

- ENSEIRB -

2^{ème} ANNEE ELECTRONIQUE



STAGE MICROPROCESSEUR 68000
- MANUEL -

Patrice KADIONIK
kadionik@enseirb.fr
www.enseirb.fr/~kadionik

TABLE DES MATIERES

1. COMMANDES DU MONITEUR DU KIT 68000 :	3
2. SOUS-PROGRAMMES DU MONITEUR :	5
3. MAPPING MEMOIRE DU KIT 68000 :	7
4. PRINCIPES DE FONCTIONNEMENT DE QUELQUES ELEMENTS :	8
4.1. LES AFFICHEURS 7 (+1) SEGMENTS :	8
4.2. LE CLAVIER :	9
4.3. LES INTERRUPTIONS :	9
5. DIRECTIVES D'ASSEMBLAGE - COMMANDES D'EDITION DE LIENS :	10
5.1. ASSEMBLER DIRECTIVES	10
5.2. LINKER COMMANDS	11
5. SCHEMA DE PRINCIPE :	12
6. FORMAT DES FICHIERS MOTOROLA S-RECORD ET INTEL :	13
6.1. FORMAT MOTOROLA	13
6.2. FORMAT INTEL	16

1. COMMANDES DU MONITEUR DU KIT 68000 :

- A@ DISPLAY/SET ADDRESS REGISTER (@=0..7)**
USAGE: A@ <cr> OR A@=<hexdata> <cr>
- AS LINE ASSEMBLER**
USAGE: AS <hexaddress> <cr>
- B@ DISPLAY/SET BREAKPOINT (@=0..7)**
USAGE: B@ <cr> OR B@=<hexaddress> <cr>
- BO BOOT CP/M OR PDOS OR OS-9 OPERATING SYSTEM**
USAGE: BO <cr>
- CA CALL A PROGRAM AT SPECIFIED ADDRESS**
USAGE: CA <hexaddress> <cr>
- CB CLEAR ALL BREAKPOINTS**
USAGE: CB <cr>
- CE ATTACH/DETACH CENTRONICS PRINTER**
USAGE: CE <cr>
- D@ DISPLAY/SET DATA REGISTER (@=0..7)**
USAGE: D@ <cr> OR D@=<hexdata> <cr>
- DB DISPLAY ALL BREAKPOINTS**
USAGE: DB <cr>
- DF DISPLAY/SET DESTINATION FUNCTION CODE REGISTER (FOR 68010 ONLY)**
USAGE: DF <cr> OR DF=<hexdata> <cr>
- DI LINE DISASSEMBLER**
USAGE: DI <startaddress> <endaddress> <cr>
- DT DISPLAY TIME**
USAGE: DT <cr>
- DU DUMP MEMORY**
USAGE: DU <startaddress> <endaddress> <cr>
- FI FILL MEMORY WITH BYTE, WORD, LONGWORD**
USAGE: FI.B(.W)(.L) <startaddress> <endaddress> <hexdata>
- GO START PROGRAM AT SPECIFIED ADDRESS**
USAGE: GO <hexaddress>
- HE DISPLAY HELP MESSAGE**
USAGE: HE <cr>
- HO ENTER HOST MODE**
USAGE: HO <exit character> <cr>
- IO DISPLAY SET ACTIVATED IO CHANNELS**
USAGE: IO <cr>
- LO LOAD S1/S2 RECORDS**
USAGE: LO <loadaddress>
- ML MEMORY LOCATE**
USAGE: ML <startaddress> <endaddress> <byte1>...<byteN>

MO MEMORY MOVE
USAGE: MO <startaddress> <endaddress> <dest. address>

MV MEMORY VERIFY
USAGE: MV <startaddress> <endaddress> <dest. address>

OP OPEN MEMORY
USAGE: OP <hexaddress>

PC DISPLAY/SET PROGRAM COUNTER
USAGE: PC <cr> OR PC=<hexaddress> <cr>

RT RAM TEST
USAGE: RT <startaddress> <endaddress> <cr>

RE DISPLAY CONTENTS OF ALL REGISTERS
USAGE: RE <cr>

RL DEFINE REGISTERS IN REGISTER LIST
USAGE: RL <cr>

SA SAVE MOTOROLA S1/S2 RECORDS
USAGE: SA <startaddress> <endaddress> <cr>

SF DISPLAY/SET * SOURCE FUNCTION CODE REGISTER (FOR 68010 ONLY)
USAGE: SF <cr> OR SF=<hexdata> <cr>

SR DISPLAY/SET STATUS REGISTER
USAGE: SR <cr> OR SR=<hexword>

SS DISPLAY/SET SYSTEM STACKPOINTER
USAGE: SS <cr> OR SS=<hexaddress>

ST SINGLE STEP
USAGE: ST <cr>

TC DISABLE FORMFEED DURING TRACE
USAGE: TC <cr>

TD ENABLE FORMFEED DURING TRACE
USAGE: TD <cr>

TR EXECUTE PROGRAMM IN TRACE MODE
USAGE: TR <cr>

TS SET REAL TIME CLOCK
USAGE: TS <cr>

TT CHANGE SIO PARAMETERS (BAUDRATE,PARITY...)
USAGE: TT <cr>

US DISPLAY/SET USER STACKPOINTER
USAGE: US <cr> OR US=<hexaddress>

VB DISPLAY/SET VECTOR BASE REGISTER (FOR 68010 ONLY)
USAGE: VB <cr> OR VB=<hexdata> <cr>

XP TURN XON/XOFF PROTOCOL ON
USAGE: XP <cr>

2. SOUS-PROGRAMMES DU MONITEUR :

Available Monitor Routines for the User

INCA

With this routine an ASCII character can be read in from a serial interface (1 - 2). The interface used is determined by the I/O flag. This routine must be called with JSR \$F50008 and the character is read in D0.B.

OUTCH

This is the corresponding output character routine. The interface used is also determined by the I/O flag. It must be present in the D0.B. The routine is called with JSR \$F5000C.

PCRLF

With this, a Carriage Return, Linefeed can be output on the interface determined by the I/O flag. The routine is called with JSR \$F50014 and transmits 1 \$0D and 1 \$0A as well as 6 \$00 (ZERO) to the selected port.

WARM

Upon JMP \$F50018, a return to the monitor takes place and sets PC, SP, SS, US and TPAPV to default value.

COLD

Upon JMP \$F5001C, a jump to the cold start address of the monitor results, the terminal baud rate must be reselected.

MLOOP

This essentially corresponds to the WARM routine but without change of any register. The call is with JMP \$F50024.

PDATA

With this routine several ASCII characters can be listed. The Address Register 5 must point to the address of the first character. An \$04 serves as an end of string identifier. The I/O flag again determines the interface. The call is with JSR \$F50020.

I/O Flag

The pointer to the I/O Flag is to be found here (\$F50010).

INTINP

This memory location (\$F50030) contains a pointer to the RAM address which contains the last entered character. If no character arrived, \$00 is in this RAM location.

Nom du sous-programme	INCA
Fonction	entrée d'un seul caractère ASCII par ligne série
Appel	JSR \$F50008
Paramètre à l'appel	aucun
Paramètre en retour	registre D0.B contient le caractère ASCII lu
Nom du sous-programme	OUTCH
Fonction	sortie d'un seul caractère ASCII par ligne série
Appel	JSR \$F5000C
Paramètre à l'appel	registre D0.B doit contenir le caractère ASCII
Paramètre en retour	aucun
Nom du sous-programme	PDATA
Fonction	sortie d'une chaîne de caractères ASCII la chaîne doit être en mémoire et nécessairement terminée par l'identificateur caractère fin de transmission EOT = \$04 (qui n'est pas transmis)
Appel	JSR \$F50020
Paramètre à l'appel	registre A5 pointe le premier caractère
Paramètre en retour	aucun
Nom de la fonction	TRAPIN
Fonction	entrée d'une chaîne de caractères ASCII, avec écho sur la ligne série de sortie; fin d'entrée par le caractère RC = \$0D (control M ou enter)
Appel	TRAP #15 DC.W 1
Paramètres à l'appel	registres A5 et A6 pointent sur l'adresse du début de la zone de mémorisation de la chaîne
Paramètre en retour	registre A6 pointe le dernier caractère lu
Nom de la fonction	TRAPOUT
Fonction	sortie d'une chaîne de caractères ASCII, suivie d'un saut et retour à la ligne (LF+RC)
Appel	TRAP #15 DC.W 2
Paramètres à l'appel	registre A5 pointe le premier caractère registre A6 pointe le dernier caractère + 1
Paramètre en retour	aucun
Nom de la fonction	MLOOP
Fonction	point d'entrée dans le moniteur sans aucune modification de la valeur des registres
Appel	JMP \$F50024
Paramètre à l'appel	aucun
Paramètre en retour	aucun

3. MAPPING MEMOIRE DU KIT 68000 :

Hex. Addresses	Description
:000000 - 000007	: Reset Vectors (EPROM)
:	:
:000008 - 0003FF	: Exception Vectors
:	:
:000400 - 03FFFF	: 256 Kbytes on-board DRAM
:000400 - 07FFFF	: 512 Kbytes on-board DRAM
:000400 - 0FFFFFFF	: 1024 Kbytes on-board DRAM
:	:
:040000 - F3FFFF	: VMEbus (with 256 Kbyte DRAM)
:080000 - F3FFFF	: VMEbus (with 512 Kbyte DRAM)
:100000 - F3FFFF	: VMEbus (with 1024 Kbyte DRAM)
:	:
:F40001 - F40039	: SIO 68564 Serial Interface Registers
:	:
:F40040 - F4007F	: FPP Busy
:	:
:F40081 - F4009B	: PIT 68230 Parallel Interface Registers
:F400AI - F400B5	: PIT 68230 Timer Registers
:	:
:F400C0 - F400FF	: FPP Floating Point Registers
:	:
:F40100 - F4013F	: System Fail Buffer
:	:
:	: F40181 : Address Mode Register
:	:
:F401CI - F401CF	: RTC 58167 Real Time Clock
:F401DI - F4010F	: RTC 58167 RAM
:F401EI - F401EB	: RTC 58167 Registers
:	:
:F40200 - F4023F	: MASTER MMU (Supervisor Mode only)
:F40240 - F4027F	: SLAVE MMU (Supervisor Mode only)
:	:
:F42000 - F43FFF	: I/O Module
:	:
:F50000 - F6FFFF	: EPROM
:	:
:F70000 - FEFFFF	: VMEbus
:	:
:FF0000 - FFFFFFFF	: Short I/O

4.2. LE CLAVIER :

Le clavier est constitué de 20 contacts répartis à l'intersection de 4 lignes et 5 colonnes, utilisant 9 entrées-sorties des ports A et B selon le câblage suivant :

PB0 ←	1	2	3	4	*
PB1 ←	5	6	7	8	#
PB2 ←	9	0	A	B	.
PB3 ←	C	D	E	F	->
	↑	↑	↑	↑	↑
	PA4	PA3	PA2	PA1	PA0

Le programme initialise le port A en sorties et le port B en entrées.
PADR contient \$ff après initialisation.

En l'absence de contact, les lignes sont reliées a un niveau haut ('1').
Afin de détecter le moindre appui (PB0-PB3), il est donc nécessaire de positionner une ou plusieurs colonnes (PA0-PA4) à un niveau bas ('0').

4.3. LES INTERRUPTIONS :

Rappel des niveaux d'interruption disponibles sur le kit :

- IRQ ABORT : autovecteur niveau 7 (poussoir Abort face avant)
- IRQ TIMER : vecteur niveau 6 (avec inter "timer" haut)
- IRQ RTC : autovecteur niveau 5
- IRQ Hx : vecteur niveau 4 (H1=poussoir, H3=clavier)

Interruption timer :

- validée et invalidée dans TCR.
- détection de l'interruption dans TSR.
- programmation de l'intervalle par CPRH, CPRM, CPRL (poids fort, moyen, faible).
durée=valeur programmée*32*100ns.

Interruption H1 :

- l'état de H1 est lu dans le bit 4 de PSR.
- H1=0->1 interruption sur un appui du poussoir.
- H1=1->0 interruption sur un relâchement du poussoir.

Interruption H3 :

- l'état de H3 est lu dans le bit 6 de PSR.
- H3=1->0 interruption par appui d'une touche clavier.

5. DIRECTIVES D'ASSEMBLAGE - COMMANDES D'EDITION DE LIENS :

5.1. Assembler Directives

Directive	Function
ALIGN	Specify Byte Alignment
CHIP	Specify Target Microprocessor
COMLINE	Define Storage
COMMON	Specify Common Section
DC	Define Constant Value
DCB	Define Constant Block
DS	Define Storage
ELSEC	Conditional Assembly Converse
END	End of Assembly
ENDC	End Conditional Assembly
ENDR	End Repeat
EQU	Equate a Symbol to an Expression
FAIL	Generate an Error
FEQU	Equate a Symbol to a Floating Point Expression
FOPT	Specify Floating Point Options
FORMAT	Format Listing
FOPT	Specify Floating Point Options for Assembly
IDNT	Specify Module Name
IFC	Conditional Assembly String Equality Test
IFEQ	Conditional Assembly Equal to Zero Test
IFGE	Conditional Assembly Nonnegative Test
IFGT	Conditional Assembly Greater than zero Test
IFLE	Conditional Assembly Nonpositive Test
IFLT	Conditional Assembly Less than Zero Test
IFNC	Conditional Assembly String Inequality Test
IFNE	Conditional Assembly Unequal to Zero Test
INCLUDE	Include Source File
IRP	Specify Indefinite Repeat
IRPC	Specify Indefinite Repeat Character
LIST	List the Assembly
LLEN	Set Length of Line in Assembler Listing
MASK2	Assemble for R9M Chip
NAME	Specify Module Name
NOFORMAT	Don't Format Listing
NOLIST	Don't List the Assembly
NOOBJ	Don't Create Object File
NOPAGE	Suppress Paging on Listing
OFFSET	Define Table of Offsets
OPT	Set Options for Assembly
ORG	Begin an Absolute Section

Directive

Function

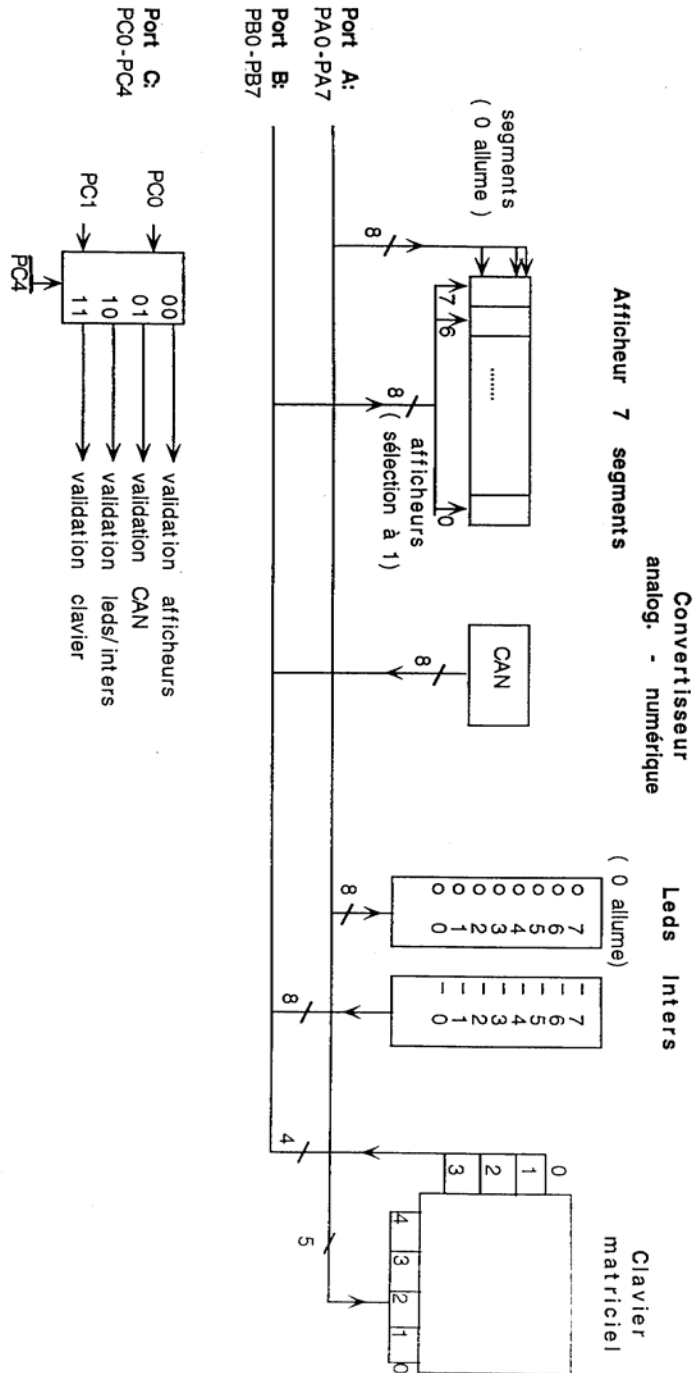
PAGE	Advance Listing Form to Next Page
PLEN	Set Length of Listing Page
REG	Define a Register List
REPT	Specify Repeat
SECT	Specify Section
SECTION	Specify Section
SET	Equate a Symbol to an Expression
SPC	Space lines on listing
TTL	Set Program Heading
XDEF	Specify External Definition
XREF	Specify External Reference

5.2. Linker Commands

Command	Function
ABSOLUTE	Specify Sections to Include in Absolute File
ALIAS	Specify Section Assumed Name
BASE	Specify Location at Which to Begin Loading
CHIP	Specify Target Microprocessor
Comment	Specify Comment
COMMON	Set Common Section Load Address
CPAGE	Set Paging for Common Section
END	End Command Stream and Finish Loading
EXIT	Exit Linker
FORMAT	Format Linker Output
IFILE	Store Information using Intermediate File
INDEX	Specify Run-time value of Register An
INITDATA	Specify Initialized Data in ROM
LIST	List Specified Elements
LOAD	Load Specified Object Modules
NAME	Specify Output Module Name
NOIFILE	Store Information using Virtual Memory
NLIST	Do not list Specified Elements
NOPAGE	Turn off Paging for Section
ORDER	Specify Long Section Order
PAGE	Set Paging for Noncommon Section
PUBLIC	Specify PUBLIC Symbols (External Definitions)
SECT	Set Noncommon Section Load Address
SORDER	Specify Short Section Order
START	Specify Output Module Starting Address

5. SCHEMA DE PRINCIPE :

Synoptique des manips du kit 6800 0



Port H:
 H2= bit 3 de PACR -> allume led si inter supérieur levé
 H4= bit 3 de FBCR -> valide huat-parleur si inter supérieur levé

6. FORMAT DES FICHIERS MOTOROLA S-RECORD ET INTEL :

6.1. Format MOTOROLA

An S-record file consists of a sequence of specially formatted ASCII character strings. An S-record will be less than or equal to 78 bytes in length.

The order of S-records within a file is of no significance and no particular order may be assumed.

The general format of an S-record follow:

```
+-----//-----//-----+
| type | count | address |          data          | checksum |
+-----//-----//-----+
```

type	A char[2] field. These characters describe the type of record (S0, S1, S2, S3, S5, S7, S8, or S9).
count	A char[2] field. These characters when paired and interpreted as a hexadecimal value, display the count of remaining character pairs in the record.
address	A char[4,6, or 8] field. These characters grouped and interpreted as a hexadecimal value, display the address at which the data field is to be loaded into memory. The length of the field depends on the number of bytes necessary to hold the address. A 2-byte address uses 4 characters, a 3-byte address uses 6 characters, and a 4-byte address uses 8 characters.
data	A char [0-64] field. These characters when paired and interpreted as hexadecimal values represent the memory loadable data or descriptive information.
checksum	A char[2] field. These characters when paired and interpreted as a hexadecimal value display the least significant byte of the ones complement of the sum of the byte values represented by the pairs of characters making up the count, the address, and the data fields.

Each record is terminated with a line feed. If any additional or different record terminator(s) or delay characters are needed during transmission to the target system it is the responsibility of the transmitting program to provide them.

S0 Record The type of record is 'S0' (0x5330). The address

field is unused and will be filled with zeros (0x0000). The header information within the data field is divided into the following subfields.

mname	is char[20] and is the module name.
ver	is char[2] and is the version number.
rev	is char[2] and is the revision number.
description	is char[0-36] and is a

text comment.

Each of the subfields is composed of ASCII bytes whose associated characters, when paired, represent one byte hexadecimal values in the case of the version and revision numbers, or represent the hexadecimal values of the ASCII characters comprising the module name and description.

- S1 Record The type of record field is 'S1' (0x5331). The address field is interpreted as a 2-byte address. The data field is composed of memory loadable data.
- S2 Record The type of record field is 'S2' (0x5332). The address field is interpreted as a 3-byte address. The data field is composed of memory loadable data.
- S3 Record The type of record field is 'S3' (0x5333). The address field is interpreted as a 4-byte address. The data field is composed of memory loadable data.
- S5 Record The type of record field is 'S5' (0x5335). The address field is interpreted as a 2-byte value and contains the count of S1, S2, and S3 records previously transmitted. There is no data field.
- S7 Record The type of record field is 'S7' (0x5337). The address field contains the starting execution address and is interpreted as 4-byte address. There is no data field.
- S8 Record The type of record field is 'S8' (0x5338). The address field contains the starting execution address and is interpreted as 3-byte address. There is no data field.
- S9 Record The type of record field is 'S9' (0x5339). The address field contains the starting execution address and is interpreted as 2-byte address. There is no data field.

EXAMPLE

Shown below is a typical S-record format file.

```
S00600004844521B
S1130000285F245F2212226A000424290008237C2A
S11300100002000800082629001853812341001813
S113002041E900084E42234300182342000824A952
S107003000144ED492
S5030004F8
S9030000FC
```

The file consists of one S0 record, four S1 records, one S5 record and an S9 record.

The S0 record is comprised as follows:

- S0 S-record type S0, indicating it is a header record.
- 06 Hexadecimal 06 (decimal 6), indicating that six character pairs (or ASCII bytes) follow.
- 00 00 Four character 2-byte address field, zeroes in this example.
- 48 ASCII H, D, and R - "HDR".
- 1B The checksum.

The first S1 record is comprised as follows:

- S1 S-record type S1, indicating it is a data record

to be loaded at a 2-byte address.

- 13 Hexadecimal 13 (decimal 19), indicating that nineteen character pairs, representing a 2 byte address, 16 bytes of binary data, and a 1 byte checksum, follow.
- 00 00 Four character 2-byte address field; hexadecimal address 0x0000, where the data which follows is to be loaded.
- 28 5F 24 5F 22 12 22 6A 00 04 24 29 00 08 23 7C Sixteen character pairs representing the actual binary data.
- 2A The checksum.

The second and third S1 records each contain 0x13 (19) character pairs and are ended with checksums of 13 and 52, respectively. The fourth S1 record contains 07 character pairs and has a checksum of 92.

The S5 record is comprised as follows:

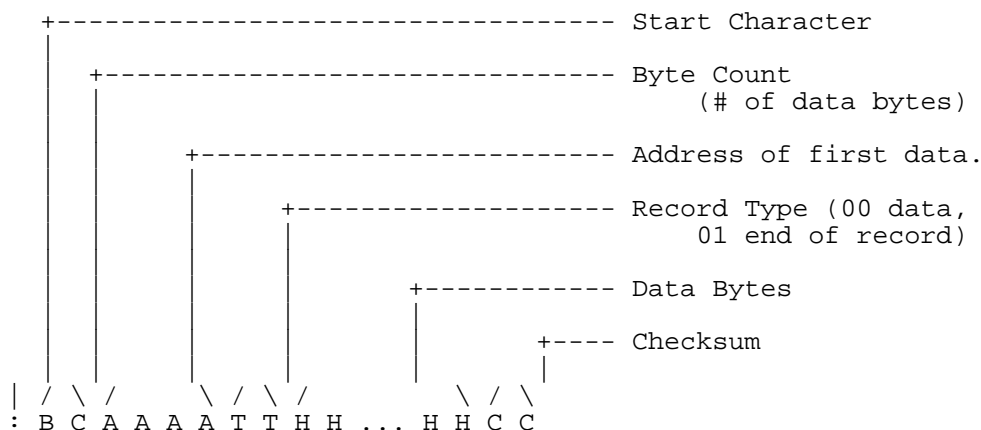
- S5 S-record type S5, indicating it is a count record indicating the number of S1 records.
- 03 Hexadecimal 03 (decimal 3), indicating that three character pairs follow.
- 00 04 Hexadecimal 0004 (decimal 4), indicating that there are four data records previous to this record.
- F8 The checksum.

The S9 record is comprised as follows:

- S9 S-record type S9, indicating it is a termination record.
- 03 Hexadecimal 03 (decimal 3), indicating that three character pairs follow.
- 00 00 The address field, hexadecimal 0 (decimal 0) indicating the starting execution address.
- FC The checksum.

6.2. Format INTEL

Intel HEX-ASCII format takes the form:



An examples:

```
:10000000DB00E60F5F1600211100197ED300C3004C
:1000100000000101030307070F0F1F1F3F3F7F7FF2
:01002000FFE0
:00000001FF
```

This information comes from *_Microprocessors and Programmed Logic_*, Second Edition, Kenneth L. Short, 1987, Prentice-Hall, ISBN 0-13-580606-2.

Provisions have been made for data spaces larger than 64 kBytes. The above reference does not discuss them. I suspect there is a start of segment type record, but I do not know how it is implemented.