
      WORD index; /* this governs where item is stored in buffer */
    } dropdown; //external to this structure
    struct {
      WORD text_offset;
      HANDLE (*pf)(): //callback function to return handle of PopUp
      WORD index;
    } dyn_drop;
  }extras;
};


packed struct DIALOG {
  WORD Name-this;
  WORD Flags;
  SCR_COORDS Width, Height;
  DWORD (*DlgCallBack)(WORD, DWORD);

  DIALOG_ENTRY titles[];
  WORD 0;
  DWORD separator=-1;

  struct {
    MENU_ENTRY Entry;
    DWORD separator=-1;
  } cascade[];

  TEXT Name;
};
```

```
DWORD CallBack(WORD a,DWORD b);
```

```
appears to be SWORD CallBack(WORD a, DWORD b);
doesn't make a difference though...use SWORD form for now

a=operation, b=id/offset
on creation etc. called for each dialog line, can decide what to do with each line.
```

Dialog Messages (gmessage.o)

```
KEYCODE DlgMessage(const char *title, const char *text, WORD Btn1, WORD Btn2);
#define DlgMessage ASAP_ref(436, KEYCODE (*const)(const char *, const char *, WORD,
WORD))
/* Standard dialog, returns keypress */
```

Variable Dialogs (vardlg.o)

These functions implement the standard save and open dialogs. They are very easy to use, only requiring a zero terminated array of types that are supported. These types are TAGs (end tokens) listed in the token tables.

```
savetypes: zero terminated list of SAVETYPE TAGs to appear
```

```
HSYM VarOpen(const TAG *savetypes);
#define VarOpen ASAP_ref(652, HSYM (*const)(const TAG *))
HSYM VarSaveAs(const TAG *savetypes, const char *TitleSym);
#define VarSaveAs ASAP_ref(653, HSYM (*const)(const TAG *, const char *))
HYSM VarNew(const TAG *savetypes);
#define VarNew ASAP_ref(654, HYSM (*const)(const TAG *))
static HSYM VarCreate(const TAG *savetypes, BOOL New, const char *TitleSym);

enum flags { VF_ALL_ENTRY = 1, VF_MAIN_ENTRY = 2 };
HANDLE VarCreateFolderPopup(WORD* MainIndex, WORD flags);
#define VarCreateFolderPopup ASAP_ref(655, HANDLE (*const)(WORD*, WORD))

static VarRcl(char *);
BOOL VarSaveTitle(HANDLE Dialog, const char *TitleSym, char *dest);
#define VarSaveTitle ASAP_ref(656, BOOL (*const)(HANDLE, const char *, char *))
//…
```

C Standard Functions

- Formatted Text
- Function Listing

Formatted Text (printf.o)

```
static void vcbprintf(void func(char, void **), void **param, char *format, void
*va_list);
/* callback (cb) printf using pointer to list arguments (v)
func is the fucntion which for example can push characters to a stream
*/

static char *ltoa(long num, char *string, int radix, BOOL uppercase);
/* long to string (ascii) */

static void strputchr(char ch, char **p) { *((*p)++) = ch; }
/* default vcbprintf callback function for sprintf */

int sprintf(char *buffer, char *format, ...);
#define sprintf ASAP_ref(83, int (*const)(char *, char *, ...))
/* standard sprintf function, returns length of final string in buffer */
```

Format Specifiers

```
%[flags][width][.precision][{h|l|L}]type
Platform specific items highlighted.
```

Flags	Meaning
'-'	Left align
'+'	force sign (prefix +ve values)
'z'	don't postfix padding
' '	Insert space before +ve values
'#'	(prefix hex values (>0) with '0x') [does not appear to work] force '.' in float output (and prevent tructation of trailing 0s)
'^'	TI Float Format (special character exponent, no '+' prefix in exponent, 0. instead of 0)
'|'	Centre

Width	Meaning
d	min number of chars printed - padded with ' ' (d > 0)
0d	(zero prefixed) same as above but padded with '0'
'*'	width specified in arg list (before value being formatted)

Precision	Meaning
d	number of chars, decimal places, or number of significant digits (d<=16) to display depending on type (see below)
-1	default = 6 digits
'*'	precision specified in argument list (before value being formatted)

Size{h|l|L}	Meaning
'h'	short int
'l'	long int
'L'	short int

Type(s)	Meaning
'd','i','u'	{signed, signed, unsigned} decimal integer
'x','X'	(lower, upper) case hexadecimal integer
'e','E'	Float, format [-]d.dddd {e|E} [sign]ddd
'f'	Float, format [-]dddd.dddd
'g','G'	Float, format; most compact float format available ('e' or 'f') This is the most common option, used for most dialog floats.
'r','R'	Float, Engineering form
'y','Y'	Float, Mode Specified float format
'c','C'	{lower, upper} case character
's','S'	{lower, upper} case string
'p'	Pointer to void, same as %x

Special Float Value	Output
+ Infinity	"+1.*∞"
- Infinity	"-1.*∞"
Unsigned Infinity	"±1.*∞"
NAN	"1.*undef"

Function Listing (various)

```
void *_memset(void *s, int c, size_t n);
#define _memset ASAP_ref(635, void *(*const)(void *, int, size_t))

char *strtok(char *s1, const char *s2);
#define strtok ASAP_ref(634, char *(*const)(char *, const char *))

char *strpbrk(const char *s1, const char *s2);
#define strpbrk ASAP_ref(630, char *(*const)(const char *, const char *))

char *strrchr(const char *s, int c);
#define strrchr ASAP_ref(631, char *(*const)(const char *, int))

sizet_t strspn(const char *s1, const char *s2);
#define strspn ASAP_ref(632, sizet_t (*const)(const char *, const char *))

char *strstr(const char *s1, const char *s2);
#define strstr ASAP_ref(633, char *(*const)(const char *, const char *))

size_t strlen(const char *s);
#define strlen ASAP_ref(638, size_t (*const)(const char *))

char *strncat(char *dest, const char *src, size_t maxlen);
#define strncat ASAP_ref(623, char *(*const)(char *, const char *, size_t))

int strncmp(const char *s1, const char *s2, size_t maxlen);
#define strncmp ASAP_ref(626, int (*const)(const char *, const char *, size_t))

char *strncpy(char *dest, const char *src, size_t maxlen);
#define strncpy ASAP_ref(621, char *(*const)(char *, const char *, size_t))

char *strerror(int errornum);
#define strerror ASAP_ref(637, char *(*const)(int))

char *strcat(char *dest, const char *src);
#define strcat ASAP_ref(622, char *(*const)(char *, const char *))

char *strchr(const char *s, int c);
#define strchr ASAP_ref(628, char *(*const)(const char *, int))
```

```
int strcmp(const char *s1, const char *s2);
#define strcmp ASAP_ref(625, int (*const)(const char *, const char *))

char *strcpy(char *dest, const char *src);
#define strcpy ASAP_ref(620, char *(*const)(char *, const char *))

size_t strcspn(const char *s1, const char *s2);
#define strcspn ASAP_ref(629, size_t (*const)(const char *, const char *))


void _du32u32( void );
#define _du32u32 ASAP_ref(682, void (*const)( void ))
void _ds32s32( void );
#define _ds32s32 ASAP_ref(680, void (*const)( void ))


void _du16u16( void );
#define _du16u16 ASAP_ref(678, void (*const)( void ))
void _ds16u16( void );
#define _ds16u16 ASAP_ref(676, void (*const)( void ))

int rand( void );
void srand(unsigned int seed);

void _mu32u32( void );
#define _mu32u32 ASAP_ref(683, void (*const)( void ))
void _ms32s32( void );
#define _ms32s32 ASAP_ref(681, void (*const)( void ))


void _mu16u16( void );
#define _mu16u16 ASAP_ref(679, void (*const)( void ))
void _ms16u16( void );
#define _ms16u16 ASAP_ref(677, void (*const)( void ))


void setjmp(void *jbuf);
#define setjmp ASAP_ref(614, void (*const)(void *))
void longjmp(void *jbuf, int ret);
#define longjmp ASAP_ref(615, void (*const)(void *, int))

long labs(long x);

void *memset(void *s, int c, size_t n);
#define memset ASAP_ref(636, void *(*const)(void *, int, size_t))

void *memchr(vonst void *s, int c, size_t n);
#define memchr ASAP_ref(627, void *(*const)(vonst void *, int, size_t))

int memcmp(const void *s1, const void *s2, size_t n);
#define memcmp ASAP_ref(624, int (*const)(const void *, const void *, size_t))

void *memcpy(void *dest, const void *src, size_t n);
#define memcpy ASAP_ref(618, void *(*const)(void *, const void *, size_t))

void *memmove(void *dest, const void *src, size_t n);
#define memmove ASAP_ref(619, void *(*const)(void *, const void *, size_t))

int abs(int x);
```

Floating Point Numbers

There are two types of floating point numbers, "single" (BCD14) and "double" (BCD16) they are implemented slightly differently than usual. Both are BCD types (every nibble stores a decimal digit), storing 14dp and 16dp respectively, and an extra word for storing the (binary) exponent and sign. They both occupy 10 bytes, with the "single" using the last (spare) byte as an identifier tag (0x23) for estack operations. System variables all use the

double form, as the identifier is not required, so most float operations are done on the double type rounding to 14dp (see round14) only when necessary.

The floating point support seems to be implemented as a compiler extension. They are implemented as stack variables, operations on these variables are performed by the routine _bcd_math, which has a fairly unusual calling mechanism. Floating point registers FP0-FP7 are dynamically created on a functions stack frame as required by the calls to _bcd_math. Such that FP0 is at –10(a6), FP1 is at –20(a6), ..., FPx is at –(10*(x+1))(a6). The routines also allow you to use the return register which will resolve to –10(a6) in the previously linked stack frame.

```
//all floats occupy 10 bytes, (least significant) 2 digits are used to store the //tag
when required.
typedef struct single { /* 10 bytes */
  struct { //WORD
    BOOL sign : 1; //1=negative
    int biased_exponent : 15; //$4000 (exponent=biased_exponent-$4000)
  } ext;
  BYTE Mantissa[7]; //(packed BCD ?.??????????????, ?=4 bits=1 digit)
  BYTE Tag; /* 0x23 */
} BCD14;

typdef struct double { /* 10 bytes */
  struct { //WORD
    BOOL sign : 1; //1=negative
    int biased_exponent : 15; //$4000 (exponent=biased_exponent-$4000)
  } ext;
  BYTE Mantissa[8]; //(packed BCD ?.??????????????, ?=4 bits=1 digit)
} BCD16;

typedef double FLOAT;

enum BCD_OP
  { FCMP=0, FADD=1, FDIV=2, FMUL=3, FSUB=4, FINTRZ=5, FMOVE=6, FNEG=7, FTST=8 };
enum BCD_SIZE
  { BYTE=0, WORD=1, LONG=2, SINGLE=3, DOUBLE=4, UNSGNED=5 };
enum BCD_SOURCE
  {   FP0=0, FP1=1, FP2=2, FP3=3, FP4=4, FP5=5, FP6=6, FP7=7,
      D0=8, D1=9, D2=10, D3=11, D4=12, D5=13, D6=14, D7=15,
      IMMED_LONG=16, IMMED_SHORT=17, FRAME_OFF=18, EFFECT_ADDR=19, IMMED_ZERO };
//R0 is either FPx (stack frame register) or Dx (register) depending on size field
enum BCD_DEST
  {   R0=0, R1=1, R2=2, R3=3, R4=4, R5=5, R6=6, R7=7,
      FRAME_OFF=8, EFFECT_ADDR=9, RETURN_REG=10 };

typedef struct {
  BCD_OP Operator : 4;        //(Operator<<12)
  BCD_SIZE int Size : 3;      //|(Size<<9)
  BCD_SRC int Source : 5;     //|(Source<<8)
  BCD_DEST int Dest : 4;      //|(Dest)
} BCD_MATH;

//some constants
const FLOAT FPBIGGEST = {0x43E8, 1}; //1e1000
const FLOAT FPNEGBIGGEST = {0xC3E8, 1}; //-1e1000

const FLOAT FP_INVALID={0x7FFF, 0xAA}; //undef
const FLOAT FP_POSINFINITY={0x7FFF, 0xAA, 0, 0xBB};
const FLOAT FP_NEGINFINITY={0xFFFF, 0xAA, 0, 0xBB};
const FLOAT FP_UNSINFINITY={0x7FFF, 0xAA, 0, 0xCC};

const FLOAT FP_POSZERO={0}; //0e-0x4000
const FLOAT FP_NEGZERO={0x8000}; //-0e-0x4000
const FLOAT FP_UNSZERO={0x4000}; //0e0


//bss from bcd.o?
```

```
//...
FLOAT seed1,seed2;
//...
```

```
void _bcd_math( void )
#define _bcd_math ASAP_ref(181, void (*const)( void ))
/* Info on following words
Source sizes with bcd_math (highlighted values valid)
```

Size\Source	IMMED_SHORT	IMMED_LONG	IMMED_ZERO (0)	FRAME_OFF, EFFECT_ADDR
0=BYTE	**2**	4	0	1
1=WORD	**2**	4	0	2
2=LONG	2	**4**	0	4
3=SINGLE	2	**10**	0	9
4/5=DOUBLE/UNSGNED	2	**10**	0	10

```
Note for IMMED values, the numbers follow the BCD_MATH word (using shown bytes).
For EFFECT_ADDR the effective address of the number, passed as first optional parameter.
For FRAME_OFF is the (-ve) WORD frame offset is stored after the call (as for IMMED)

For the destination, either Rx represents FPx for SINGLE/DOUBLE/UNSGNED or Dx for
BYTE/WORD/LONG operations.
FRAME_OFF follows the bcd call and the source operand, and is a word offset.
EFFECT_ADDR is pushed as the next available paramater (i.e. 2nd, or 1st if not source EA)
*/
```

- Utility Routines
- Floating Point Maths
- Complex Float Maths
- Trigonometric Calculations

Utility Routines (number.o)

```
const FLOAT FPZERO = {0x4000}; //0.0
const FLOAT FPONE = {0x4000, 1}; //1.0
const FLOAT FPNONE = {0xC000, 1}; //-1.0
const FLOAT FPPI = {0x4000, 0x31, 0x41, 0x59, 0x26, 0x53, 0x58, 0x98};
const FLOAT FPPTFIVE = {0x3FFF, 0x50};
const FLOAT FPTWO = {0x4000, 2};
const FLOAT RAD2DEG = {0x3FFE, 0x17, 0x45, 0x32, 0x92, 0x51, 0x99, 0x44};
const FLOAT DEG2RAD = {0x4001, 0x57, 0x29, 0x57, 0x79, 0x51, 0x30, 0x82);
const FLOAT FPEXP = {0x4000, 0x27, 0x18, 0x28, 0x18, 0x38, 0x45, 0x90};
```

```
void init_float( void );
#define init_float ASAP_ref(765, void (*const)( void ))

SLONG signum_Float(const EStackIndex f);
#define signum_Float ASAP_ref(786, SLONG (*const)(const EStackIndex))
SLONG compare_Floats(EStackIndex f1, EStackIndex f2);
#define compare_Floats ASAP_ref(759, SLONG (*const)(EStackIndex, EStackIndex))
FLOAT _hypot(FLOAT x, FLOAT y);
/* return sqrt(x^2 + y^2) */
FLOAT estack_to_float(EStackIndex esi);

BOOL is_nan(FLOAT f);
#define is_nan ASAP_ref(774, BOOL (*const)(FLOAT))
BOOL is_float_infinity(FLOAT f);
#define is_float_infinity ASAP_ref(767, BOOL (*const)(FLOAT))
BOOL is_float_transfinite(FLOAT f);
#define is_float_transfinite ASAP_ref(771, BOOL (*const)(FLOAT))
BOOL is_float_signed_infinity(FLOAT f);
#define is_float_signed_infinity ASAP_ref(770, BOOL (*const)(FLOAT))
BOOL is_float_unsigned_inf_or_nan(FLOAT f);
#define is_float_unsigned_inf_or_nan ASAP_ref(772, BOOL (*const)(FLOAT))
BOOL is_float_unsigned_zero(FLOAT f);
#define is_float_unsigned_zero ASAP_ref(773, BOOL (*const)(FLOAT))
```

```
BOOL is_float_positive_zero(FLOAT f);
#define is_float_positive_zero ASAP_ref(769, BOOL (*const)(FLOAT))
BOOL is_float_negative_zero(FLOAT f);
#define is_float_negative_zero ASAP_ref(768, BOOL (*const)(FLOAT))
void push_overflow_to_infinity(TAG inf);
#define push_overflow_to_infinity ASAP_ref(782, void (*const)(TAG inf))
/* displays status line error message, and pushes inf) */

void push_pow(EStackIndex fbase, EStackIndex fexp);
#define push_pow ASAP_ref(783, void (*const)(EStackIndex, EStackIndex))
void push_Float(FLOAT f);
#define push_Float ASAP_ref(778, void (*const)(FLOAT))

static FLOAT estack_int_to_Float(EStackIndex esi);
static FLOAT estack_frac_to_Float(EStackIndex esi);
FLOAT estack_number_to_Float(EStackIndex esi);
#define estack_number_to_Float ASAP_ref(761, FLOAT (*const)(EStackIndex))

DWORD float_class(FLOAT f);
#define float_class ASAP_ref(762, DWORD (*const)(FLOAT))
/* return type of float
```

1	NAN
2	NEG_INFINIY
3	NEGATIVE
5	NEG_ZERO
6	UNS_ZERO
7	POS_ZERO
9	POSITIVE
10	POS_INFINITY
11	UNS_INFINITY

```
*/

void push_Float_to_nonneg_int(FLOAT f);
#define push_Float_to_nonneg_int ASAP_ref(779, void (*const)(FLOAT))
BOOL did_push_cnvrt_Float_to_integer(EStackIndex esi);
#define did_push_cnvrt_Float_to_integer ASAP_ref(760, BOOL (*const)(EStackIndex))
void push_cnvrt_integer_if_whole_nmb(EStackIndex esi);
#define push_cnvrt_integer_if_whole_nmb ASAP_ref(781, void (*const)(EStackIndex))
void push_Float_to_rat(EStackIndex esi);
#define push_Float_to_rat ASAP_ref(780, void (*const)(EStackIndex))

BOOL is_complex_float(EStackIndex esi); //or real float
BOOL is_matrix_all_floats?(EStackIndex esi);

FLOAT norm1_complex_Float(EStackIndex esi);
#define norm1_complex_Float ASAP_ref(777, FLOAT (*const)(EStackIndex))
SWORD compare_complex_magnitudes(EStackIndex esi1, EStackIndex esi2);
#define compare_complex_magnitudes ASAP_ref(758, SWORD (*const)(EStackIndex,
EStackIndex))
void push_round_Float(EStackIndex f);
#define push_round_Float ASAP_ref(784, void (*const)(EStackIndex))
/* convert f to rat, then back to float, in effect rounding to precision 1e-5 */

?
?
?
?

BOOL likely_approx_to_number(EStackIndex esi);
#define likely_approx_to_number ASAP_ref(776, BOOL (*const)(EStackIndex))
BOOL likely_approx_to_complex_number(EStackIndex esi);
#define likely_approx_to_complex_number ASAP_ref(775, BOOL (*const)(EStackIndex))
BOOL should_and_did_push_approx_arg2(EStackIndex arg1, EStackIndex arg2);
#define should_and_did_push_approx_arg2 ASAP_ref(785, BOOL (*const)(EStackIndex,
EStackIndex))
/* arg1 is float and arg2 approxes to float */
```

```
BOOL can_be_approxed(EStackIndex esi, BOOL complex);
#define can_be_approxed ASAP_ref(757, BOOL (*const)(EStackIndex, BOOL))
FLOAT frexp10(FLOAT f, int *exp);
#define frexp10 ASAP_ref(763, FLOAT (*const)(FLOAT, int *))
BOOL is_Float_exact_whole_number(EStackIndex esi);
#define is_Float_exact_whole_number ASAP_ref(766, BOOL (*const)(EStackIndex))
FLOAT gcd_exact_whole_Floats(EStackIndex f1, EStackIndex f2);
#define gcd_exact_whole_Floats ASAP_ref(764, FLOAT (*const)(EStackIndex, EStackIndex))

void ?(*, WORD, WORD, *);


FLOAT NEGZERO
FLOAT POSZERO
const EStackIndex FPZERO, FPONE, FPNONE, FPPTFIVE, FPTWO, FPPI, FPEXP;
FLOAT FPBIG, FPNBIG; //[+|-]9.9999999999999e999
FLOAT FP_INVALID;
```

Floating Point Maths

modf.o

```
FLOAT modf(FLOAT x, FLOAT *pint);
#define modf ASAP_ref(258, FLOAT (*const)(FLOAT, FLOAT *))
/* breaks down x into fractional part (which is returned) and integer part (stored at
pint), both having the same sign as x
*/
```

fmod.o

```
FLOAT fmod(FLOAT x, FLOAT y);
#define fmod ASAP_ref(264, FLOAT (*const)(FLOAT, FLOAT))
/* returns the floating-point remainder of x/y */
```

fmath.o

```
FLOAT floor(FLOAT x);
#define floor ASAP_ref(263, FLOAT (*const)(FLOAT))
FLOAT ceil(FLOAT x);
#define ceil ASAP_ref(261, FLOAT (*const)(FLOAT))
FLOAT fabs(FLOAT x);
#define fabs ASAP_ref(262, FLOAT (*const)(FLOAT))

FLOAT ? (FLOAT x, FLOAT y);

FLOAT acos(FLOAT x);
#define acos ASAP_ref(245, FLOAT (*const)(FLOAT))
FLOAT asin(FLOAT x);
#define asin ASAP_ref(246, FLOAT (*const)(FLOAT))

FLOAT atan2(FLOAT x, FLOAT y);
#define atan2 ASAP_ref(248, FLOAT (*const)(FLOAT, FLOAT))
FLOAT atan(FLOAT x);
#define atan ASAP_ref(247, FLOAT (*const)(FLOAT))

FLOAT cos(FLOAT x);
#define cos ASAP_ref(249, FLOAT (*const)(FLOAT))
FLOAT sin(FLOAT x);
#define sin ASAP_ref(250, FLOAT (*const)(FLOAT))
FLOAT tan(FLOAT x);
#define tan ASAP_ref(251, FLOAT (*const)(FLOAT))

FLOAT cosh(FLOAT x);
#define cosh ASAP_ref(252, FLOAT (*const)(FLOAT))
FLOAT sinh(FLOAT x);
#define sinh ASAP_ref(253, FLOAT (*const)(FLOAT))
```

```
FLOAT tanh(FLOAT x);
#define tanh ASAP_ref(254, FLOAT (*const)(FLOAT))

FLOAT exp(FLOAT x);
#define exp ASAP_ref(255, FLOAT (*const)(FLOAT))
FLOAT log(FLOAT x);
#define log ASAP_ref(256, FLOAT (*const)(FLOAT))
FLOAT log10(FLOAT x);
#define log10 ASAP_ref(257, FLOAT (*const)(FLOAT))
FLOAT pow(FLOAT x, FLOAT y);
#define pow ASAP_ref(259, FLOAT (*const)(FLOAT, FLOAT))
FLOAT negPow(FLoat x, FLOAT y);

FLOAT sqrt(FLOAT x);
#define sqrt ASAP_ref(260, FLOAT (*const)(FLOAT))

void PolToRect(FLOAT *x, FLOAT *y, FLOAT r, FLOAT theta);
?(FLOAT *, FLOAT *, FLOAT, FLOAT);
FLOAT chop15?(FLOAT x);
FLOAT ?(FLOAT, FLOAT);
?

BOOL fpisanint(BYTE *mantissa, WORD exponent);
#define fpisodd ASAP_ref(371, BOOL (*const)(BYTE *, WORD))
BOOL fpisodd(BYTE *mantissa, WORD exponent);
#define fpisanint ASAP_ref(370, BOOL (*const)(BYTE *, WORD))

FLOAT round12(FLOAT x);
#define round12 ASAP_ref(372, FLOAT (*const)(FLOAT))
FLOAT round14(FLOAT x);
#define round14 ASAP_ref(373, FLOAT (*const)(FLOAT))

?
```

Complex Float Maths (cbcdmath.o)

```
void cacos(FLOAT src, FLOAT isrc, FLOAT *dest, FLOAT *idest);
#define cacos ASAP_ref(314, void (*const)(FLOAT, FLOAT, FLOAT *, FLOAT *))
void casin(FLOAT src, FLOAT isrc, FLOAT *dest, FLOAT *idest);
#define casin ASAP_ref(315, void (*const)(FLOAT, FLOAT, FLOAT *, FLOAT *))
void casinh(FLOAT src, FLOAT isrc, FLOAT *dest, FLOAT *idest);
#define casinh ASAP_ref(318, void (*const)(FLOAT, FLOAT, FLOAT *, FLOAT *))
void catanh(FLOAT src, FLOAT isrc, FLOAT *dest, FLOAT *idest);
#define catanh ASAP_ref(319, void (*const)(FLOAT, FLOAT, FLOAT *, FLOAT *))
void cacosh(FLOAT src, FLOAT isrc, FLOAT *dest, FLOAT *idest);
#define cacosh ASAP_ref(317, void (*const)(FLOAT, FLOAT, FLOAT *, FLOAT *))
void catan(FLOAT src, FLOAT isrc, FLOAT *dest, FLOAT *idest);
#define catan ASAP_ref(316, void (*const)(FLOAT, FLOAT, FLOAT *, FLOAT *))

void csin(FLOAT src, FLOAT isrc, FLOAT *dest, FLOAT *idest);
#define csin ASAP_ref(321, void (*const)(FLOAT, FLOAT, FLOAT *, FLOAT *))
void ccos(FLOAT src, FLOAT isrc, FLOAT *dest, FLOAT *idest);
#define ccos ASAP_ref(320, void (*const)(FLOAT, FLOAT, FLOAT *, FLOAT *))
void ctan(FLOAT src, FLOAT isrc, FLOAT *dest, FLOAT *idest);
#define ctan ASAP_ref(322, void (*const)(FLOAT, FLOAT, FLOAT *, FLOAT *))
void csinh(FLOAT src, FLOAT isrc, FLOAT *dest, FLOAT *idest);
#define csinh ASAP_ref(324, void (*const)(FLOAT, FLOAT, FLOAT *, FLOAT *))
void ccosh(FLOAT src, FLOAT isrc, FLOAT *dest, FLOAT *idest);
#define ccosh ASAP_ref(323, void (*const)(FLOAT, FLOAT, FLOAT *, FLOAT *))
void ctanh(FLOAT src, FLOAT isrc, FLOAT *dest, FLOAT *idest);
#define ctanh ASAP_ref(325, void (*const)(FLOAT, FLOAT, FLOAT *, FLOAT *))

void csqrt(FLOAT src, FLOAT isrc, FLOAT *dest, FLOAT *idest);
#define csqrt ASAP_ref(326, void (*const)(FLOAT, FLOAT, FLOAT *, FLOAT *))
void cln(FLOAT src, FLOAT isrc, FLOAT *dest, FLOAT *idest);
#define cln ASAP_ref(327, void (*const)(FLOAT, FLOAT, FLOAT *, FLOAT *))
```

```
void clog10(FLOAT src, FLOAT isrc, FLOAT *dest, FLOAT *idest);
#define clog10 ASAP_ref(328, void (*const)(FLOAT, FLOAT, FLOAT *, FLOAT *))
void cexp(FLOAT src, FLOAT isrc, FLOAT *dest, FLOAT *idest);
#define cexp ASAP_ref(329, void (*const)(FLOAT, FLOAT, FLOAT *, FLOAT *))
```

Trigonometric Calculations

- Standard Trig Functions
- Inverse Trig Functions

Standard Trig Functions (trig.o)

```
void sincos(FLOAT x, BOOL Deg, FLOAT *sin, FLOAT *cos);
#define sincos ASAP_ref(646, void (*const)(FLOAT, BOOL, FLOAT *, FLOAT *))
FLOAT asinh(FLOAT x);
#define asinh ASAP_ref(647, FLOAT (*const)(FLOAT))
FLOAT acosh(FLOAT x);
#define acosh ASAP_ref(648, FLOAT (*const)(FLOAT))
FLOAT atanh(FLOAT x);
#define atanh ASAP_ref(649, FLOAT (*const)(FLOAT))

void trig(int op, BOOL Deg, const FLOAT *px, FLOAT *sin, FLOAT *cos, FLOAT *tan);
#define trig ASAP_ref(651, void (*const)(int, BOOL, const FLOAT *, FLOAT *, FLOAT *,
FLOAT *))

void hyptrig(int op, const FLOAT *px, FLOAT *sin, FLOAT *cos, FLOAT *tan);
static void _hyptrig(const FLOAT *px, FLOAT *sin, FLOAT *cos);
```

Inverse Trig Functions (trigi.o)

```
void itrig(int op, BOOL Deg, FLOAT *px, FLOAT *dest);
#define itrig ASAP_ref(650, void (*const)(int, BOOL, FLOAT *, FLOAT *))

void _itan2(BOOL Deg, FLOAT *x, FLOAT *y, FLOAT *dest);
/* used by atan2, and itrig (with y set to FPONE) */

void ihyptrig(int op, FLOAT *px, FLOAT *dest);

void ihyptan(int op, FLOAT *x, FLOAT *y, FLOAT *dest);

?test?
?log?
```

Applications

- Home
- Y= Editor
- Window
- Graph
- Table
- Data/Matrix Editor
- Geometry
- Text
- Numeric Solver
- Self-Test

Home

- Algebra Application

- Home Stack

Algebra Application (apalgebra.o)

```
static WORD SymDelAToZ( void );
/* returns how many Syms remaining before error ... i.e. 0 if OK */

void HomeInit( void );

static HANDLE UnlockNode(HANDLE h);

static void ShowStackInfo( void );

void AP_algebra(EVENT *event);

void HomeClearAToZ( void );
/* Handle command, show dialogs etc. */

static void authHEnableFkeys( void );

static void homeExec( void );
/* handle enter key in home author */

static void homeGraphErr( void );
/* error passed from Graph screen, bad function  */

static void homeFormat( void );
/* Format menu option */

static void authH_handleEvent(EVENT *event);

static void kbdprgmExec( void );

static void authH_paint( void );

static void AnsOut( void );
/* Send the current result out the link port, via OSAlexOut */

?
?

static void authHtoStack( void );
static void authHtoPrgmIO( void );

void Homeclear( void );
/* Menu option, clear all nodes, switch to auth */

void HS_copycur( void );
/* copy an entry/ans */

static void GetCurAns(HANDLE *hNode, FIFO_ELEM *Ans);
/* Get the node and ans element
Ans==NULL if unable to get node, i.e. if ER_MSG_TAG
*/

static void DeleteCurNode( void );

?

static void HS_enableFKeys( void );

static void HS_handleEvent(EVENT *event);

?
?

static void HS_paint( void );

static void DrawFIFIElement(FIFO_ELEM *elem, BOOL torf);
```

```
static void authPut( void );

static BOOL initFIFONode(HANDLE hNode);

static BOOL positionNodes(WORD Width);
/* repositions all FIFO nodes to display */

static void ClearBuffer( void );
/* clear current text line */

void Homesaveas( void );

static void ElemScrollDown( void );
static void ElemRedraw( void );
/* Redraw element from Buffer, or Parse2DExpr on the estack */

static void ElemScrollLeft( void );
static void ElemToLeft( void );
static void ElemScrollRight( void );
static void ElemToRight( void );
static void ElemScrollUp( void );

?

void HS_redraw_after_add( void );

WORD GetCurNodeNum( void );

void StackToHauth( void );

to prgmio

void ClearHome( void );

void PrgmIO_PrintExpr(EStackIndex esi);
void PrgmIO_EnableFKeys(BOOL );
void PrgmIO_handleEvent( void );

void PrgmIOToHome( void );

void cmds_input(EStackIndex esi, Bool str);
/* input, inputstr (note input with specified paramer requests string) */

void InputEventHandler(EVENT *event);

void Input(EStackIndex Prompt, EStackIndex var, BOOL str);

void homeToPrgmIO( void );
void switchToPrgmIO(BOOL);

toHauth
toStack

void PrgmIO_ShiftUp( void );

HomeExecute
#define HomeExecute ASAP_ref(270, ASAP_UNDEF)
HomePushEStack
#define HomePushEStack ASAP_ref(271, ASAP_UNDEF)
void cmd_clrhome( void );
#define cmd_clrhome ASAP_ref(816, void (*const)( void ))
void cmd_clrio( void );
#define cmd_clrio ASAP_ref(817, void (*const)( void ))
void cmd_disp(EStackIndex esi);
#define cmd_disp ASAP_ref(829, void (*const)(EStackIndex))
void cmd_disphome( void );
#define cmd_disphome ASAP_ref(831, void (*const)( void ))
void cmd_input(EStackIndex esi);
```

```
#define cmd_input ASAP_ref(854, void (*const)(EStackIndex))
void cmd_inputstr(EStackIndex esi);
#define cmd_inputstr ASAP_ref(855, void (*const)(EStackIndex))
void cmd_newprob( void );
#define cmd_newprob ASAP_ref(871, void (*const)( void ))
void cmd_output(EStackIndex row, EStackIndex column, EStackIndex exprOrString);
#define cmd_output ASAP_ref(873, void (*const)(EStackIndex, EStackIndex, EStackIndex))
void cmd_pause(EStackIndex esi);
#define cmd_pause ASAP_ref(875, void (*const)(EStackIndex))
void cmd_prompt(EStackIndex esi);
#define cmd_prompt ASAP_ref(881, void (*const)(EStackIndex))

static TEXT_EDIT PrgmIO_textEdit;
static TEXT_EDIT Author_textEdit;
static BOOL PrgmIO_bFKeysSet;
static BOOL focus;
static WORD Mode; /* Author, Stack, PrgmIO */
static WORD Flags;
static HANDLE hTextBuffer;
static WIN_RECT StackRect;
static WORD cursel;
static DWORD Height;

static WINDOW wAuthor;
static WINDOW wHomeStack;
static WINDOW wPrgmIO;

//Resources
MENU HomeMenu;
DIALOG HomeFormatDlg;
DIALOG HomeClearAToZDlg;
DIALOG UnableDeleteWarnDlg;
```

Home Stack (FIFO Nodes) (homestack.o)

Home screen stores a single FIFO_NODE per home screen line. The left node is built from a call to NG_tokenise on the TEXT editor line,
the right node is built from a call to NG_execute on the tokenised text.

They are both displayed by a call to Parse2DMultiExpr, and then Print2DExpr if bPretty on otherwise just WinCharXY.
These nodes are maintained in a doubly linked list (of handles) containing information for the left and right nodes of a home
screen line. This information included the handle of the expression, position,
and some mode information.

```
struct FIFOElement {
  WORD x;
  DWORD y;
  DWORD StartX;
  WORD Width;
  WORD Height;
  WORD yCentre;
  HANDLE hTok;
  BOOL bTooLong;
  BOOL bPretty;
  WORD ExpForm;
  WORD Fix;
} FIFO_ELEM;

struct FIFOStruct {
  FIFO_ELEM Entry;
  FIFO_ELEM Ans;
  HANDLE hPrev;
  HANDLE hNext;
} FIFO_NODE;
```

```
void HS_chopFIFO( void );
#define HS_chopFIFO ASAP_ref(572, void (*const)( void ))
WORD HS_countFIFO( void );
#define HS_countFIFO ASAP_ref(573, WORD (*const)( void ))
HANDLE HS_deleteFIFONode(HANDLE node);
#define HS_deleteFIFONode ASAP_ref(574, HANDLE (*const)(HANDLE))
/* returns handle of next node */

void HS_freeAll( void );
#define HS_freeAll ASAP_ref(575, void (*const)( void ))

void HS_freeFIFONode(HANDLE node);
#define HS_freeFIFONode ASAP_ref(576, void (*const)(HANDLE))

//(entry on left, ans on right of screen)
HANDLE HS_getAns(WORD num);
#define HS_getAns ASAP_ref(577, HANDLE (*const)(WORD))
/* returns H_NULL if ERROR_MSG_TAG */

HANDLE HS_getEntry(WORD num);
#define HS_getEntry ASAP_ref(578, HANDLE (*const)(WORD))

HANDLE HS_getFIFONode(WORD num);
#define HS_getFIFONode ASAP_ref(579, HANDLE (*const)(WORD))
/* Get the actual node strucutre */

HANDLE HS_newFIFONode( void );
#define HS_newFIFONode ASAP_ref(581, HANDLE (*const)( void ))

HANDLE HS_popEStack( void );
#define HS_popEStack ASAP_ref(580, HANDLE (*const)( void ))
/* Pop expression off estack and copy it into a handle
This Handle is allocated with HeapAllocHigh, and is therefore inteded to be temporary
*/

void HS_pushFIFONode(HANDLE node);
#define HS_pushFIFONode ASAP_ref(582, void (*const)(HANDLE))

EStackIndex HToESI(HANDLE h);
#define HToESI ASAP_ref(583, EStackIndex (*const)(HANDLE))
/* Return EStackIndex of top of handle for interpreting
Simply dereferences handle and adds size to the address
*/

void HS_Init( void );
/* Calls:
  ER_Init, GL_Init, WinInit, ST_Init, MO_Init, CU_Init, push_Init, estack_Init,
  CB_Init, EM_Init, PP_Init, SymInit, SeedInit
*/

HANDLE HS_hFirstFIFONode;
WORD HS_maxFIFONode;
```

Y= Editor (apequed.o)

```
void AP_equ(EVENT *event);


EQU_deStatus
#define EQU_deStatus ASAP_ref(366, ASAP_UNDEF)
EQU_getNameInfo
#define EQU_getNameInfo ASAP_ref(288, ASAP_UNDEF)
EQU_select
#define EQU_select ASAP_ref(286, ASAP_UNDEF)
EQU_setStyle
```

```
#define EQU_setStyle ASAP_ref(287, ASAP_UNDEF)

static BOOL bHighlight
static WIN_RECT rHighlight
static TEXT_EDIT textEdit;
static WINDOW wAuthor;

//Resources:
MENU EquMenu;
DIALOG EquSeqAxesDlg;
DIALOG EquDEAxesDlg;
```

Window (apwinded.o)

```
//...

MENU WindowMenu;
```

Graph

- Application and Variables
- Graphing
- 3D Graphs
- GDB Variables
- Plots
- Drawing Menu
- Maths Menu
- Graph Tools Menu
- Graph Zoom Menu

Application and Variables (aptigraph.o)

```
#ifdef PLUS
#define MAXVARS 12
#define MODES 6
#else
#define MAXVARS 11
#define MODES 5
#endif

// Range structures:

typedef struct  {
  FLOAT xmin,xmax;
  union {
    FLOAT xscl;
    FLOAT xgrid; //3D
  } x;
  FLOAT ymin,ymax;
  union {
    FLOAT yscl;
    FLOAT ygrid; //3D
  } y;
  FLOAT _DELTA_x,_DELTA_y;
} GR2D_RNG;

typedef struct {
  GR2D_VARS box;
  FLOAT xres;
} GRFUNC_RNG;

typedef struct {
  GR2D_VARS box;
  FLOAT tmin,tmax,tstep;
```

```
} GRPARM_RNG;

typedef struct {
  GR2D_VARS box;
  FLOAT _THETA_min,_THETA_max,_THETA_step;
} GRPOL_RNG;


typedef struct {
  GR2D_VARS box;
  FLOAT nmin,nmax;
  FLOAT plotStrt,plotStep;
} GRSEQ_RNG;


typedef struct {
  GR2D_VARS box; //deltas used but not accessible
  FLOAT zmin,zmax,zscl;
  FLOAT eye_THETA_,eye_PHI_;
#ifdef PLUS
  FLOAT eye_PSI_;
  FLOAT ncontour;
  FLOAT scale[3];
#else
  FLOAT scale[2]; //not named
#endif
} GR3D_RNG;

#ifdef PLUS
typedef struct {
  GR2D_VARS box; //deltas just used for padding
  FLOAT t0,tmax,tstep,tplot;
  FLOAT diftol;
  FLOAT Estep;
  FLOAT fldres;
  FLOAT ncurves;
  FLOAT dtime;
} GRDE_RNG;
#endif

typedef union {
  GRFUNC_RNG *func;
  GRPARM_RNG *parm;
  GRPOL_RNG *pol;
  GRSEQ_RNG *seq;
  GR3D_RNG *gr3;
  GRDE_RNG *de;
} PTR_ALL_RNG_VARS;
```

```
typedef struct GraphFormatStruct{
  WORD normal; //AXES only available in SEQ & some DE modes, all AXES bits 0 otherwise
    //.0 : GF.Coordinates.ON=POLAR:RECT; //(1:0)
    //.1 : GF.Leading Cursor=ON:OFF;
    //.2 : GF.Labels=ON:OFF;
    //.3 : GF.Axes.ON=BOX:AXES;
    //.4 : GF.Axes=OFF:ON(.3)
    //.5 : GF.Grid=ON:OFF
    //.6 : GF.Graph Order=SIMUL:SEQ
    //.7 : GF.Coordinates=OFF:ON(.0)
    //.8 : PLUS?zoom_flag :GF.Style=WIRE FRAME:HIDDEN SURFACE;
    //.11 : GR3.QuickZoom;
    //.13 : AXES.Build Web=AUTO:TRACE;
    //.14 : AXES.Axes.OTHER=WEB:CUSTOM;
    //.15 : AXES.Axes=TIME:OTHER;
  BYTE AXES.X Axis; //Seq: ((x==-1)?"n":form("u%d",x)), DE:
((x==0)?"t":(x==100)?"y":(x==-100)?"y'"?(x>0)>form("y%d",x):form("y%d'",-x))
  BYTE AXES.Y Axis; //"
#ifdef PLUS
```

```
  WORD deflags;
    //.0: GF.Solution Method=EULER:RK;
    //.1: GF.Fields.!FLDOFF=DIRFLD:SLPFLD; //=.4
    //.2: GF.Fields=(.1):FLDOFF;
  BYTE GF.Graph Style; //3D Graph Style 1..5 -> 0..4
  BYTE 0; //space
#endif
} GR_FORMAT;
```

```
//3D Graphing database on PLUS
typedef struct {
  WORD w1; /* 0 LONG?*/
  WORD w2; /* 2 */

  HANDLE _h1; /* 4 */
  LONG Max1;
  HANDLE _h2; /* 0xA */

  HANDLE _hMain; /* 0xC */
  LONG Max2; /* 0xE */
  HANDLE h4; /* 0x12 */

  LONG Max3; /* 0x14 */
  HANDLE hRedraw; /* 0x18 */

  WORD CurMat[3][3];    /* 1A The matrix that is finally used in the output */
  WORD LastMat[3][3];   /* 2C - stored on first kepress (for spinning) */
  WORD InitialMat[3][3];      /* 3E - pressing '0' restores this matrix */
  WORD scale; //50
  WORD last_dir_key; //52
  BYTE step; /* 54 - spinning angle step ('+' and '-') 1..3
  BYTE key_lock; //55
  WORD factor1; //56
  WORD factor2; //58
} GR3_DB;
```

```
typedef struct GraphVarsStruct {
  FLOAT xc,yc,zc,tc,rc,_THETA_c,nc;
  FLOAT tmp_DELTA_x,tmp_DELTA_y,tmp_xmin,tmp_xmax,FPZERO;
  FLOAT tblStart_entered_value, lines away from tblStart;
  FLOAT tblStart,_DELTA_tbl; //this tblStart changes when scrolling through tblInput
//A0, A0
  Graph_vars_all *cur_mode_vars;
  FLOAT ZoomPrev[MAXVARS]=copy_of_previous_defined_vars; //in order (cleared on mode
change)
  FLOAT zxmin,zxmax,zxscl; //ZoomSto settings (z*) (transferrable across modes)
  FLOAT zymin,zymax,zyscl;
  FLOAT zxres;
  FLOAT ztmin,ztmax,ztstep;
  FLOAT z_THETA_min,z_THETA_max,z_THETA_step;
  FLOAT znmin,znmax;
  FLOAT zpltstrt,zpltstep;
  FLOAT zzmin,zzmax,zzscl;
  FLOAT zeye_THETA_,zeye_PHI_;
#ifdef PLUS
  FLOAT zeye_PSI_;
#endif
  FLOAT zxgrid,zygrid
#ifdef PLUS
  FLOAT zt0de,ztmaxde,ztstepde,ztplotde;
#endif
//202, 23E
  GR_FORMAT *cur_mode_flags;
  WINDOW *wGraph;
  WINDOW *wTable;
  WINDOW *wWindow;
```

```
  TableWinVars *table_win_vars; //a 0x26 byte structure in .bss of aptabled.o (2
sequential)
//216, 252
  WINDOW Yeq;
//240, 27C
  WORD Yeq_Flags;
    //entering eqn
  WORD Yeq_FuncSel;
  WORD Yeq_YPosFuncSel;
  WORD Yeq_ScrollX; /* =0 */
  WORD Yeq_curY;
  WORD Yeq_FuncSubType;
  WORD Graph_plotXpos;
  WORD Yeq_flags; /* ? */
  WORD zero;
  WORD Graph_height; /* ? */
  WORD TABLEFORMATS.Cell_Widths;
  HANDLE hGR3Calc;
//258, 294
#ifdef PLUS
  GR3_DB GR3_Data; /* (5A) */
#endif
//258, 2EE
  HANDLE tblInput;  //2EE
#ifdef PLUS
  HANDLE fldpic; //2F0
#endif
  WORD flags; //2F2
   //.0-5: GRAPH.flags & tblset.redraw?
   //.6&.7 entered data;
   //.8 complete
  BYTE Graph_width; // 2F4
  BYTE Graph_height; //2F5
  BYTE SysGraphRef; //2F6
  BYTE MODE_Graph //2F7
  BYTE GRAPH_current_active; //2F8 if 2
  BYTE solver_graph_count; //2F9
  BYTE graph_unknown; //2FA
  BYTE WINDOW_Width; //2FB
  BYTE WINDOW_Height; //2FC
  BYTE tblset_flags; //2FD
   //6 : Independent: ASK:AUTO
   //7 : Graph <-> Table: ON:OFF
  BYTE DE_flags; //2FE ?
  BYTE graph_flags; //2FF ?
   //0: cleared on entering
  BYTE trace_flags; //300
    //.2 func
  BYTE trace_SLPFLD?; //301
  BYTE trace_funcsel;; //302 (x==-1)?none:(x<100)?form("%d",x):form("P%d",x-100) //Note: dd for func, Pdd for data
  BYTE trace_seq_DE_funcsel; //303 (funcsel=WEB trace.funcsel)
  WORD trace_prevdir; //304 -1 for up, 0 for down
  WORD trace_plots_pointsel; //306
  WORD trace_point_flags; //308
    //.8 notfirst
  WORD ZoomPrev_Mode; //30A mode which Recalled vars
//272, 30c
} GR_VARS;
```

```
const char * const = {
      0, Int_f_x, Diff_y_x, Diff_r_theta, Diff_y_t, Diff_x_t, Distance, Arc,
      0, Zero, Minimum, Maximum, Inflection, Intersection
};

WORD StatsDef[7];
WORD StatsFunc[6];
WORD StatsPol[11];
WORD StatsSeq[18];
```

```
WORD StatsGR3[31];
WORD StatsDE[18];

const WORD * const ModeStats = { StatsFunc, StatsPol, StatsSeq, StatsGR3, StatsDE };
const char * const df = { DrawFunc, DrawInv, DrawPol, DrawParm, DrawSlp, 0, DrawCtour };
const char * yeq = "y=";
const char * x = "x";
const char * eq = "=";
```

```
void AP_graph(EVENT *event);
```

```
void init_gr_vars( void );
```

```
void SetGraphMode(BYTE GraphMode);
#define SetGraphMode ASAP_ref(437, void (*const)(BYTE))
```

```
?
?
?
?
```

```
void GrAxes(WORD, GR_VARS *gv);
#define GrAxes ASAP_ref(439, void (*const)(WORD, GR_VARS *))
```

```
FLOAT XCvtPtoF(WORD pt, GR_VARS *gv);
#define XCvtPtoF ASAP_ref(442, FLOAT (*const)(WORD, GR_VARS *))
/* convert Pos to Float (X) */
FLOAT CptFuncX(FLOAT f, GR_VARS *gv);
#define CptFuncX ASAP_ref(441, FLOAT (*const)(FLOAT, GR_VARS *))
```

```
FLOAT YCvtPtoF(WORD pt, GR_VARS *gv);
#define YCvtPtoF ASAP_ref(443, FLOAT (*const)(WORD, GR_VARS *))
/* Convert point to Float position */
```

```
WORD YCvtFtoP(FLOAT f, GR_VARS *gv);
#define YCvtFtoP ASAP_ref(444, WORD (*const)(FLOAT, GR_VARS *))
WORD XCvtFtoP(FLOAT f, GR_VARS *gv);
#define XCvtFtoP ASAP_ref(445, WORD (*const)(FLOAT, GR_VARS *))
```

```
?
?
?
?
?
```

```
GrClipLine
#define GrClipLine ASAP_ref(448, ASAP_UNDEF)
FuncLineFlt
#define FuncLineFlt ASAP_ref(447, ASAP_UNDEF)
GrLineFlt
#define GrLineFlt ASAP_ref(446, ASAP_UNDEF)
CptDeltax
#define CptDeltax ASAP_ref(449, ASAP_UNDEF)
CptDeltay
#define CptDeltay ASAP_ref(450, ASAP_UNDEF)
CkValidDelta
#define CkValidDelta ASAP_ref(451, ASAP_UNDEF)
```

```
void gr_DelFolder(GRAPH_VARS *gv);
#define gr_DelFolder ASAP_ref(460, void (*const)(GRAPH_VARS *))
/* delete the graph folder opened with gr_openFolder below */
```

```
void gr_openFolder(GRAPH_VARS *gv);
#define gr_openFolder ASAP_ref(461, void (*const)(GRAPH_VARS *))
/* adds the graph temporary folder, named "@" */
```

```
DWORD rngLen(BYTE GraphMode);
#define rngLen ASAP_ref(479, DWORD (*const)(BYTE))
/* returns the length of the range vars in GraphMode, e.g. 9*10 for Func */
```

```
HANDLE CreateEmptyList( void );
#define CreateEmptyList ASAP_ref(486, HANDLE (*const)( void ))
BOOL ck_valid_float( FLOAT f );
#define ck_valid_float ASAP_ref(485, BOOL (*const)( FLOAT f ))
void gr_delete_fldpic(GRAPH_VARS *gv);
#define gr_delete_fldpic ASAP_ref(500, void (*const)(GRAPH_VARS *))
void gr_remove_fldpic(GRAPH_VARS *gv);
#define gr_remove_fldpic ASAP_ref(501, void (*const)(GRAPH_VARS *))
void gr_add_fldpic( void );
#define gr_add_fldpic ASAP_ref(502, void (*const)( void ))
void gr_ck_solvergraph(GRAPH_VARS *gv);
#define gr_ck_solvergraph ASAP_ref(506, void (*const)(GRAPH_VARS *))

//map of exports from apgraph.o
//($5DDA,$60BA,$6552)
WORD gr_cur_mode;
WORD gr_flags;
  //.0 drawing
  //.7 solver

//pointers are just swapped on change:
GR_VARS *gr_active = GR1;
#define gr_active ASAP_ref(267, GR_VARS* (*const))
GR_VARS *gr_other = GR2;
#define gr_other ASAP_ref(268, GR_VARS* (*const))
WINDOW wGraph1,wGraph2;
WINDOW wWindow1,wWindow2;
WINDOW wTable1,wTable2;

PTR_ALL_RNG_VARS CurGraphModeVars[MODES];
GR_FORMAT *CurGraphFormat[MODES];

GRFUNC_RNG Func1, Func2;
GRPARM_RNG Parm1, Parm2;
GRPOL_RNG Pol1, Pol2;
GRSEQ_RNG Seq1, Seq2;
GR3D_RNG 3D1, 3D2;
#ifdef PLUS
GRDE_RNG Diff1, Diff2;
#endif

GR_FORMAT Func1Formats, Func2Formats;
GR_FORMAT Parm1Formats, Parm2Formats;
GR_FORMAT Pol1Formats, Pol2Formats;
GR_FORMAT Seq1Formats, Seq2Formats;
GR_FORMAT 3D1Formats, 3D2Formats;

#ifdef PLUS
GR_FORMAT DE1Formats, DE2Formats;
#endif

GR_VARS GR1, GR2;

FLOAT xfact,yfact,zfact;
FLOAT sysMath;

BYTE StatFlags?; //8
HANDLE hStatVars;
```

Dynamically allocated statvars

```
//FP_INVALID if undefined
struct StatVars {
  FLOAT x_BAR_,y_BAR_;
  FLOAT _SIGMA_X,_SIGMA_x^2;
  FLOAT _SIGMA_y,_SIGMA_y^2;
  FLOAT _SIGMA_xy;
  FLOAT Sx,Sy;
```

```
  FLOAT _sigma_x,_sigma_y;
  FLOAT nStat;
  FLOAT minX,minY;
  FLOAT q1,medStat,q3;
  FLOAT maxX,maxY;
  FLOAT corr;
  FLOAT R^2;
  FLOAT medx1,medx2,medx3;
  FLOAT medy1,medy2,medy3;
};

//Resources
MENU GraphMenu;
DIALOG ZoomFactorsDlg;
DIALOG GraphFormatsDlg;
DIALOG R3FormatsDlg;
DIALOG GRDEFormatsDlg;
DIALOG InitCondsDlg;
```

Graphing (regraph.o)

```
Regraph
#define Regraph ASAP_ref(438, ASAP_UNDEF)
FindGrFunc
#define FindGrFunc ASAP_ref(454, ASAP_UNDEF)
FindFunc
#define FindFunc ASAP_ref(453, ASAP_UNDEF)
grFuncName
#define grFuncName ASAP_ref(455, ASAP_UNDEF)
gr_initCondName
#define gr_initCondName ASAP_ref(456, ASAP_UNDEF)

?

execute_graph_func
#define execute_graph_func ASAP_ref(464, ASAP_UNDEF)
cpt_gr_fun
#define cpt_gr_fun ASAP_ref(465, ASAP_UNDEF)
cpt_gr_param
#define cpt_gr_param ASAP_ref(466, ASAP_UNDEF)
cpt_gr_polar
#define cpt_gr_polar ASAP_ref(467, ASAP_UNDEF)
setup_more_graph_fun
#define setup_more_graph_fun ASAP_ref(462, ASAP_UNDEF)
unlock_more_graph_fun
#define unlock_more_graph_fun ASAP_ref(463, ASAP_UNDEF)
gr_del_locals
#define gr_del_locals ASAP_ref(459, ASAP_UNDEF)

?

seqStepCk
#define seqStepCk ASAP_ref(478, ASAP_UNDEF)
StepCk
#define StepCk ASAP_ref(477, ASAP_UNDEF)
deStepCk
#define deStepCk ASAP_ref(505, ASAP_UNDEF)
CptIndep
#define CptIndep ASAP_ref(457, ASAP_UNDEF)
gr_CptIndepInc
#define gr_CptIndepInc ASAP_ref(458, ASAP_UNDEF)

?
?
?
?
?
```

```
?
?

gr_xres_pixel
#define gr_xres_pixel ASAP_ref(440, ASAP_UNDEF)

?
?

GR_Pan
#define GR_Pan ASAP_ref(452, ASAP_UNDEF)
gr_execute_seq
#define gr_execute_seq ASAP_ref(468, ASAP_UNDEF)
time_loop
#define time_loop ASAP_ref(472, ASAP_UNDEF)

?

CountGrFunc
#define CountGrFunc ASAP_ref(469, ASAP_UNDEF)
InitTimeSeq
#define InitTimeSeq ASAP_ref(473, ASAP_UNDEF)

?

FirstSeqPlot
#define FirstSeqPlot ASAP_ref(470, ASAP_UNDEF)
cleanup_seq_mem
#define cleanup_seq_mem ASAP_ref(471, ASAP_UNDEF)
gr_seq_value
#define gr_seq_value ASAP_ref(476, ASAP_UNDEF)
run_one_seq
#define run_one_seq ASAP_ref(475, ASAP_UNDEF)
seqWebInit
#define seqWebInit ASAP_ref(474, ASAP_UNDEF)

?
?

gr_DispLabels
#define gr_DispLabels ASAP_ref(483, ASAP_UNDEF)

?

gr_de_axes_lbl
#define gr_de_axes_lbl ASAP_ref(498, ASAP_UNDEF)
gr_del_vars_in_folder
#define gr_del_vars_in_folder ASAP_ref(497, ASAP_UNDEF)
gr_find_de_result
#define gr_find_de_result ASAP_ref(488, ASAP_UNDEF)
gr_execute_de
#define gr_execute_de ASAP_ref(499, ASAP_UNDEF)
de_rng_no_graph
#define de_rng_no_graph ASAP_ref(674, ASAP_UNDEF)
InitDEAxesRng
#define InitDEAxesRng ASAP_ref(489, ASAP_UNDEF)
InitDEMem
#define InitDEMem ASAP_ref(490, ASAP_UNDEF)
de_initRes
#define de_initRes ASAP_ref(496, ASAP_UNDEF)

?

de_loop
#define de_loop ASAP_ref(491, ASAP_UNDEF)

?

cleanup_de_mem
```

```
#define cleanup_de_mem ASAP_ref(492, ASAP_UNDEF)
gr_de_value
#define gr_de_value ASAP_ref(493, ASAP_UNDEF)

?

gr_find_el
#define gr_find_el ASAP_ref(504, ASAP_UNDEF)

?

CptLastIndepDE
#define CptLastIndepDE ASAP_ref(495, ASAP_UNDEF)
gr_find_func_index
#define gr_find_func_index ASAP_ref(494, ASAP_UNDEF)

?
?
?
?
?

FLOAT zFit[4];
WORD seqmode;
BYTE seqsel?;
#ifdef PLUS
HANDLE deMem?;
#endif
```

3D Graphing (graph3d.o)

```
#ifdef PLUS
FLOAT FP_GR3MUL = 0x4000; /* {0x4004, 0x1638400000000000 } */
WORD StepSettings[3][3];
FLOAT FP_GR3SCALE = 0.577350269190;

GR3_addContours
#define GR3_addContours ASAP_ref(507, ASAP_UNDEF)
GR3_freeDB
#define GR3_freeDB ASAP_ref(509, ASAP_UNDEF)
GR3_handleEvent
#define GR3_handleEvent ASAP_ref(510, ASAP_UNDEF)

#endif

GR3_paint3d
#define GR3_paint3d ASAP_ref(511, ASAP_UNDEF)
GR3_xyToWindow
#define GR3_xyToWindow ASAP_ref(512, ASAP_UNDEF)

#ifndef PLUS
addPointToPoly
addTripleToPoly
#endif

apply(*,*,*)

void cmd_drwctour(EStackIndex esi);
```

GDB Variables (l???.o)

```
static const KEYCODE gr_keys[9] = {
  KB_UP, KB_DOWN, KBM_2ND|KB_UP, KBM_2ND|KB_DOWN
  KB_LEFT, KB_RIGT, KBM_2ND|KB_LEFT, KBM_2ND|KB_RIGHT, 0
};

void cmd_clrdraw( void );
```

```
#define cmd_clrdraw ASAP_ref(813, void (*const)( void ))
void cmd_clrgraph( void );
#define cmd_clrgraph ASAP_ref(815, void (*const)( void ))
void cmd_clrtable( void );
#define cmd_clrtable ASAP_ref(818, void (*const)( void ))
void cmd_dispg( void );
#define cmd_dispg ASAP_ref(830, void (*const)( void ))
void cmd_disptbl( void );
#define cmd_disptbl ASAP_ref(832, void (*const)( void ))
void cmd_fnoff(EStackIndex esi);
#define cmd_fnoff ASAP_ref(845, void (*const)(EStackIndex))
void cmd_fnon(EStackIndex esi);
#define cmd_fnon ASAP_ref(846, void (*const)(EStackIndex))
void cmd_graph(EStackIndex esi);
#define cmd_graph ASAP_ref(851, void (*const)(EStackIndex))
void cmd_rclgdb(EStackIndex GDBvar);
#define cmd_rclgdb ASAP_ref(897, void (*const)(EStackIndex))
void cmd_stogdb(EStackIndex GDBvar);
#define cmd_stogdb ASAP_ref(912, void (*const)(EStackIndex))
void GraphOrTableCmd(EStackIndex expr, BOOL liberal);
#define GraphOrTableCmd ASAP_ref(484, void (*const)(EStackIndex, BOOL))
/* Add function z1 to graph folder ('@') formed from expr, which is stored as a function
dependent on current graphing paramaters.
BOOL liberal is actually a BYTE, and stops errors being thrown on many events */

DWORD gdb_len( void );
#define gdb_len ASAP_ref(480, DWORD (*const)( void ))

// static funcs for gdb_len

void gdb_store(EStackIndex dest);
#define gdb_store ASAP_ref(481, void (*const)(EStackIndex))

// static funcs for gdb_store

void gdb_recall(HSYM var);
#define gdb_recall ASAP_ref(482, void (*const)(HSYM))

// static funcs for gdb_recall

packed struct GDB_VAR {
  BYTE MODE.Number_Of_Graphs;
  BYTE MODE.Angle;
  BYTE MODE.Complex_Format;
  BYTE MODE.Graph;
  if(MODE.Graph==2) {
    BYTE Graph # Active; //0..1
    BYTE MODE.Graph_2;
    BYTE MODE.Split_Screen;
    BYTE MODE.Split_Screen_Ratio;
  }

  packed struct GDB_Single {
    (*CurGraphModeVars);

    GR_FORMAT format;

    BYTE NumFuncs;
    packed struct {
      BYTE num;
      STORE FUNC function;
    } Functions to plot [NumFuncs];
    BYTE NumConsts; //only seen with DEs
    packed struct {
      BYTE num;
      STORE EXPR const;
    } Constants to plot [NumConst];
    if(MODE.Graph!=3D) {
      packed struct TblSet {
```

```
        BYTE Table Setup flags; //TblSet
        FLOAT tblStart;
        FLOAT _DELTA_ tbl;
        STORE LIST tblInput;
      }
    } TblSet;
  } Graph_data[MODE.Number_Of_Graphs];
  token $DE;
};
```

Plots (gstat.o)

```
GS_PlotTrace
#define GS_PlotTrace ASAP_ref(513, ASAP_UNDEF)
GZ_Stat
#define GZ_Stat ASAP_ref(566, ASAP_UNDEF)
PlotInit
#define PlotInit ASAP_ref(518, ASAP_UNDEF)
PlotSize
#define PlotSize ASAP_ref(520, ASAP_UNDEF)
PlotDup
#define PlotDup ASAP_ref(519, ASAP_UNDEF)
PlotPut
#define PlotPut ASAP_ref(516, ASAP_UNDEF)
PlotGet
#define PlotGet ASAP_ref(517, ASAP_UNDEF)
QActivePlots
#define QActivePlots ASAP_ref(522, ASAP_UNDEF)
PlotDel
#define PlotDel ASAP_ref(515, ASAP_UNDEF)
PlotLookup
#define PlotLookup ASAP_ref(521, ASAP_UNDEF)
GS_PlotAll
#define GS_PlotAll ASAP_ref(514, ASAP_UNDEF)
QPlotActive
#define QPlotActive ASAP_ref(523, ASAP_UNDEF)

Note: if no NAME arg: $0000, entries in no particular order
HANDLE ($10,$D)
```

```
packed struct Plots {
  WORD total size;
  struct tPlot_entry {
    NA_WORD num;
    NA_WORD size;
    BYTE selected;
    BYTE DefinePlot.Plot_Type //(0..4)
    BYTE DefinePlot.Mark; // (0..4)
    BYTE 0?;
    VAR DefinePlot.x;
    VAR DefinePlot.y;
    VAR DefinePlot.Freq;
    VAR DefinePlot.Category;
    packed union {
      INT DefinePlot.Hist_Bucket_Width;
      BYTE 0;
    } p_widths;
    packed union {
      LIST DefinePlot.Include_Categories;
      BYTE 0;
    } p_includes;
  } Plot_entry[];
};
```

```
// ...
HANDLE GS_hPlots;
```

```
// ...
```

Drawing Menu (gdraw.o)

(not started)

```
void GD_Eraser( void );
#define GD_Eraser ASAP_ref(378, void (*const)( void ))
void GD_Text( void );
#define GD_Text ASAP_ref(379, void (*const)( void ))
void GD_Pen( void );
#define GD_Pen ASAP_ref(377, void (*const)( void ))
void GD_HVLine(BOOL Horz);
#define GD_HVLine ASAP_ref(376, void (*const)(BOOL))
void GD_Line( void );
#define GD_Line ASAP_ref(375, void (*const)( void ))
void GD_Circle( void );
#define GD_Circle ASAP_ref(374, void (*const)( void ))
void GD_Select( void );
#define GD_Select ASAP_ref(380, void (*const)( void ))
void GD_Contour( void );
#define GD_Contour ASAP_ref(381, void (*const)( void ))
```

Maths Menu (gmath.o)

(not started)

```
YCvtFtoWin
#define YCvtFtoWin ASAP_ref(435, ASAP_UNDEF)
GM_Value
#define GM_Value ASAP_ref(426, ASAP_UNDEF)
GM_Math1
#define GM_Math1 ASAP_ref(431, ASAP_UNDEF)
GM_Intersect
#define GM_Intersect ASAP_ref(427, ASAP_UNDEF)
GM_Derivative
#define GM_Derivative ASAP_ref(432, ASAP_UNDEF)
GM_TanLine
#define GM_TanLine ASAP_ref(430, ASAP_UNDEF)
GM_Inflection
#define GM_Inflection ASAP_ref(429, ASAP_UNDEF)
GM_Integrate
#define GM_Integrate ASAP_ref(428, ASAP_UNDEF)
GM_DistArc
#define GM_DistArc ASAP_ref(433, ASAP_UNDEF)
GM_Shade
#define GM_Shade ASAP_ref(434, ASAP_UNDEF)
```

Tools Menu (gtools.o)

(not started)

```
GT_QFloatCursorsInRange
#define GT_QFloatCursorsInRange ASAP_ref(536, ASAP_UNDEF)
GT_BackupToScr
#define GT_BackupToScr ASAP_ref(524, ASAP_UNDEF)
GT_FreeTrace
#define GT_FreeTrace ASAP_ref(533, ASAP_UNDEF)
GT_IncXY
#define GT_IncXY ASAP_ref(534, ASAP_UNDEF)
GT_WinBound
#define GT_WinBound ASAP_ref(547, ASAP_UNDEF)
GT_DspTraceCoords
```

```
#define GT_DspTraceCoords ASAP_ref(529, ASAP_UNDEF)
GT_DspFreeTraceCoords
#define GT_DspFreeTraceCoords ASAP_ref(528, ASAP_UNDEF)
GT_CursorKey
#define GT_CursorKey ASAP_ref(527, ASAP_UNDEF)
GT_WinCursor
#define GT_WinCursor ASAP_ref(548, ASAP_UNDEF)
GT_KeyIn
#define GT_KeyIn ASAP_ref(535, ASAP_UNDEF)
GT_SetGraphRange
#define GT_SetGraphRange ASAP_ref(542, ASAP_UNDEF)
GT_CenterGraphCursor
#define GT_CenterGraphCursor ASAP_ref(526, ASAP_UNDEF)
GT_Error
#define GT_Error ASAP_ref(531, ASAP_UNDEF)
GT_SetCursorXY
#define GT_SetCursorXY ASAP_ref(543, ASAP_UNDEF)
GT_ShowMarkers
#define GT_ShowMarkers ASAP_ref(544, ASAP_UNDEF)
GT_SelFunc
#define GT_SelFunc ASAP_ref(541, ASAP_UNDEF)
GT_DspMsg
#define GT_DspMsg ASAP_ref(530, ASAP_UNDEF)
GT_Open
#define GT_Open ASAP_ref(539, ASAP_UNDEF)
GT_SaveAs
#define GT_SaveAs ASAP_ref(540, ASAP_UNDEF)
GT_CalcDepVals
#define GT_CalcDepVals ASAP_ref(525, ASAP_UNDEF)
GT_Trace
#define GT_Trace ASAP_ref(545, ASAP_UNDEF)
GT_Regraph
#define GT_Regraph ASAP_ref(537, ASAP_UNDEF)
GT_Regraph_if_neccy
#define GT_Regraph_if_neccy ASAP_ref(538, ASAP_UNDEF)
GT_DE_Init_Conds
#define GT_DE_Init_Conds ASAP_ref(554, ASAP_UNDEF)
GT_Format
#define GT_Format ASAP_ref(532, ASAP_UNDEF)
GT_Set_Graph_Format
#define GT_Set_Graph_Format ASAP_ref(552, ASAP_UNDEF)
GT_ValidGraphRanges
#define GT_ValidGraphRanges ASAP_ref(546, ASAP_UNDEF)
GT_PrintCursor
#define GT_PrintCursor ASAP_ref(553, ASAP_UNDEF)
GYcoord
#define GYcoord ASAP_ref(549, ASAP_UNDEF)
GXcoord
#define GXcoord ASAP_ref(550, ASAP_UNDEF)
FLOAT round12_err(FLOAT x);
#define round12_err ASAP_ref(551, FLOAT (*const)(FLOAT))
```

Zoom Menu (gzoom.o)

(not started)

```
GZ_Center
#define GZ_Center ASAP_ref(556, ASAP_UNDEF)
GZ_Box
#define GZ_Box ASAP_ref(555, ASAP_UNDEF)
GZ_InOut
#define GZ_InOut ASAP_ref(559, ASAP_UNDEF)
GZ_Standard
#define GZ_Standard ASAP_ref(565, ASAP_UNDEF)
GZ_Previous
#define GZ_Previous ASAP_ref(561, ASAP_UNDEF)
GZ_Fit
#define GZ_Fit ASAP_ref(558, ASAP_UNDEF)
GZ_Square
```

```
#define GZ_Square ASAP_ref(564, ASAP_UNDEF)
GZ_Trig
#define GZ_Trig ASAP_ref(568, ASAP_UNDEF)
GZ_Decimal
#define GZ_Decimal ASAP_ref(557, ASAP_UNDEF)
GZ_SetFactors
#define GZ_SetFactors ASAP_ref(563, ASAP_UNDEF)
GZ_Integer
#define GZ_Integer ASAP_ref(560, ASAP_UNDEF)
GZ_Store
#define GZ_Store ASAP_ref(567, ASAP_UNDEF)
GZ_Recall
#define GZ_Recall ASAP_ref(562, ASAP_UNDEF)
```

Table (aptabled.o)

Everything but the paramaters is given by the object file.

```
const TAG Tableundef = UNDEFINED_TAG;
const TAG SaveTypes[2] = { GDB_TAG, 0 };

void te_init( void ); /* table editor init */
void AP_table(EVENT *event);

static void tablepaint(GRAPH_VARS *gv);
static void tableFormat( void ); /* format dialog */
static void TABLEMoveHighlight(WIN_RECT *r);
static void TABLEUnhighlight( void );
static void tablecelldisp( void );
static void clrcolstr(char* tmp);
static WORD getcolelem(HSYM);
static ESI getelement(HSYM y, WORD x);
static ESI getxelement(HANDLE h, WORD x);
/* step through var h for x element */
static BYTE columnsym( void );
static ESI tesymtoesi(HSYM tesym);
static void tetabletowin(EStackIndex expr, GRAPH_VARS *gv);
static void resetclip(GRAPH_VARS *gv); /* reset clipping region */
static void tableEnableFKeys( void ); /* set command state */
static void authTEnableFKeys( void ); /* and for author */

void Tablesetup( void );
WORD TA_callBack(WORD a, DWORD b);
FLOAT TblCptX(FLOAT f, GRAPH_VARS *gv);

static void tblauthFillEdit ( void );
static void getNameInfo (BYTE t, BOOL x, char *dest, char *sym, EStackIndex sym, WORD
tmpnamestart);
static void tableShiftDown( void );
static void tableShiftUp( void );
static void calcdispX(GRAPH_VARS *);
static void setuptblfunc(NG_func_vars, WORD x);
static void storetblfunc(LONG, EStackIndex src, WORD, *, BOOL d);
static void dipcurcoltbl(BOOL d);
static void storetblinputstart( void );
static void dispcolandprompt( void );

void cpt_tbl_fun(lwlw);
WORD get_tbl_point(EStackIndex lastParam);
void calcnewcol(EStackIndex esi, WORD col);
void setupcolexec();
void prepcoltostore();
void dispcolumnstbl();
void dispcolandprompt2();
ESI getnextcol();
void putnextcol();
void dispcolname();
```

```
void drawlines();
void TblStartValue();
ULONG FindTblFunc();

static void displayX();
static void displayXASK();
static void independtitle();

void cpt_tbinput_ASK();
void Tableinsert( void ); /* table menu options */
void Tabledelete( void );

static void authPutAwayFunc();
static void authPutAwayIndep();

void TblClear();
void empty_tblinput();

static void tbl_paint_seq();
static void tbl_cpt_seq_row();
static void tbl_cpt_seq_row_and_disp();
static void switchtoTauth();
static void incClipy();

.bss
static ULONG Options;
static WORD maxcolumn;
static BOOL focus;
static WINDOW wAuthor;
static WORD keyCode;
static WIN_RECT r;
static WIN_RECT rHighlight;
static BOOL bHighlight;
static WORD celloffset;
static FLOAT column;
static char columnstr[12];
static HANDLE hEdit2, hEdit1;
static HANDLE hText;
static TEXT_EDIT textEdit;
static WORD errheader;
static TAG tabletok[3];

struct TableWinVars {
  WORD NumCols;
  WORD NumRows;
  WORD Width;
  WORD Height;
  BYTE CurRow;
  BYTE CurCol;
  BYTE LeftCol;
  BYTE RightCol;
  WORD TopRow;    //Independent: ASK
  WORD BottomRow; //"
  BYTE ColNum[10];
  BYTE ScrollWid; //for Graph <-> Table
  FLOAT tblStart; //"
} table_win_vars1, table_win_vars2;

WORD table_flags;


//Resource entries:
DIALOG DataClearDlg;
MENU TableMenu;
DIALOG TableFormatDlg;
DIALOG TableDialog;
DIALOG StartValueWarnDlg;
```

Data/Matrix Editor

- Application
- Data Variables
- Data Variable Building

Application (apdmed.o)

```
void datapaint(KEYCODE key);
void datacelldisp( void );
```

```
const TAG Dataundef = UNDEFINED_TAG;
const TAG SaveTypes[4] = { DATA_TAG, MATRIX_TAG, LIST_TAG, 0 };
const TAG MatrixSaveTypes[2] = { MATRIX_TAG, 0 };
const TAG NonMatrixSaveTypes = DATA_TAG;
const TAG ListSaveTypes[2] = { LIST_TAG, 0 };

static const XREF CalcTypeRefs[12];

const BITMAP * const PlotMarks[5]; //bitmaps in resource object
const BITMAP * const PlotTypes[5];
```

```
void dm_init( void );
void AP_data(EVENT *event);
```

```
static SBYTE columnsym(void);
EStackIndex dmsymtoesi(HSYM hSym);
```

```
GetStatValue
#define GetStatValue ASAP_ref(283, ASAP_UNDEF)
dv_findColumn
#define dv_findColumn ASAP_ref(281, ASAP_UNDEF)
spike_in_editor
#define spike_in_editor ASAP_ref(278, ASAP_UNDEF)
spike_titles_in_editor
#define spike_titles_in_editor ASAP_ref(280, ASAP_UNDEF)
paint_all_except
#define paint_all_except ASAP_ref(285, ASAP_UNDEF)
```

```
?
?
?
?
```

```
SP_Define
#define SP_Define ASAP_ref(272, ASAP_UNDEF)
```

```
void initedit( void );
void copycurname(const char * name);
/* perhaps */
```

```
void resizelist(HSYM hsym, WORD IndexRow, EStackIndex OldTop);
```

```
static void growmat(WORD oldrow, WORD oldcol, SWORD newrow, SWORD newcol, BYTE matdata);
static void dmtempfolder(void);
```

```
void storelist(void);
void storematrix(EStackIndex OldTop);
void openmatrix(EStackIndex esi);
```

```
static HSYM VarStoreData( BYTE *DestVar, WORD Flags, WORD SourceSize, long parm1, short
parm2 );
```

```
void restorelist(WORD Zerosize);
void disp1DExpr(EStackIndex esi, HANDLE *hText, BOOL torf, unsigned short width);
void clearDMLedit(void);
void initcurdataptr(void);
```

```
static WORD CalcType, CalcRegEQ, CalcUseCategories;
static WORD PlotType, PlotMark, PlotUseCategories;
static WORD colint;
static WORD rowint;
static WORD width; //total used (changes with col width)
static BOOL focus;
static WINDOW wAuthor;
static WINDOW wData;
static WIN_RECT r;
static WIN_RECT rHighlight;
static BOOL bHighlight;
static WORD errheader;
static WORD PlotNum;
static WORD CellWidth;
static FLOAT row, column;
static char rowstr[4];
static char columnstr[12];
static HANDLE hEdit1; //Calc: Period
static HANDLE hEdit2; //Calc: Iterations
static HANDLE hEdit3; //Plot: Bucket Width
static HANDLE hEdit4; //Include Categories
static HANDLE hText;
static TEXT_EDIT textEdit;

static char _dmTokToStr[2];   //used to convert single char VAR
static char DlgTitle[30];

static WORD HeaderOldRow;     //Saved while changing header
static WORD StartType;

static char name[20];
```

```
WORD dmFlags;
HANDLE hdmfolder
BYTE dmTag;
WORD dmCellWidthOption;

char plotname[20];
EStackIndex plotsym;
char dmname[20]; //current name
EStackIndex dmsym;

BYTE dmCol; //used as a temporary, e.g. recalling vars
```

```
//Calculate dialog results (may well not be static)
static HSYM x, y, Freq, Category;
static BYTE xRow, yRow, FreqRow, CategoryRow; //decremented to 0

static WORD RowDim; //Dimension of first col in Calculation
```

```
struct DataVars {
  WORD col;
  WORD cur_col;
  WORD row;
  WORD cur_row;
  WORD NumCols;
  WORD NumRows;
  WORD cur_dispcol;
  WORD cur_disprow;
  WORD Width;
  WORD Height;
```

```
} dmdata;

DataVars* dataptr;
```

```
//Resources:
MENU DataMenu;
DIALOG DataFormatDlg;
DIALOG ResizeMatrixDlg;

MENU PlotSetupMenu;
DIALOG PlotSetupDlg;

DIALOG DataCalcDlg;
DIALOG AutoCalcWarnDlg;
```

```
Temporary Folder (hdmfolder):
9998
  c%d  STORE LIST for each column (& calculated cols)
  hc%d STORE RET EXPR calculation
  tc%d STORE RET STR fo each column header
```

Data Variables (data.o)

```
void store_data_var(EStackIndex name, BYTE bSetFlags);
#define store_data_var ASAP_ref(273, void (*const)(EStackIndex, BYTE))

ULONG total_len(BYTE *maxlist, BYTE *maxheader, BYTE *maxtitle);
/* returns total size of 'c', 'h', & 't' files in hdm folder
stores the max cols numbers for each of these respectively
*/

ULONG partial_len(const BYTE *name, BYTE *maxlist);
#define partial_len ASAP_ref(284, ULONG (*const)(const BYTE *, BYTE *))
/* scans hdm folder, calculate size of files matching "name%d" d<100
maxlist = last col found */

BYTE CharNumber(BYTE num, BYTE pos, BYTE *string);
#define CharNumber ASAP_ref(275, BYTE (*const)(BYTE, BYTE, BYTE *))
/* write num into &string[pos], returns new pos */

void _store_data_var(EStackIndex dest, BYTE bSetFlags, HSYM destvar, BYTE maxlist, BYTE
maxheader, BYTE maxtitle);
/* stores entire DATA structure to dest */

EStackIndex store_data_list(BYTE *name, EStackIndex dest, BYTE bSetFlags, HSYM destvar,
BYTE max);
/* stores lists matching name (1...max) to dest
returns final address
*/

recall_data_var
#define recall_data_var ASAP_ref(274, ASAP_UNDEF)
?
spike_optionD
#define spike_optionD ASAP_ref(276, ASAP_UNDEF)
?
?
?
?
spike_geo_titles
#define spike_geo_titles ASAP_ref(277, ASAP_UNDEF)
?
?
dv_create_graph_titles
```

```
#define dv_create_graph_titles ASAP_ref(279, ASAP_UNDEF)
spike_chk_gr_dirty
#define spike_chk_gr_dirty ASAP_ref(282, ASAP_UNDEF)
```

DATA format

```
packed struct DATA_VAR {
  BYTE CellWidthOption;
  BYTE NumCols;
  packed struct { //Including calculated ones
    BYTE col;
    STORE LIST rows;
  } Columns [NumCols];
  BYTE NumCalcs;
  packed struct {
    BYTE col;
    STORE RET EXPR colfunc;
  } Calculations [NumCalcs];
  BYTE NumTitles;
  packed struct {
    BYTE col;
    STORE RET STR col_name;
  } Titles [NumTitles];
  token $DD;
};
```

Data Variable Building (tabledata.o)

Just contains cmd_table & cmd_newdata prior to plus

```
void cmd_table(EStackIndex esi);
#define cmd_table ASAP_ref(915, void (*const)(EStackIndex))
void cmd_newdata(EStackIndex esi);
#define cmd_newdata ASAP_ref(867, void (*const)(EStackIndex))

static DWORD map_newdata(EStackIndex esi, BYTE *num);
/* returns amount of memory used */

BOOL map_eigVc(EStackIndex esi);
BOOL map_eigVl(EStackIndex esi);

cmd_blddata
#define cmd_blddata ASAP_ref(811, ASAP_UNDEF)

//more blddata routines
```

Program Editor (apprgmed.o)

```
void AP_program(EVENT *event);

static WORD NameLen;
static EStackIndex pesym;
static char pename[20];

static WORD StartType;
static TAG CurPrgmTok; //Prgm/Func
static TEXT_EDIT textEdit;
static WINDOW wProgEdit;

//Resources:
MENU ProgramMenu;
```

FUNC and PRGM Format

```
packed struct FUNC_VAR {
```

```
  packed union {
    packed struct {
      TEXT text; //$D: line separator
      NA_WORD cursor_offset;
      func_token prgm/func;
      token $E5;
    } untokenised;

    packed struct {
      packed struct { //Prog's require Prgm:...:EndPrgm
        token_deliminator delim;
        token tokens[];
        token 0;
      } statement[];
      VAR parameters [pars];
    } tokenised;
  } main_func;
  BYTE entry_count; //(limits recursion depth to 255)
  BYTE link flag
    //.0 : in use
  BYTE main flags
    //.0-2 : Style (not displayed order)
    //.3 : untokenised
    //.4 : Plot var
    //.6 : Graph 1 Plot On:Off
    //.7 : Graph 2 Plot On:Off
  token $DC;
};
```

(HOME screen functions have current plot flag set (as plots can be defined here))

Geometry

I really haven't looked into this one much:
is is a complicated program, and perhaps beyond the scope of this file.
Other notes on Geometry, it has a rather large bss of ~3.5kb, containing pointers to all of its definitions
One of the last object files on the calc so the differences between the 89 & 92+ are minor

```
9999      geometry
  a-j STORE FLOAT invalid?
```

GEOM format

```
struct GeomNum {
  SBYTE id;
  packed union {
    NA_WORD w;    // id=0;
    FLOAT f;      // id=-1;
  } num;
};

typedef struct GeomObject{
  char ID[3];
  NA_WORD polysides;
  NA_WORD side;
  NA_WORD p_a; //number previous attached points;
  NA_WORD type;
  BYTE 0;
  NA_WORD 1;
  packed union { //whatever else is required
    GeomNum inf[];
    WORD pt_num[p_a];
  };
```

```
} OBJECT;
```

```
packed struct GEOM {
  WORD format_flags;
  LONG 0;
  LONG flags?;
  WORD $1D;
  WORD ?
  LONG ?
  GeomNum ?;
  BYTE ?[2];
  OBJECT entry[];
};

packed struct MACRO {
  NAME Object_Name;
  TEXT Name;
  WORD flags?[8];
  BYTE ?;
  OBJECT create[];
};
```

OBJECT:
2 Axes
3 Grid
4+Object in file:

objects which automatically create points, create objects before them
i.e. Cirlce 'CE' creates a point 'PL'
Regular Polygon 'PR' is based on 2 Points 'PL' although automatically creates the others 'PP'
first 'PP' contains number of points: ? (auto generated poly point)

ID	store	type	name	
PL	1	1	01:Point	COORD x, COORD y
PP	0	-1	01:Point	OBJECT def[2], COORD [(f) angle|(i) prev]
PS	0	3	01:Point	OBJECT on, COORD (f)x, COORD (f)y (on Object)
PI	0	1	01:Point	OBJECT [2?n]
AX	1	3	0B:Axis	
GR	1	0	0C:Grid	
SE	1	0	02:Segment	OBJECT [2]
VE	1	0	03:Vector	OBJECT [2]
TT	1	0	04:Triangle	OBJECT [3]
PO	1	0	05:Polygon	OBJECT [n]
PR	1	0	05:Polygon	OBJECT [n] (regular)
DR	1	4	07:Line	OBJECT, COORD dx, COORD dy
DD	1	3	08:Ray	OBJECT, COORD dx, COORD dy
CE	1	2	09:Circle	OBJECT, COORD radius
AR	1	4	0A:Arc	OBJECT [3]
AN	1	4	0E:Angle	Angle
MI	1	1	01:Point	Midpoint
ME	1	4	07:Line	
PA	1	4	07:Line	Parallel Line
PE	1	4	07:Line	Perpendicular Line
BI	1	4	07:Line	Perpendicular Bisector
CO	1	2	09:Circle	
RE	0	-1	01:Point	
LI	0	-1	0D:Locus	
S2	1	2	03:Vector	
TR	1	1	01:Point	
SD	1	1	01:Point	
SP	1	1	01:Point	
RO	1	1	01:Point	
DI	1	1	01:Point	
IN	1	1	01:Point	

TE	0	-1	0F:Text	
NO	0	1	10:Number	
LO	0	3	10:Number	
SU	0	3	10:Number	
PN	0	3	10:Number	
MA	0	3	10:Number	
EQ	0	3	12:Equation	
AL	0	0	11:Property	
Pa	0	0	11:Property	
Pe	0	0	11:Property	
Me	0	0	11:Property	
Eq	0	0	11:Property	
FO	0	1	10:Number	

Text (aptexted.o)

```
const TAG SaveTypes[2] = { TEXT_TAG, 0 };

void AP_text(EVENT *event);

HSYM checkCurrent(EStackIndex SymName, TAG Type);
#define checkCurrent ASAP_ref(289, HSYM (*const)(EStackIndex, TAG))


static char *CurSym; //ptr in TextedName
static char TextedName[20];
static TEXT_EDIT textEdit;
static WINDOW wTextEd;

//Resource entries:
MENU TextMenu;
```

TEXT format

```
packed struct TEXT_VAR {
  WORD offset of cursor; //from struct:
  struct {
    char identifier to start line with;
      //' ' : None
      //$C  : Page Break
      //'C' : Command
      //'P' : PrintObj
    char text line [chars];
    BYTE last line ?0:KB_ENTER;
  } lines of text [lines];
  token $E0;
};
```

Numeric Solver (apslvr.o)

```
const char * eqn = "\0eqn";
#define EQN_VAR $(eqn)
const char * exp = "\0exp";
#define EXP_VAR $(exp)

FLOAT FP_LOWBOUND = -1e14; /* {0xC00E, 0x10} */
FLOAT FP_HIGHBOUND = 1e14; /* {0x400E, 0x10} */

void sl_init( void );
void AP_solver(EVENT *event);

MENU SolverMenu;
DIALOG SolverLastEqnsDlg;
DIALOG SolverFormatsDlg;
```

Self-Test (apslftst.o)

```
void AP_selftest(EVENT *event);

HANDLE hSlfTstWin;
```

PC Interfacing and Files

LITTLE ENDIAN USED

- Group Files
- Linking

Group Files

No definitions extend outside the sections in which they are defined, except for TGroupTag;

```
enum TGroupTag {
  EExpr=0,
  EList=4,EMatrix=6,
  EData=0xA,
  EText=0xB,
  EString=0xC,
  EGDB=0xD,
  EFigure=0xE,
  EPic=0x10,
  EPrgm=0x12,EFunc=0x13,
  EMacro=0x14,
  EExecute=0x15, /* In Link protocol, cause execution of assembly block */
  EDirL=0x19, //requesting directory lists
  PLUS EDirL=0x1A,
  PLUS ELDir=0x1B,
  PLUS EOther=0x1C,
  EBackup=0x1D,
  EFolder=0x1F, //group files
  PLUS EAsm=0x21,
};

TI92/TI92+ groupfile:
struct {
  char Signature[8]=PLUS?"**TI92P*":"**TI92**";
  WORD Version=1;
  char DefaultFolder[8];
  char Comment[40];
  WORD NumVars;
} Header;

//all entries have a common format, easiest to implement this by a function in writing to
files
template class<T> class Entry {
  Entry(LONG aOffset,aT,WORD aNum)
  :iOffset(aOffset),iT(aT),iNum(aNum) {}
  LONG iOffset;
  T iT;
  WORD iNum;
};

struct Sym {
  char name[8];
  WORD TGroupTag type;
};
```

```
Entry<Sym> vars(next,sym,entries); //all files & folders (note files belong to preceding
folders, if none: default)
Entry<void> separator(totalLen,,0x5AA5);
Entry<TIVar> entries(0,var,checksum);
```

Linking

- Sending and Receiving of Bytes
- Packets

Sending and Receiving of Bytes

This code is modified from tidev by David Ellsworth. It should compile under Windows.

```
#include <conio.h>

#define TIMAXTIME 100000
#define IO_DELAY 10

int parallel_port_table[3] = {0x3bc, 0x378, 0x278};
int lpt_base;

int get_lpt_base( void )
{
  int lpt_base = 0;
  for(int i=0; i<3, i++)
    if( probe_parallel(parallel_port_table[i]) )
       int lpt_base = parallel_port_table[i]);
  return lpt_base;
}

int init_ti_parallel(int lpt_base)
{
  _outp(lpt_base+0, 3);
  return 0;
}

int put_ti_parallel(int lpt_base, unsigned char data)
{
  int bit,i,titime=0;

  for (bit=0; bit<8; bit++) {
    for(i=0; i<IO_DELAY; i++) _inp(lpt_base+1);
    if (data&1) {
      _outp(lpt_base+0, 2);
      while (_inp(lpt_base+1)&0x10) if(titime++>TIMAXTIME) return -1;
      _outp(lpt_base+0, 3);
      while (!(_inp(lpt_base+1)&0x10)) if(titime++>TIMAXTIME) return -1;
    } else {
      _outp(lpt_base+0, 1);
      while (_inp(lpt_base+1)&0x20) if(titime++>TIMAXTIME) return -1;
      _outp(lpt_base+0, 3);
      while (!(_inp(lpt_base+1)&0x20)) if(titime++>TIMAXTIME) return -1;
    }
    data>>=1;
  }
  return 0;
}

int probe_parallel(int lpt_base)
{
  if((_inp(lpt_base+1)&0x30)==0x30) return 0;
  return 1;
}

int get_ti_parallel(int lpt_base)
{
  int bit,i,titime=0;
```

```
  unsigned char data=0, v;

  for (bit=0; bit<8; bit++) {
    while ((v=_inp(lpt_base+1)&0x30)==0x30) if(titime++>TIMAXTIME) return -1;

    if (v==0x10) {
      data=(data>>1)|0x80;
      _outp(lpt_base+0, 1);
      while (!(_inp(lpt_base+1)&0x20)) if(titime++>TIMAXTIME) return -1;
      _outp(lpt_base+0, 3);
    } else {
      data=data>>1;
      _outp(lpt_base+0, 2);
      while (!(_inp(lpt_base+1)&0x10)) if(titime++>TIMAXTIME) return -1;
      _outp(lpt_base+0, 3);
    }
    for(i=0; i<IO_DELAY; i++) _inp(lpt_base+1);
  }
  return (int)data;
}
```

Packets

```
enum calcType {
  TI82=2,
  TI83?=3,
  TI85=5,
  TIPLUS=8,
  TI92=9
}

//enum DEVICE_FLAGS { FROM_CALC = 0x80, FROM_SPECIAL };
#define CUSTOM_DEVICE 0x19
/* Has abililiy to run ASM on the machine */

struct deviceType {
  BYTE FromCalc : 1 ; //0 for computer to calc, 1 for calc->calc or calc->computer
  int reserved : 2;
  BYTE special : 1 ; //CBL / TI89
  calcType CalcSeries : 4 ;
};
//special types:
//12 : TI92 > CBL ???
//15 : ?? > TI85

enum command {
  VAR_HEADER=0x06, //varHeader
  WAIT_DATA=0x09,
  SEND=0x0D,
  DATA_PART=0x15,
  OK=0x56,
  CHK_ERROR=0x5A,
  ISREADY=0x68,
  SCREEN_DUMP=0x6D,
  CONTINUE=0x78,
  DIRECT_CMD=0x87, //WORD(KEY) (stored in packet word [dataLength])
  EOT=0x92,
  REQUEST=0xA2, //varHeader (var size 0)
  EXT_VAR_HEADER=0xC9, //varHeader (PLUS)
};

struct packet {
  BYTE deviceType;
  BYTE command;
  WORD dataLength; //or if only one word is required it is sent here (no checksum) (e.g.
WAIT_DATA)
  char data[dataLength];
```

```
  if(check)
    WORD checksum;
};

class varHeader {
  varHeader(WORD avarLen=0, WORD aflags=0, WORD avarType=0, char* aname);
  :varLen=avarLen, varType=aVarType
  {
    nameLen=strlen(name);
    strncpy(name,aname,nameLen);
  }
  LONG varLen;
  BYTE TGroupTag varType;
  BYTE nameLen;
  char name[nameLen];
};
```

After a packet is sent, OK or CHK_ERROR (maybe even nothing) is sent. Bail out if not OK
Note: for WAIT_DATA(x) x can be anything (supposed to be size)

exact BACKUP example:
sent in (size/1024) * 1024 byte blocks + whatever remainder

Computer	TI92
ISREADY REQUEST(0,0,EBackup,main/backup)	 OK OK

{
 	
OK
WAIT_DATA(size)
 	
OK
}
	{
	VAR_HEADER(size,0,EBackup,ROM_VER_STR)
	
	OK
	DATA_PART(size,data[])
	
	}

OK	EOT

simplifies to (ignoring OK's - fail if one not received after sending every packet):

Request Backup

```
send(ISREADY);
send(REQUEST(0,0,EBackup,"main\backup"));
for (all blocks) {
  if(receive() != VAR_HEADER(size,0,EBackup,ROM_VER_STR)) error();
  send(WAIT_DATA(size));
  if(receive() != DATA_PART(size,data[size]) error();
}
if(receive() != EOT) error();
```

Send Backup

```
send(ISREADY);
for (all blocks) {
  send(VAR_HEADER(size,0,EBackup,ROM_VER_STR));
  if (receive() != WAIT_DATA(size)) error();
  send(DATA_PART(size,data[size]));
}
send(EOT);
```

Request Single Variable

```
send(ISREADY);
```

```
send(REQUEST(0,0,type?,name)); //?probably not required
if (receive() != VAR_HEADER(size,0,type,name)) error();
send(WAIT_DATA(size));
if (receive() != DATA_PART(size,data[size])) error();
if (receive() != EOT) error();
```

Send Single Variable

```
send(ISREADY);
send(VAR_HEADER(size,0,type,name));
if (receive() != WAIT_DATA(size)) error();
send(DATA_PART(size,data[size]));
send(EOT);
```

Screen Dump

```
send(ISREADY);
send(SCREEN_DUMP);
if (receive() != DATA_PART(size,data[size])) error();
```

Direct Commands

Direct commands are limited to sending keypresses to the home screen, there is no feedback provided by this command. However, it can be used to be the calculator in remote mode, through sending <F5>+<DIAMOND>+<(>+<R>.

```
send(ISREADY);
for (;;) {
  send(DIRECT_CMD(key));
}
```

Remote mode does not use the standard packets. Keypresses are sent in plain form (WORDS), the TI then responds with the result of the output terminated by (BYTE)0xFF. I.e. 0xFF is sent after every character, when an entire expression has been entered it is returned with another trailing 0xFF.

92 Directory List

```
send(ISREADY);
send(REQUEST(0,0,EDirL,0));
if (receive() == VAR_HEADER(0,0,EDir,current)) {
  for (all vars) {
    send(WAIT_DATA(0));
    if(receive() != DATA_PART(size,var_name...)) error();
    trail = receive();
    if (trail == EOT) break;
    if (trail != CONTINUE) error();
}
```

Object File Formats

- Amiga DOS
- Fargo II
- Motorola COFF

Amiga DOS

```
//How names are stored (long length, and null padded to long boundry)
typedef struct {
  LONG sizeof(name);
  char szname[];
  ALIGN  4;
} NAME;
```

```
//Generic reloc
typedef struct {
  struct {
    LONG number of offsets in table;
    LONG hunk number;
    LONG offsets[offsets];
  } [];                 //all hunks
  LONG 0;
} RELOC;

enum ext_symbol {
  Ext_symb=0            //Symbol table
  Ext_def=1             //Relocatable definition
  Ext_abs=2             //Absolute definition
  Ext_res=3             //Reference to resident library [Obsolete]
  Ext_CommonDef=4       //Common definition (value is size in bytes)
  Ext_ref32=129         //32 bit absolute reference to symbol
  Ext_common=130        //32 bit absolute reference to COMMON block
  Ext_ref16=131         //16 bit PC-relative reference to symbol
  Ext_ref8=132          //8 bit PC-relative reference to symbol
  Ext_dext32=133        //32 bit data relative reference
  Ext_dext16=134        //16 bit data relative reference
  Ext_dext8=135         //8 bit data relative reference
//* These are to support some of the '020 and up modes that are rarely used
  Ext_relref32=136      //32 bit PC-relative reference to symbol
  Ext_relcommon=137     //32 bit PC-relative reference to COMMON block
//* for completeness... All 680x0's support this
  Ext_absref16=138      //16 bit absolute reference to symbol
//* this only exists on '020's and above, in the (d8,An,Xn) address mode
  Ext_absref8=139       //8 bit absolute reference to symbol
};


#define HunkUnit 999
#define HunkName 1000
#define HunkCode 1001
#define HunkData 1002
#define HunkBSS  1003
#define HunkR32  1004
#define HunkR16  1005
#define HunkR8   1006
#define HunkExt  1007
#define HunkSym  1008
#define HunkDbg  1009
#define HunkEnd  1010
#define HunkHeader 1011

struct HunkUnit {
  NAME Unit_Name="";
};
struct HunkName {
  NAME section_name;
};
struct HunkCode {
  LONG size;
  LONG code[size];
};
struct HunkData {
  LONG size;
  LONG data[size];
};
struct HunkBSS {
  LONG bsslen?;
};
struct HunkR32 {
  RELOC 32bitrelocs;
};
struct HunkR16 {
```

```
  RELOC 16bitrelocs;
};
struct HunkR8 {
  RELOC 8bitrelocs;
};
struct HunkExt {
  struct {
    int symbol_type : 8;
    LONG len : 24; //i.e len&0xFFFFFF
  } sym_info;
    char szname[];
    ALIGN 4;
  } symbol[];
  WORD 0;
};

struct Hunk_Sym
  struct {
    NAME symbol;
    LONG address;
  } [];
  LONG 0;
};
struct Hunk_Dbg {
  //undefined
  //...
};
struct HunkEnd {
  //always present (no extra data)
};
struct HunkHeader {
  NAME hunk[];
  LONG 0;
  LONG highest_hunk+1;
  LONG num_load_first;
  LONG num_load_last;
  LONG hunk_size[highest_hunk];
};
```

Fargo II

```
entries are displacements from previous entry -2
word defines value greater than byte => displ=(word)+$100
a word value of $FFFE specifies a jump of $10000 (no reloc is performed)
```

```
packed struct RELOC {
  bool nonbyte : 1 ;
  bool word : 1 ;
  packed union {
    int byte : 6;
    int word : 14;
    struct {
      int bytes : 2; //-1
      int first : 4;
      BYTE nibbles[bytes+1];
    } nibble;
  } reloc_entry;
};

struct runtime_info {
  WORD reloc_count;
  HANDLE libraries[lib_count];
  BYTE bss[bss_size];
};
```

```
struct {
  WORD TIsize; //00
  WORD ver='2' //02
  char type[4]="EXE "; //04
  char subtype[4]; //one of "APPL","PRGM","DLL " //08
  HANDLE runtime_info; //0C
  WORD reloc_offset; //0E
  WORD bss_offset; //10
  WORD import_offset; //12
  WORD export_offset; //14
  WORD comment_offset; //16
  WORD flags=1; //18 (.0=fargo 1 support; other bits not supported [exec error])
} header;


BYTE program [];


packed struct {
  RELOC reloc_address[];
  BYTE 0;
} reloc_table;

packed struct {
  WORD bss_size;
  RELOC bss_address[];
  BYTE 0;
} bss_table;

packed struct {
  WORD export_count;
  WORD entry_offset[export_count];
} export_table;

packed struct {
  WORD lib_count;
  WORD names_offset; //from start
  packed struct {
    BYTE ordinal_disp; //+1
    RELOC import_addresses[];
    BYTE 0;
  } ordinal[];
  BYTE 0;
} import_table;

TEXT lib_names[lib_count];
```

First ordinal_disp is first ordinal number referred to +1. Next entries are displacements from this first entry.

Motorola COFF

The format used by TI's compiler.
Found dumped in ROM 2.1, a search for ".text" in the ROM reveals the addresses:
all padded with junk from memory up to 32kb boundry, interesting in final case:
address fsize padded filename

Address	Size	Total	Name
$4D8000	$3083	32kb	win.o
$4E0000	$25D4	32kb	dialog.o
$4E8000	$524F	32kb	graph3d.o
$4F0000	$C592	64kb	aptabled.o (part of apdmed.c in padding)

First symbol is .file, with aux entry the given file name (aux entries follow their symbol entries)
Next are the exports from the program, with 1 aux entry each for size etc.
Next are the imports with no aux entries
The string table follows this.

Note: All symbol entries & auxiliary symbol entries occupy 18 bytes...
...relocs refer to a symbol index which includes all entries (index*18)

Here is a reduced form of the COFF specification (copied) from:
<http://www.delorie.com/djgpp/doc/incs-2.01/coff.h>
Note: magic number 0x150

```
/********************** FILE HEADER **********************/

#define M68KMAGIC 0x150
struct {
  WORD f_magic;         // magic number
  WORD f_NumSections;   // number of sections
  LONG f_timdat;        // time & date stamp
  LONG f_SymbolPtr;     // file pointer to symtab
  LONG f_NumSyms;       // number of symtab entries
  WORD f_opthdr;        // sizeof(optional hdr)
  WORD f_flags;         // flags
} external_filehdr;


/* Bits for f_flags:
 *     F_RELFLG    relocation info stripped from file
 *     F_EXEC              file is executable (no unresolved external references)
 *     F_LNNO              line numbers stripped from file
 *     F_LSYMS             local symbols stripped from file
 */

#define F_RELFLG  (0x0001)
#define F_EXEC          (0x0002)
#define F_LNNO          (0x0004)
#define F_LSYMS         (0x0008)


/********************** SECTION HEADER **********************/
//After file header, we have a section header for all the sections

struct {
  char s_name[8]; // section name
  LONG s_PhysAddr;      // physical address, aliased s_nlib
  LONG s_VirtAddr;      // virtual address
  LONG s_size;          // section size
  LONG s_SectionPtr;    // file ptr to raw data for section
  LONG s_RelocPtr;      // file ptr to relocation
  LONG s_LineNumsPtr;   // file ptr to line numbers
  WORD s_NumRelocs;     // number of relocation entries
  WORD s_NumLineNums;   // number of line number entries
  LONG s_flags;         // flags
} external_SectionHdr[NumSections];

/*
 * names of "special" sections
 */
#define _TEXT     ".text"
#define _DATA     ".data"
#define _BSS      ".bss"
#define _COMMENT ".comment"
#define _LIB".lib"

/*
```

```
 * s_flags "type"
 */
#define STYP_TEXT  (0x0020)   // section contains text only
#define STYP_DATA  (0x0040)   // section contains data only
#define STYP_BSS   (0x0080)   // section contains bss only


/********************** RELOCATION DIRECTIVES **********************/

struct {
  LONG r_vaddr;
  LONG r_symndx;
  WORD r_type;
} external_reloc[NumRelocs];


/********************** SYMBOLS **********************/
struct sym_and_aux {
  struct sym
  {
    union {
      char e_name[8];
      struct {
        LONG e_zeroes;
        LONG e_offset;
      } e;
    } e;
    LONG e_value; //i.e. address of symbol in file
    short e_scnum; //section number
    WORD e_type;
    BYTE e_auexent_sclass;
    BYTE e_num_auxent;
  } external_syment;

  union aux {
    struct {
      LONG x_tagndx;    // str, un, or enum tag indx (size of section, 0 otherwise)
      union {
        struct {
          WORD x_lnno;  // declaration line number
          WORD x_size;  // str/union/array size
        } x_lnsz;
        LONG x_fsize;   // size of function
      } x_misc;
      union {
        struct {        // if ISFCN, tag, or .bb
          LONG x_lnnoptr;     // ptr to fcn line # (0)
          LONG x_endIndex;    // entry ndx past block end (count up exports *2)
        } x_fcn;
        struct {        // if ISARY, up to 4 dimen.
          WORD x_dimen[4];
        } x_ary;
      } x_fcnary;
      WORD x_tvndx;     // tv index (not sure how this is used)
    } x_sym;

    union {
      char x_fname[14]; //zero padded up to 18 (not packed union)
      struct { //or if in string table
        LONG x_zeroes;
        LONG x_offset;
      } x_n;
    } x_file;
  } external_auxent[e_num_auxent];
}[NumSyms];

//Symbol is not part of section (i.e. -ve):
#define N_UNDEF   ((short)0)  /* undefined symbol */
#define N_ABS     ((short)-1) /* value of symbol is absolute */
```

```
#define N_DEBUG   ((short)-2) /* debugging symbol -- value is meaningless */

//exernal_syment.e_type:

enum main_type {
  DT_NON=0x00, // no derived type
  DT_PTR=0x10, // pointer
  DT_FCN=0x20, // function
  DT_ARY=0x30  // array
};
//or with:
enum var_type {
  T_NULL=0,
  T_VOID=1,
  T_CHAR=2,
  T_SHORT=3,
  T_INT=4,
  T_LONG=5,
  T_FLOAT=6,
  T_DOUBLE=7,
  T_STRUCT=8,
  T_UNION=9,
  T_ENUM=10,
  T_MOE=11,
  T_UCHAR=12,
  T_USHORT=13,
  T_UINT=14,
  T_ULONG=15,
  T_LNGDBL=16 //clash!?
};


/****************AUX_ENT STORAGE CLASSES **********************/

// This used to be defined as -1, but now n_sclass is unsigned.
#define C_EFCN          0xff  // physical end of function
#define C_NULL          0
#define C_AUTO          1     // automatic variable
#define C_EXT           2     // external symbol
#define C_STAT          3     // static
#define C_REG           4     // register variable
#define C_EXTDEF  5     // external definition
#define C_LABEL         6     // label
#define C_ULABEL  7     // undefined label
#define C_MOS           8     // member of structure
#define C_ARG           9     // function argument
#define C_STRTAG  10    // structure tag
#define C_MOU           11    // member of union
#define C_UNTAG         12    // union tag
#define C_TPDEF         13    // type definition
#define C_USTATIC 14    // undefined static
#define C_ENTAG         15    // enumeration tag
#define C_MOE           16    // member of enumeration
#define C_REGPARM 17    // register parameter
#define C_FIELD         18    // bit field
#define C_AUTOARG 19    // auto argument
#define C_LASTENT 20    // dummy entry (end of block)
#define C_BLOCK         100   // ".bb" or ".eb"
#define C_FCN           101   // ".bf" or ".ef"
#define C_EOS           102   // end of structure
#define C_FILE          103   // file name
#define C_LINE          104   // line # reformatted as symbol table entry
#define C_ALIAS         105   // duplicate tag
#define C_HIDDEN  106   // ext symbol in dmert public lib

LONG size_stringtable;
char string_table[];
```

Token Lists

```
enum TOK_TYPE {
  varname = 0x01, var = 0x02,
  local = 0x3, //ignored
  systok = 0x04,
  arb_const = 0x05,
  INT = 0x06,
  FRAC = 0x07,
  BCD = 0x08,
  symb_const = 0x09,
  imag = 0x0A,
  inf_undef = 0x0B,
  bool = 0x0C,
  STR = 0x0D,
  NOTHING = 0x0E,
  archyp = 0x0F,
  hyptrig = 0x10,
  arctrig = 0x11,
  rarctrig = 0x12,
  trig = 0x13,
  tanvar = 0x14,
  log = 0x15, //log, exp
  f1expr = 0x16
  texpand = 0x17, //texpand, tcollect, approx, ?
  post_pr_vec_op = 0x18
  store_with = 0x19
  binary = 0x1A
  relationalop = 1x1B
  addsub = 0x1C
  muldiv = 0x1D
  pow = 0x1E
  solve = 0x1F
  f2expr = 0x20
  f_list_expr = 0x21, //polyeval, ?
  rowop3 = 0x22,
  desolve = 0x23,
  rowop4 = 0x24,
  arclen = 0x25,
  seqop = 0x26,
  f_varparm = 0x27,
  list = 0x28,
  type = 0x29,
  ext_func = 0x2A,
  ext_instr = 0x2B,
  delim = 0x2C,
}
```

Structures

```
BYTE struct Parms {
  int Min : 4;
  int Max : 4;
};
#define NO_MAX 0xF

//Defparms depends on Parms, if:
//(Min==Max) Starting parm number (cyclic, i.e. order to display)
//(Min!=Max) Number of variable parameters
```

Conventions

- `TYPE` prefixed token identifying a new type;

- otherwise token represents an expression or a subtype of an expression:
- TOKEN another token follows, groups similar tokens together
- INTERNAL used in internal calculations, never stored or seen in final results (used for complex numbers & all trig operations)
- SAVETYPE can be used as a file type identifier for VarOpen, VarSave etc.
- STRUCT a special defined structure follows this token
- SUBTYPE trivial - identified with a subtype
- NOTHING is special ... as named
- DISPLACEMENT for jumps are 2 byte quantities of same endianness as INT (LLHH +ve mem)
- CF_* flags as defined (section-??)
- _ERROR_ id used for a bad token (i.e. not present)

Any other parameters of undefined type to be processed as EXPR, mostly numbers

For fixed number of paramaters the defined tokens in (...) follow on the estack
For variable number, the end is marked by the END_TAG token.

Where the parameters are unclear, or unconventional the estack order is listed in {...}
...the same conventions are followed

Other trivial subtypes: both matricies

- DMSNUMBER An angle in Degrees, Minutes, & Seconds.
- VEC A vector of any type:

Vector Types

Rectangular	[X,Y[,Z]]
Polar	[R,THETA]
Cylind	[R,THETA,Z]
Spherical	[R,THETA,PHI]

tokenisation conventions

How expressions are observed in parameter lists etc.

Equation	standard tokenisation
x/y	x*y^(-1)
sin(x)	trig(x,0) [radian conversion]
cos(x)	trig(x,1)
tan(x)	trig(x,0)*trig(x,1)^(-1)
x^y	e^(ln(x)*y)
sqrt(x)	x^(1/2)

Note: Mode dependant calculations are performed by converting expressions to a specific format. i.e. for trig values are converted to radian trig functions.

This table was not present prior to the PLUS. Jump tables used to be used instead. However, tokens have been placed in gaps, so the tokens are less easily processed this way.
This tables was introduced, with a TYPE field grouping the tokens:

- Main
- System Variables
- Extra
- Instruction
- Enumerations

Main

Tag	Group	Description
00	01	**SUBTYPE STRUCT VAR** {NAME}
01	02	**CF_ENHANCED SUBTYPE VAR** q (not used - $1B is used normally)
02	02	**SUBTYPE VAR** r
03	02	**SUBTYPE VAR** s

04	02	**SUBTYPE VAR** t
05	02	**SUBTYPE VAR** u
06	02	**SUBTYPE VAR** v
07	02	**SUBTYPE VAR** w
08	02	**SUBTYPE VAR** x
09	02	**SUBTYPE VAR** y
0A	02	**SUBTYPE VAR** z
0B	02	**SUBTYPE VAR** a
0C	02	**SUBTYPE VAR** b
0D	02	**SUBTYPE VAR** c
0E	02	**SUBTYPE VAR** d
0F	02	**SUBTYPE VAR** e
10	02	**SUBTYPE VAR** f
11	02	**SUBTYPE VAR** g
12	02	**SUBTYPE VAR** h
13	02	**SUBTYPE VAR** I
14	02	**SUBTYPE VAR** j
15	02	**SUBTYPE VAR** k
16	02	**SUBTYPE VAR** l
17	02	**SUBTYPE VAR** m
18	02	**SUBTYPE VAR** n
19	02	**SUBTYPE VAR** o
1A	02	**SUBTYPE VAR** p
1B	02	**SUBTYPE VAR** q
1C	04	**SUBTYPE VAR TOKEN** system variable token
1D	05	@xxx {BYTE xxx} (Arbitrary real)
1E	05	@nxxx {BYTE xxx} (Arbitrary integer)
1F	06	**SUBTYPE STRUCT INT** positive integer
20	06	**SUBTYPE STRUCT INT** negative integer
21	07	**STRUCT** positive fraction
22	07	**STRUCT** negative fraction
23	08	**STRUCT** BCD efloat
24	09	_pi_
25	09	_exp_
26	0A	_imaginary_
27	0B	-_infinity_
28	0B	_infinity_
29	0B	**(PLUS) CF_ENHANCED** _(+/-)__infinity_
2A	0B	undef (UNDEFINED_TAG)
2B	0C	false
2C	0C	true
2D	0D	**TYPE STRUCT STR** (string)
2E	0E	**NOTHING** (allows expressions to be accepted with missing terms)
2F	0F	acosh(EXPR)
30	0F	asinh(EXPR)
31	0F	atanh(EXPR)
		Internal variant of inverse hyperbolic trig functions? [3]
35	10	cosh(EXPR)
36	10	sinh(EXPR)
37	10	tanh(EXPR)
		Internal variant of hyperbolic trig functions? [3]
3B	11	acos(EXPR)
3C	11	asin(EXPR)
3D	11	atan(EXPR)
		Internal variant of inverse trig functions? [3]
41	12	**INTERNAL** arcos {EXPR} (not used)
42	12	**INTERNAL** arsin {EXPR} (radians - also used for acos)
43	12	**INTERNAL** artan {EXPR} (radians)

```
44   13     cos(EXPR)
45   13     sin(EXPR)
46   13     tan(EXPR)
       Internal variant of trig functions? [3]
4A   14     INTERNAL tan(EXPR) (not used)
4B   16     abs(EXPR)
4C   16     angle(EXPR)
4D   16     ceiling(EXPR)
4E   16     floor(EXPR)
4F   16     int(EXPR)
50   16     sign(EXPR)
51   16     _sqrt_(EXPR)
52   15     _exp_^(EXPR)
53   15     ln(EXPR)
54   15     log(EXPR)
55   16     fPart(EXPR)
56   16     iPart(EXPR)
57   16     conj(EXPR)
58   16     imag(EXPR)
59   16     real(EXPR)
5A   17     approx(EXPR)
5B   17     tExpand(EXPR)
5C   17     tCollect(EXPR)
5D   16     getDenom(EXPR)
5E   16     getNum(EXPR)
5F   0      CF_ENHANCED _ERROR_(EXPR)
60   16     cumSum(LIST)
61   16     det(MAT)
62   16     colNorm(MAT)
63   16     rowNorm(MAT)
64   16     norm(MAT)
65   16     mean(LIST)
66   16     median(LIST)
67   16     product(LIST)
68   16     stdDev(LIST)
69   16     sum(LIST)
6A   16     variance(LIST)
6B   16     unitV(VEC)
6C   16     dim(MAT)
6D   16     mat|>list(MAT)
6E   16     newList(NUMELEMENTS)
6F   16     rref(MAT)
70   16     ref(MAT)
71   16     identity(INT)
72   16     diag(MAT)
73   16     colDim(MAT)
74   16     rowDim(MAT)
75   18     MAT_transpose_ {MAT}
76   18     EXPR ! {EXPR}
77   18     EXPR % {EXPR}
78   18     EXPR _radians_ {EXPR}
79   18     not(EXPR) (no brackets on PLUS)
7A   18     _negative_ EXPR {EXPR}
7B   18     SUBTYPE STRUCT VEC_Polar {MAT} ([[R,_ANGLE_THETA_]])
7C   18     SUBTYPE STRUCT VEC_Cylind {MAT} ([[R,_ANGLE_THETA_,Z]])
7D   18     SUBTYPE STRUCT VEC_Sphere {MAT} ([[R,_ANGLE_THETA_,_ANGLE_PHI_]])
7E   18     CF_ENHANCED INTERNAL ( {EXPR}
7F   19     CF_ENHANCED EXPR -> VAR {VAR,EXPR} ? combine (AB) {EXPR A,EXPR B) (no art", strange behaviour)
```

```
80   19     EXPR -> VAR {VAR,EXPR}
81   19     EXPR | CONDITION {EXPR,CONDITION}
82   1A     EXPR1 xor EXPR2 {EXPR1,EXPR2}
83   1A     EXPR1 or EXPR2 {EXPR1,EXPR2}
84   1A     EXPR1 and EXPR2 {EXPR1,EXPR2}
85   1B     EXPR1 < EXPR2 {EXPR1,EXPR2}
86   1B     EXPR1 <= EXPR2 {EXPR1,EXPR2}
87   1B     SAVETYPE EXPR1 = EXPR2 {EXPR1,EXPR2}
88   1B     EXPR1 >= EXPR2 {EXPR1,EXPR2}
89   1B     EXPR1 > EXPR2 {EXPR1,EXPR2}
8A   1B     EXPR1 /= EXPR2 {EXPR1,EXPR2}
8B   1C     EXPR1 + EXPR2 {EXPR2,EXPR1}
8C   1C     EXPR1 .+ EXPR2 {EXPR2,EXPR1}
8D   1C     EXPR1 - EXPR2 {EXPR2,EXPR1}
8E   1C     EXPR1 .- EXPR2 {EXPR2,EXPR1}
8F   1D     EXPR1 * EXPR2 {EXPR2,EXPR1}
90   1D     EXPR1 .* EXPR2 {EXPR2,EXPR1}
91   1D     EXPR1 / EXPR2 {EXPR2,EXPR1}
92   1D     EXPR1 ./ EXPR2 {EXPR2,EXPR1}
93   1E     EXPR1 ^ EXPR2 {EXPR1,EXPR2}
94   1E     EXPR1 .^ EXPR2 {EXPR1,EXPR2}
95   20     INTERNAL trig {EXPR,INT} (=cos(EXPR+(INT-1)*_pi_/2) => INT=0 sin; 1
96   1F     solve(EQUATION,VAR)
97   1F     cSolve(EQUATION,VAR)
98   1F     nSolve(EQUATION,VAR)
99   1F     zeros(EXPR,VAR)
9A   1F     cZeros(EXPR,VAR)
9B   1F     fMin(EXPR,VAR)
9C   1F     fMax(EXPR,VAR)
9D   20     INTERNAL complex {EXPRi,EXPRr}
9E   21     polyEval(LIST,EXPR)
9F   20     randPoly(VAR,ORDER)
A0   20     crossP(VEC1,VEC2)
A1   20     dotP(VEC1,VEC2)
A2   20     gcd(EXPR1,EXPR2)
A3   20     lcm(EXPR1,EXPR2)
A4   20     mod(EXPR1,EXPR2)
A5   20     intDiv(EXPR1,EXPR2)
A6   20     remain(EXPR1,EXPR2)
A7   20     nCr(EXPR1,EXPR2)
A8   20     nPr(EXPR1,EXPR2)
A9   20     P|>Rx(REXPR,_THETA_EXPR)
AA   20     P|>Ry(REXPR,_THETA_EXPR)
AB   20     R|>P_THETA_(XEXPR,YEXPR)
AC   20     R|>Pr(XEXPR,YEXPR)
AD   20     augment(MAT1,MAT2)
AE   20     newMat(NUMROWS,NUMCOLUMNS)
AF   20     randMat(NUMROWS,NUMCOLUMNS)
B0   20     simult(MAT,VEC)
B1   27     CF_NEW part(EXPR[,#]) (removed from 92, strange position)
B2   21     exp|>list(EXPR,VAR)
B3   20     randNorm(MEAN,SD)
B4   22     mRow(EXPR,MAT,INDEX)
B5   22     rowAdd(MAT,RINDEX1,RINDEX2)
B6   22     rowSwap(MAT,RINDEX1,RINDEX2)
B7   25     arcLen(EXPR,VAR,START,END)
B8   26     nInt(EXPR,VAR,LOW,UP)
B9   26     _PI(product)_(EXPR,VAR,LOW,HIGH)
```

```
BA   26    _SIGMA_(EXPR,VAR,LOW,HIGH)
BB   24    mRowAdd(EXPR,MAT,INDEX1,INDEX2)
BC   27    CF_ENHANCED ans([INT])
BD   27    CF_ENHANCED entry([INT])
BE   27    exact(EXPR[,TOL])
BF   27    (PLUS) log[EXPR2](EXPR1) {EXPR1,EXPR2} (ln(EXPR1)/ln(EXPR2))
C0   27    comDenom(EXPR[,VAR])
C1   27    expand(EXPR[,VAR])
C2   27    factor(EXPR[,VAR])
C3   27    cFactor(EXPR[,VAR])
C4   27    _integrate_(EXPR,VAR[,LOW,UP])
C5   27    _differentiate_(EXPR,VAR[,ORDER])
C6   27    avgRC(EXPR,VAR[,H])
C7   27    nDeriv(EXPR,VAR[,H])
C8   27    taylor(EXPR,VAR,ORDER[,POINT])
C9   27    limit(EXPR,VAR,POINT[,DIRECTION])
CA   27    propFrac(EXPR[,VAR][,TOL])
CB   27    when(CONDITION,TRUE[,FALSE][,UNDEF])
CC   27    round(EXPR[,DIGITS])
CD   27    SUBTYPE STRUCT DMSNUMBER {EXPR DD[,[EXPR MM][,EXPR SS]]}
CE   27    left(STRING[,NUM])
CF   27    right(STRING[,NUM])
D0   27    mid(STRING,START[,COUNT])
D1   27    shift(LIST[,INT]) (PLUS...)
D2   27    seq(EXPR,VAR,LOW,HIGH[,STEP])
D3   27    list|>mat(LIST[,ELEMENTS PER ROW])
D4   27    subMat(MAT[,ROW1][,COL1][,ROW2][,COL2])
D5   27    VAR[ROW[,COL]] {VAR,ROW[,COL]} (subscript - matrix/list element access)
D6   27    rand([INT])
D7   27    min(LIST|EXPR1,EXPR2)
D8   27    max(LIST|EXPR1,EXPR2)
D9   28    TYPE STRUCT LIST/MAT (LIST_TAG={) (MAT=list of rows)
DA   28    VAR([EXPR1[,EXPR2[,...]]]) {VAR,[EXPR1[,EXPR2[,...]]]} (user func)
DB   28    CF_ENHANCED SAVETYPE INTERNAL TYPE MAT (MATRIX_TAG=[) (Data Editor Token
           [(EXPR1,...) )
DC   29    TYPE STRUCT PRGM/FUNC (program/function)
DD   29    SAVETYPE TYPE STRUCT DATA
DE   29    SAVETYPE TYPE STRUCT GDB
DF   29    SAVETYPE TYPE STRUCT PIC (picture)
E0   29    SAVETYPE TYPE STRUCT TEXT_VAR
E1   29    SAVETYPE TYPE STRUCT FIG (figure)
E2   29    SAVETYPE TYPE STRUCT MAC (macro)
E3   2A    TOKEN extended token
E4   2B    TOKEN instruction token (doesn't return a value)
E5   2C    ) or } (END_TAG)
E6   2C    (C)STR (comment) {STR}
E7   2C    : between tokens on same line
E8   2C    : [ENTER] marks end of line in TI-BASIC
E9   2C    : end of estack (TI-BASIC program)
EA   18    CF_NEW _(+/-)_EXPR
EB   1C    CF_NEW EXPR1 _(+/-)_ EXPR2 {EXPR2,EXPR1}
EC   29    CF_ENHANCED INTERNAL STRUCT HOME_MESSAGE {NAME} (error/Done without ")
ED   16    CF_NEW eigVc(MAT)
EE   16    CF_NEW eigVl(MAT)
EF   18    CF_NEW EXPR ' (prime) {EXPR}
F0   3     CF_CONVERT VAR {VAR} (parameter/local variable reference)
F1   23    CF_NEW deSolve(EQUATION,INDEP-VAR,DEPEND-VAR[])
F2   18    CF_NEW FUNC_NAME'(FUNC_PARAMS) (prime) {FUNC_NAME,FUNC_PARAMS)}
```

```
F3   29     CF_NEW TYPE STRUCT ASM
F4   16     CF_NEW isPrime(INT)
F5   21     CF_ENHANCED INTERNAL ? _ERROR_(EXPR1,EXPR2)
F6   17     CF_ENHANCED INTERNAL ? _ERROR_(EXPR)
F7   17     CF_ENHANCED INTERNAL ? _ERROR_(EXPR)
F8   29     CF_NEW TYPE OTH (other)
F9   27     CF_NEW rotate(LIST|STRING|INT[,COUNT])
```

System Variables ($1C)

Tag	Struct ID	Description
01	1	x_bar_
02	1	y_bar_
03	1	_SIGMA_x
04	1	_SIGMA_x_^2_
05	1	_SIGMA_y
06	1	_SIGMA_y_^2_
07	1	_SIGMA_xy
08	1	Sx
09	1	Sy
0A	1	_sigma_x
0B	1	_sigma_y
0C	1	nStat
0D	1	minX
0E	1	minY
0F	1	q1
10	1	medStat
11	1	q3
12	1	maxX
13	1	maxY
14	1	corr
15	1	R_^2_
16	1	medx1
17	1	medx2
18	1	medx3
19	1	medy1
1A	1	medy2
1B	1	medy3
1C	2	xc
1D	2	yc
1E	2	zc
1F	2	tc
20	2	rc
21	2	_THETA_c
22	2	nc
23	2	xfact *
24	2	yfact *
25	2	zfact *
26	2	xmin
27	2	xmax
28	2	xscl
29	2	ymin
2A	2	ymax

```
2B 2     yscl
2C 2     _DELTA_x
2D 2     _DELTA_y
2E 2     xres
2F 2     xgrid
30 2     ygrid
31 2     zmin
32 2     zmax
33 2     zscl
34 2     eye_THETA_
35 2     eye_PHI_
36 2     _THETA_min
37 2     _THETA_max
38 2     _THETA_step
39 2     tmin
3A 2     tmax
3B 2     tstep
3C 2     nmin
3D 2     nmax
3E 2     plotStrt
3F 2     plotStep
40 3     zxmin
41 3     zxmax
42 3     zxscl
43 3     zymin
44 3     zymax
45 3     zyscl
46 3     zxres
47 3     z_THETA_min
48 3     z_THETA_max
49 3     z_THETA_step
4A 3     ztmin
4B 3     ztmax
4C 3     ztstep
4D 3     zxgrid
4E 3     zygrid
4F 3     zzmin
50 3     zzmax
51 3     zzscl
52 3     zeye_THETA_
53 3     zeye_PHI_
54 3     znmin
55 3     znmax
56 3     zpltstep
57 3     zpltstrt
58 4     seed1 *
59 4     seed2 *
5A 4     ok
5B 4     errornum
5C 4     sysMath *
5D 6     sysData
5E 6     regEq (Name=Null)
5F 6     regCoef
60 7     tblInput
61 7     tblStart
```

```
62 7      _DELTA_tbl
63 8      _ERROR_ fldpic?
64 2      CF_NEW eye_PSI_
65 2      CF_NEW tplot
66 2      CF_NEW diftol
67 3      CF_NEW zeye_PSI_
68 2      CF_NEW t0
69 2      CF_NEW dtime
6A 2      CF_NEW ncurves
6B 2      CF_NEW fldres
6C 2      CF_NEW Estep
6D 3      CF_NEW zt0de
6E 3      CF_NEW ztmaxde (Name=ztmax)
6F 3      CF_NEW ztstepde (Name=ztstep)
70 3      CF_NEW ztplotde
71 2      CF_NEW ncontour
```

Extra ($E3)

```
Tag Description
01  # STRING-EXPR (indirection) {STRING-EXPR}
02  getKey()
03  getFold()
04  switch([INT])
05  CF_NEW UNIT1 |> UNIT2 {UNIT2,UNIT1} (convert)
06  ord(STRING)
07  expr(STRING)
08  char(INT)
09  string(EXPR)
0A  getType(VAR)
0B  getMode(STRING)
0C  setFold(VAR)
0D  ptTest(X,Y)
0E  pxlTest(ROW,COLUMN)
0F  setGraph(STRING,STRING)
10  setTable(STRING,STRING)
11  setMode(STRING,STRING)
12  format(EXPR[,STRING])
13  inString(SEARCH-STRING,SUBSTRING[,START])
14  STRING1 & STRING2 (append) {STRING2,STRING1}
15  DMSNUMBER |>DD {DMSNUMBER}
16  EXPR |>DMS {EXPR}
17  VEC |>Rect {VEC}
18  VEC |>Polar {VEC}
19  VEC |>Cylind {VEC}
1A  VEC |>Sphere {VEC}
1B  CF_NEW PARSER (  (CF_NEW means unportable here) (no table entry)
1C  CF_NEW PARSER )
1D  CF_NEW PARSER [
1E  CF_NEW PARSER ]
1F  CF_NEW PARSER { (no table entry)
20  CF_NEW PARSER }
21  CF_NEW PARSER ,
22  CF_NEW PARSER ;
```

```
23 CF_NEW PARSER /_ (angle sign)
24 CF_NEW PARSER ' (no table entry)
25 CF_NEW PARSER " (no table entry)
26 CF_NEW (EXPR1 /_ EXPR2) {EXPR2,EXPR1} (complex number - magnitude, angle)
27 CF_NEW tmpCnv(EXPR1,EXPR2)
28 CF_NEW _DELTA_tmpCnv(EXPR1,EXPR2)
29 CF_NEW getUnits([])
2A CF_NEW setUnits(LIST)
2B CF_NEW 0bx {INT x}
2C CF_NEW 0hx {EXPR x}
2D CF_NEW INT|>Bin {INT}
2E CF_NEW INT|>Dec (INT)
2F CF_NEW INT|>Hex {INT}
30 CF_NEW det(MAT,TOL)
31 CF_NEW ref(MAT,TOL)
32 CF_NEW rref(MAT,TOL)
33 CF_NEW simult(MAT,VEC,TOL)
34 CF_NEW getConfg([])
35 CF_NEW augment(MAT1;MAT2)
```

Instruction ($E4)

```
Tag Description
01  ClrDraw
02  ClrGraph
03  ClrHome
04  ClrIO
05  ClrTable
06  Custom
07  Cycle
08  Dialog
09  DispG
0A  DispTbl
0B  Else (If...Endif)
0C  EndCustm
0D  EndDlog
0E  EndFor {DISPLACEMENT}
0F  EndFunc
10  EndIf
11  EndLoop {DISPLACEMENT}
12  EndPrgm
13  EndTBar
14  EndTry {DISPLACEMENT}
15  EndWhile {DISPLACEMENT}
16  Exit
17  SAVETYPE Func
18  Loop
19  SAVETYPE Prgm
1A  ShowStat
1B  Stop
1C  Then
1D  Toolbar
1E  Trace
1F  Try
20  ZoomBox
21  ZoomData
22  ZoomDec
23  ZoomFit
```

```
24 ZoomIn
25 ZoomInt
26 ZoomOut
27 ZoomPrev
28 ZoomRcl
29 ZoomSqr
2A ZoomStd
2B ZoomSto
2C ZoomTrig
2D DrawFunc EXPR
2E DrawInv EXPR
2F Goto LABEL {EXPR}
30 Lbl LABEL {EXPR}
31 Get VAR
32 Send LIST
33 GetCalc VAR
34 SendCalc VAR
35 NewFold FOLDERNAME
36 PrintObj VAR
37 RclGDB VAR
38 StoGDB VAR
39 ElseIf CONDITION
3A If CONDITION
3B If CONDITION Then {CONDITION}
3C RandSeed EXPR
3D While CONDITION
3E LineTan EXPR,POINT
3F CopyVar VAR1,VAR2
40 Rename OLDVARNAME,NEWVARNAME
41 Style EXPR,STRING
42 Fill EXPR,VAR
43 Request STRING,VAR
44 PopUp ITEMLIST,VAR
45 PtChg X,Y
46 PtOff X,Y
47 PtOn X,Y
48 PxlChg ROW,COLUMN
49 PxlOff ROW,COLUMN
4A PxlOn ROW,COLUMN
4B MoveVar VAR,OLD FOLDER,NEW FOLDER
4C DropDown TITLE-STRING,{STRING1,...},VAR (note: {...} means list here)
4D Output ROW,COLUMN,EXPR
4E PtText STRING,X,Y
4F PxlText STRING,ROW,COLUMN
50 DrawSlp X,Y,SLOPE
51 Pause [EXPR]
52 Return [EXR]
53 Input [STRING,][VAR]
54 PlotsOff [1][,2] ... [,9]
55 PlotsOn [1][,2] ... [,9]
56 Title STRING[,LABEL]
57 Item STRING[,LABEL]
58 InputStr [STRING,]VAR
59 LineHorz Y[,DRAWMODE]
5A LineVert X[,DRAWMODE]
5B PxlHorz ROW[,DRAWMODE]
5C PxlVert COLUMN[,DRAWMODE]
5D AndPic PICVAR[,PXLROW,PXLCOL]
5E RclPic PICVAR[,PXLROW,PXLCOL]
5F RplcPic PICVAR[,PXLROW,PXLCOL]
60 XorPic PICVAR[,PXLROW,PXLCOL]
61 DrawPol EXPR[,_THETA_MIN][,_THETA_MAX][,_THETA_STEP]
62 Text STRING
63 OneVar L1[,L2][,L3][,L4]
64 StoPic VAR[,PXLROW,PXLCOL][,WIDTH,HEIGHT]
```

```
65 Graph EXPR1[,EXPR2][,VAR1][,VAR2]
66 Table EXPR1[,EXPR2][,VAR1]
67 NewPic MAT,PICVAR[,MAXROW][,MAXCOL]
68 DrawParm EXPR,EXPR[,TMIN,][,TMAX][,TSTEP]
69 CyclePic PICNAME-STRING,N[,WAIT],[CYCLES],[DIRECTION]
6A CubicReg L1,L2[,[L3][,L4,L5]]
6B ExpReg L1,L2[,[L3][,L4,L5]]
6C LinReg L1,L2[,[L3][,L4,L5]]
6D LnReg L1,L2[,[L3][,L4,L5]]
6E MedMed L1,L2[,[L3][,L4,L5]]
6F PowerReg L1,L2[,[L3][,L4,L5]]
70 QuadReg L1,L2[,[L3][,L4,L5]]
71 QuartReg L1,L2[,[L3][,L4,L5]]
72 TwoVar L1,L2[,[L3][,L4,L5]]
73 Shade EXPR1,EXPR2,[XLOW],[XHIGH],[PAT],[PATRES]
74 For VAR,LOW,HIGH[,STEP]
75 Circle X,Y,R[,DRAWMODE]
76 PxlCrcl ROW,COL,R[,DRAWMODE]
77 NewPlot N,TYPE,XLIST[,YLIST][,FRQ][,CAT][,INC][,MARK][,BUCKET]
78 Line XSTART,YSTART,XEND,YEND[,DRAWMODE]
79 PxlLine ROWSTART,COLSTART,ROWEND,COLEND[,DRAWMODE]
7A Disp [EXPR,...]
7B FnOff [1][,2] ... [,99]
7C FnOn [1][,2] ... [,99]
7D Local VAR1,...
7E DelFold VAR1,...
7F DelVar VAR1,...
80 Lock VAR1,...
81 Prompt VAR1,...
82 SortA LIST1,...
83 SortD LIST1,...
84 UnLock VAR1,...
85 NewData DATAVAR,LIST1,...
86 Define VAR(ARG1,...)=EXPR
87 Else (Try...Endtry) CONDITION
88 ClrErr
89 PassErr
8A CF_NEW DispHome
8B CF_NEW Exec STRING[,EXPR,...]
8C CF_NEW Archive VAR1,...
8D CF_NEW Unarchiv VAR1,...
8E CF_NEW LU MAT,LVAR,UVAR,PVAR[,TOL]
8F CF_NEW QR MAT,QVAR,RVAR[,TOL]
90 CF_NEW BldData(VAR|SYSDATA)
91 CF_NEW DrwCtour EXPR
92 CF_NEW NewProb
93 CF_NEW SinReg L1,L2[,ITER],[PERIOD][,L3,L4]]
94 CF_NEW Logistic L1,L2[,[ITER],[L3][,L4,L5]]
95 CF_NEW CustmOn
96 CF_NEW CustmOff
97 CF_NEW SendChat VAR
```

Enumerations

I have not required an enumeration for instruction tags and system variable tags as yet...So here are a few enumerations of the tags, the main and extended tags are complete.

```
enum TAGS {
  VAR_NAME_TAG = 0,
  _VAR_Q_TAG = 1,
  VAR_R_TAG = 2,
  VAR_S_TAG = 3,
  VAR_T_TAG = 4,
```

```
  VAR_U_TAG = 5,
  VAR_V_TAG = 6,
  VAR_W_TAG = 7,
  VAR_X_TAG = 8,
  VAR_Y_TAG = 9,
  VAR_Z_TAG = 0xA,
  VAR_A_TAG = 0xB,
  VAR_B_TAG = 0xC,
  VAR_C_TAG = 0xD,
  VAR_D_TAG = 0xE,
  VAR_E_TAG = 0xF,
  VAR_F_TAG = 0x10,
  VAR_G_TAG = 0x11,
  VAR_H_TAG = 0x12,
  VAR_I_TAG = 0x13,
  VAR_J_TAG = 0x14,
  VAR_K_TAG = 0x15,
  VAR_L_TAG = 0x16,
  VAR_M_TAG = 0x17,
  VAR_N_TAG = 0x18,
  VAR_O_TAG = 0x19,
  VAR_P_TAG = 0x1A,
  VAR_Q_TAG = 0x1B,
  VAR_SYSTEM_TAG = 0x1C,
  ARB_REAL_TAG = 0x1D,
  ARB_INT_TAG = 0x1E,
  POSINT_TAG = 0x1F,
  NEGINT_TAG = 0x20,
  POSFRAC_TAG = 0x21,
  NEGFRAC_TAG = 0x22,
  BCD_TAG = 0x23,
  PI_TAG = 0x24,
  EXP_TAG = 0x25,
  IM_TAG = 0x26,
  NEGINFINITY_TAG = 0x27,
  INFINITY_TAG = 0x28,
  PN_INFINITY_TAG = 0x29,
  UNDEF_TAG = 0x2A,
  FALSE_TAG = 0x2B,
  TRUE_TAG = 0x2C,
  STRING_TAG = 0x2D,
  NOTHING_TAG = 0x2E,
  ACOSH_TAG = 0x2F,
  ASINH_TAG = 0x30,
  ATANH_TAG = 0x31,

  COSH_TAG = 0x35,
  SINH_TAG = 0x36,
  TANH_TAG = 0x37,

  RACOS_TAG = 0x41,
  RASIN_TAG = 0x42,
  RATAN_TAG = 0x43,

  COS_TAG = 0x44,
  SIN_TAG = 0x45,
  TAN_TAG = 0x46,

  ITAN_TAG = 0x4A,
  ABS_TAG = 0x4B,
  ANGLE_TAG = 0x4C,
  CEILING_TAG = 0x4D,
  FLOOR_TAG = 0x4E,
  INT_TAG = 0x4F,
  SIGN_TAG = 0x50,
  SQRT_TAG = 0x51,
  EXPF_TAG = 0x52,
  LN_TAG = 0x53,
  LOG_TAG = 0x54,
```

```
FPART_TAG = 0x55,
IPART_TAG = 0x56,
CONJ_TAG = 0x57,
IMAG_TAG = 0x58,
REAL_TAG = 0x59,
APPROX_TAG = 0x5A,
TEXPAND_TAG = 0x5B,
TCOLLECT_TAG = 0x5C,
GETDENOM_TAG = 0x5D,
GETNUM_TAG = 0x5E,
CUMSUM_TAG = 0x60,
DET_TAG = 0x61,
COLNORM_TAG = 0x62,
ROWNORM_TAG = 0x63,
NORM_TAG = 0x64,
MEAN_TAG = 0x65,
MEDIAN_TAG = 0x66,
PRODUCT_TAG = 0x67,
STDDEV_TAG = 0x68,
SUM_TAG = 0x69,
VARIANCE_TAG = 0x6A,
UNITV_TAG = 0x6B,
DIM_TAG = 0x6C,
MAT2LIST_TAG = 0x6D,
NEWLIST_TAG = 0x6E,
RREF_TAG = 0x6F,
REF_TAG = 0x70,
IDENTITY_TAG = 0x71,
DIAG_TAG = 0x72,
COLDIM_TAG = 0x73,
ROWDIM_TAG = 0x74,
TRANSPOSE_TAG = 0x75,
FACTORIAL_TAG = 0x76,
PERCENT_TAG = 0x77,
RADIANS_TAG = 0x78,
NOT_TAG = 0x79,
MINUS_TAG = 0x7A,
VEC_POLAR_TAG = 0x7B,
VEC_CYLIND_TAG = 0x7C,
VEC_SPHERE_TAG = 0x7D,
START_TAG = 0x7E,
ISTORE_TAG = 0x7F,
STORE_TAG = 0x80,
WITH_TAG = 0x81,
XOR_TAG = 0x82,
OR_TAG = 0x83,
AND_TAG = 0x84,
LT_TAG = 0x85,
LE_TAG = 0x86,
EQ_TAG = 0x87,
GE_TAG = 0x88,
GT_TAG = 0x89,
NE_TAG = 0x8A,
ADD_TAG = 0x8B,
ADDELT_TAG = 0x8C,
SUB_TAG = 0x8D,
SUBELT_TAG = 0x8E,
MUL_TAG = 0x8F,
MULELT_TAG = 0x90,
DIV_TAG = 0x91,
DIVELT_TAG = 0x92,
POW_TAG = 0x93,
POWELT_TAG = 0x94,
SINCOS_TAG = 0x95,
SOLVE_TAG = 0x96,
CSOLVE_TAG = 0x97,
NSOLVE_TAG = 0x98,
ZEROS_TAG = 0x99,
CZEROS_TAG = 0x9A,
```

```
  FMIN_TAG = 0x9B,
  FMAX_TAG = 0x9C,
  COMPLEX_TAG = 0x9D,
  POLYEVAL_TAG = 0x9E,
  RANDPOLY_TAG = 0x9F,
  CROSSP_TAG = 0xA0,
  DOTP_TAG = 0xA1,
  GCD_TAG = 0xA2,
  LCM_TAG = 0xA3,
  MOD_TAG = 0xA4,
  INTDIV_TAG = 0xA5,
  REMAIN_TAG = 0xA6,
  NCR_TAG = 0xA7,
  NPR_TAG = 0xA8,
  P2RX_TAG = 0xA9,
  P2RY_TAG = 0xAA,
  P2PTHETA_TAG = 0xAB,
  P2PR_TAG = 0xAC,
  AUGMENT_TAG = 0xAD,
  NEWMAT_TAG = 0xAE,
  RANDMAT_TAG = 0xAF,
  SIMULT_TAG = 0xB0,
  PART_TAG = 0xB1,
  EXP2LIST_TAG = 0xB2,
  RANDNORM_TAG = 0xB3,
  MROW_TAG = 0xB4,
  ROWADD_TAG = 0xB5,
  ROWSWAP_TAG = 0xB6,
  ARCLEN_TAG = 0xB7,
  NINT_TAG = 0xB8,
  PI_PRODUCT_TAG = 0xB9,
  SIGMA_SUM_TAG = 0xBA,
  MROWADD_TAG = 0xBB,
  ANS_TAG = 0xBC,
  ENTRY_TAG = 0xBD,
  EXACT_TAG = 0xBE,
  LOGB_TAG = 0xBF,
  COMDENOM_TAG = 0xC0,
  EXPAND_TAG = 0xC1,
  FACTOR_TAG = 0xC2,
  CFACTOR_TAG = 0xC3,
  INTEGRATE_TAG = 0xC4,
  DIFFERENTIATE_TAG = 0xC5,
  AVGRC_TAG = 0xC6,
  NDERIV_TAG = 0xC7,
  TAYLOR_TAG = 0xC8,
  LIMIT_TAG = 0xC9,
  PROPFRAC_TAG = 0xCA,
  WHEN_TAG = 0xCB,
  ROUND_TAG = 0xCC,
  DMS_TAG = 0xCD,
  LEFT_TAG = 0xCE,
  RIGHT_TAG = 0xCF,
  MID_TAG = 0xD0,
  SHIFT_TAG = 0xD1,
  SEQ_TAG = 0xD2,
  LIST2MAT_TAG = 0xD3,
  SUBMAT_TAG = 0xD4,
  SUBSCRIPT_TAG = 0xD5,
  RAND_TAG = 0xD6,
  MIN_TAG = 0xD7,
  MAX_TAG = 0xD8,
  LIST_TAG = 0xD9,
  USERFUNC_TAG = 0xDA,
  MATRIX_TAG = 0xDB,
  FUNC_TAG = 0xDC,
  DATA_TAG = 0xDD,
  GDB_TAG = 0xDE,
  PIC_TAG = 0xDF,
```

```
  TEXT_TAG = 0xE0,
  FIG_TAG = 0xE1,
  MAC_TAG = 0xE2,
  EXT_TAG = 0xE3,
  EXT_INSTR_TAG = 0xE4,
  END_TAG = 0xE5,
  COMMENT_TAG = 0xE6,
  NEXTEXPR_TAG = 0xE7,
  NEWLINE_TAG = 0xE8,
  ENDSTACK_TAG = 0xE9,
  PN1_TAG = 0xEA,
  PN2_TAG = 0xEB,
  ERROR_MSG_TAG = 0xEC,
  EIGVC_TAG = 0xED,
  EIGVL_TAG = 0xEE,
  DASH_TAG = 0xEF,
  LOCALVAR_TAG = 0xF0,
  DESOLVE_TAG = 0xF1,
  FDASH_TAG = 0xF2,
  ASM_TAG = 0xF3,
  ISPRIME_TAG = 0xF4,
// Unknown 3
  OTH_TAG = 0xF8,
  ROTATE_TAG = 0xF9
};

enum EXT_TAGS {
  INDIR_TAG = 0x1,
  GETKEY_TAG = 0x2,
  GETFOLD_TAG = 0x3,
  SWITCH_TAG = 0x4,
  UNITCONV_TAG = 0x5,
  ORD_TAG = 0x6,
  EXPR_TAG = 0x7,
  CHAR_TAG = 0x8,
  STRING_TAG = 0x9,
  GETTYPE_TAG = 0xA,
  GETMODE_TAG = 0xB,
  SETFOLD_TAG = 0xC,
  PTTEST_TAG = 0xD,
  PXLTEST_TAG = 0xE,
  SETGRAPH_TAG = 0xF,
  SETTABLE_TAG = 0x10,
  SETMODE_TAG = 0x11,
  FORMAT_TAG = 0x12,
  INSTRING_TAG = 0x13,
  APPEND_TAG = 0x14,
  DMS_TAG = 0x15,
  EXPR2DMS_TAG = 0x16,
  VEC2RECT_TAG = 0x17,
  VEC2POLAR_TAG = 0x18,
  VEC2CYLIND_TAG = 0x19,
  VEC2SPHERE_TAG = 0x1A,
  PARENTH_START_TAG = 0x1B,
  PARENTH_END_TAG = 0x1C,
  MAT_START_TAG = 0x1D,
  MAT_END_TAG = 0x1E,
  LIST_START_TAG = 0x1F,
  LIST_END_TAG = 0x20,
  COMMA_TAG = 0x21,
  SEMICOLON_TAG = 0x22,
  COMPLEX_ANGLE_TAG = 0x23,
  DASH_TAG = 0x24,
  QUOTE_TAG = 0x25,
  ANGLE_TAG = 0x26,
  TMPCNV_TAG = 0x27,
  _DELTA_TMPCNV_TAG = 0x28,
  GETUNITS_TAG = 0x29,
  SETUNITS_TAG = 0x2A,
```

```
  BIN_TAG = 0x2B,
  HEX_TAG = 0x2C,
  INT2BIN_TAG = 0x2D,
  INT2DEC_TAG = 0x2E,
  INT2HEX_TAG = 0x2F,
  DET_TAG = 0x30,
  REF_TAG = 0x31,
  RREF_TAG = 0x32,
  SIMULT_TAG = 0x33,
  GETCONFG_TAG = 0x34,
  AUGMENT_TAG = 0x35
};

enum INSTRUCTION_TAGS {
//Haven't needed many
  DEFINE_ITAG = 0x86,
  IFTHEN_ITAG = 0x3B,
  ELSEIF_ITAG = 0x39
};
```

END OF FILE