PRZEMYSŁOWY INSTYTUT AUTOMATYKI I POMIARÓW
MERA-PIAP
Al. Jerozolimskie 202 02-222 Warszawa Telefon 23-70-81

ZESPÓŁ URZĄDZEŃ I SYSTEMÓW STEROWANIA

440 BE 10

Główny wykonawca dr inż. Wiesław Stańczak

Wykonawcy

Konsultant dr inż. Andrzej Syryczyński

Nr zlecenia S 1340

Opracowanie pakietu oprogramowania wspomagającego korzystanie z mikroprocesorów firmy MOTOROLA w laboratoriach Instytutu

Zleceniodawca praca statutowa

Pracę rozpoczęto dnia 93.01.15

Kierownik Zespołu

dr inż. A. Syryczyński

zakończono dnia 93.02.26

Z-ca Dyrektora d/s Badawczo Rozwojowych

dr inż. J. Jabłkowski

Praca zawiera:

stron 3

rysunków

fotografii

tabel

tablic

załączników 1

Nr rejestr. 6939

Rozdzielnik - ilość egz: 5

Egz. 1 BOINTE

Egz. 2 ORC

Egz. 3 POS

Egz. 4 ZAE

Egz. 5 ZSS

Egz. 6

Analiza deskryptorowa

ASEMBLACJA + KOMPILACJA + KOD WYNIKOWY + MIKROPROCESORY + MIKROSTEROWNIKI

Analiza dokumentacyjna

Opracowanie pakietu oprogramowania wspomagającego korzystanie z mikroprocesorów firmy MOTOROLA w laboratoriach Instytutu. Sprawozdanie obejmuje wykaz zawartości pakietów oprogramowania dla mikroprocesorów rodziny 6800, 68HC11 i 68000 firmy MOTOROLA wraz z uwagami dotyczącymi ich przydatności.

Tytuły poprzednich sprawozdań

brak

UKD

PIAP 41/88 10000

1. Geneza pracy

W roku 1992 w ramach realizacji jednego z etapów zlec. S1303 była prowadzona współpraca z firmą MOTOROLA. W trakcie tej współpracy MOTOROLA BDEE (Business Development Eastern Europe) nawiązało bezpośredni kontakt z PIAP oferując nieodpłatnie Instytutowi w wersji źródłowej zbiór skrośnych asemblerów, obejmujący mikroprocesory z rodziny: MC6800 / MC6802, MC6801, MC6804, MC6805 / MC68HC05, MC6809, MC68HC11 i MC68000 / MC68020 / MC68881 / MC68851, a także 2 skrośne kompilatory języka C (dedykowane na MC68HC11 oraz MC68000). Powyższe oprogramowanie zostało przekazane PIAP w grudniu 1992 r. wraz z życzeniem jego szerokiego rozpowszechnienia w laboratoriach PIAPu. Należy sądzić, że spełnienie tego życzenia będzie rzutować na przebieg dalszej współpracy.

Wstępne rozeznanie wskazało, że przekazane Instytutowi źródła oprogramowania przystosowane były do kompilacji przy użyciu produktów firmy Borland, których licencjonowanej wersji PIAP nie posiada. W związku z tym zaszła konieczność weryfikacji poszczególnych plików w celu ich przystosowania do obowiązujących standardów języka C, co jest równoznaczne z przysposobieniem ich do przetworzenia przy wykorzystaniu kompilatora C firmy Microsoft wer. 6.00A (PIAP zakupił licencjonowaną wersję tego oprogramowania).

2. Wyniki pracy

Zarysowane w p. 1 zadanie zrealizowano tworząc następujące pakiety oprogramowania:

- MC6800 / MC6802 zawierający następujące pliki:

A80.EXE	- asembler;
ASEMBLER.DOC	- opis asemblera (kopia oryginału);
T0.S	- przykładowy plik źródłowy;
T0.LST	- listing pliku T0.S;
T0.S19	- plik wynikowy w formacie S-Record przeznaczony do zapisania w EPROMie (opis formatu S-Record zamieszczono w załączniku);

3

- MC6801 zawierający następujące pliki:

AS1.EXE - asembler;
ASEMBLER.DOC - opis asemblera (kopia oryginału);
T1.S - przykładowy plik źródłowy;
T1.LST - listing pliku T1.S;
T1.S19 - plik wynikowy w formacie S-Record przeznaczony do zapisania w EPROMie;

- MC6804 zawierający następujące pliki:

AS4.EXE - asembler;
ASEMBLER.DOC - opis asemblera (kopia oryginału);
T4.S - przykładowy plik źródłowy;
T4.LST - listing pliku T4.S;
T4.S19 - plik wynikowy w formacie S-Record przeznaczony do zapisania w EPROMie;

- MC6805 / MC68HC05 zawierający następujące pliki:

AS5.EXE - asembler;
ASEMBLER.DOC - opis asemblera (kopia oryginału);
T5.S - przykładowy plik źródłowy;
T5.LST - listing pliku T5.S;
T5.S19 - plik wynikowy w formacie S-Record przeznaczony do zapisania w EPROMie;

- MC6809 zawierający następujące pliki:

AS9.EXE - asembler;
ASEMBLER.DOC - opis asemblera (kopia oryginału);
T9.S - przykładowy plik źródłowy;
T9.LST - listing pliku T9.S;
T9.S19 - plik wynikowy w formacie S-Record przeznaczony do zapisania w EPROMie;

- MC68HC11 zawierający następujące pliki:

AS11.EXE - asembler;
ASEMBLER.DOC - opis asemblera (kopia oryginału);
T11.S - przykładowy plik źródłowy;
T11.LST - listing pliku T11.S;
T11.S19 - plik wynikowy w formacie S-Record przeznaczony do zapisania w EPROMie;

4

- MC68000 / MC68020 / MC68881 / MC68851 zawierający następujące pl

A68K.EXE - asembler dla MC68000 / MC68020 / MC68881 / MC68851;

C68K.EXE - kompilator języka C dla MC68000;

CASE20.S - przykładowy plik źródłowy;

CASE20.LST - listing pliku CASE20.S;

CASE20.S19 - plik wynikowy w formacie S-Record przeznaczony do zapisania w EPROMie;

SIEVE.C - przykładowy plik źródłowy w języku C;

SIEVE.LIS - listing pliku SIEVE.C;

SIEVE.S - przykładowy plik źródłowy w asemblerze uzyskany z SIEVE.C;

SIEVE.LST - listing pliku SIEVE.S;

SIEVE.S19 - plik wynikowy w formacie S-Record przeznaczony do zapisania w EPROMie;

README.X68 - opis pakietu (kopia oryginału);

README.A68 - opis asemblera (kopia oryginału);

README.C68 - opis kompilatora (kopia oryginału);

- MC68HC11 zawierający następujące pliki:

SC11.EXE - kompilator języka C dla MC68HC11;

SIEVE.C - przykładowy plik źródłowy;

SIEVE.ASM - wynik kompilacji pliku SIEVE.C.

Format wywołania kompilatora SC11.EXE wyświetla się na ekranie po podaniu komendy:

SC11 -?

Stwierdzono małą przydatność ostatniego z pakietów. Mianowicie wymaga on znacznych przeróbek, żeby mógł współpracować z asemblerem AS11.EXE, a sam generuje wyłącznie mnemoniki instrukcji asemblerowych. Może więc jedynie służyć jako narzędzie uzupełniające - do tworzenia wstępnego kodu dla AS11.EXE, który jednakże potem powinien zostać przerobiony przez programistę.

Dużo większe nadzieje należy wiązać z A68K.EXE. Przeprowadzone eksperymenty wykazały, że można go bez trudu przerobić na asembler dla mikroprocesora MC68HC16. Ponadto wydaje się, że uda się, po pewnych modyfikacjach przerobić go do stosowania w przypadku MC68030, MC68040 oraz całej rodziny 68300.

ZAŁĄCZNIK

APPENDIX C
S-RECORD OUTPUT FORMAT

The S-record format for output modules is for encoding programs or data files in a printable format for transportation between computer systems. The transportation process can be visually monitored, and the S-records can be easily edited.

C.1 S-RECORD CONTENT

Visually, S-records are essentially character strings made of several fields that identify the record type, record length, memory address, code/data, and checksum. Each byte of binary data encodes as a two-character hexadecimal number: the first character represents the high-order four bits, and the second character represents the low-order four bits of the byte. Figure C-1 illustrates the five fields that comprise an S-record. Table C-1 lists the composition of each S-record field.

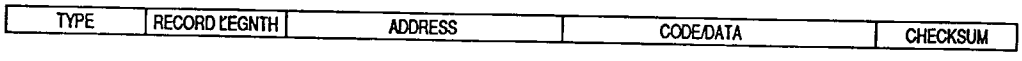

Figure C-1. Five Fields of an S-Record

Table C-1. Field Composition of an S-Record

Field	Printable Characters	Contents
Type	2	S-record type—S0, S1, etc.
Record Length	2	The count of the character pairs in the record, excluding the type and record length.
Address	4, 6, or 8	The 2-, 3-, or 4-byte address at which the data field is to be loaded into memory.
Code/Data	0–2n	From 0 to n bytes of executable code, memory loadable data, or descriptive information. For compatibility with teletypewriters, some programs may limit the number of bytes to as few as 28 (56 printable characters in the S-record).
Checksum	2	The least significant byte of the one's complement of the sum of the values represented by the pairs of characters making up the record length, address, and the code/data fields.

When downloading S-records, each must be terminated with a CR. Additionally, an S-record may have an initial field that fits other data such as line numbers generated by some time-sharing systems. The record length (byte count) and checksum fields ensure transmission accuracy.

C.2 S-RECORD TYPES

There are eight types of S-records to accommodate the encoding, transportation, and decoding functions. The various Motorola record transportation control programs (e.g. upload, download, etc.), cross assemblers, linkers, and other file creating or debugging programs, only utilize S-records serving the program's purpose. For more information on support of specific S-records, refer to the user's manual for that program.

An S-record format module may contain S-records of the following types:

- S0— The header record for each block of S-records. The code/data field may contain any descriptive information identifying the following block of S-records. Under VERSAdos, the resident linker's IDENT command can be used to designate module name, version number, revision number, and description information that will make up the header record. The address field is normally zeros.
- S1— A record containing code/data and the 2-byte address at which the code/data is to reside.
- S2— A record containing code/data and the 3-byte address at which the code/data is to reside.
- S3— A record containing code/data and the 4-byte address at which the code/data is to reside.
- S5— A record containing the number of S1, S2, and S3 records transmitted in a particular block. This count appears in the address field. There is no code/data field.
- S7— A termination record for a block of S3 records. The address field may optionally contain the 4-byte address of the instruction to which control is to be passed. There is no code/data field.
- S8— A termination record for a block of S2 records. The address field may optionally contain the 3-byte address of the instruction to which control is to be passed. There is no code/data field.
- S9— A termination record for a block of S1 records. The address field may optionally contain the 2-byte address of the instruction to which control is to be passed. Under VERSAdos, the resident linker's ENTRY command can be used to specify this address. If this address is not specified, the first entry point specification encountered in the object module input will be used. There is no code/data field.

Each block of S-records uses only one termination record. S7 and S8 records are only active when control is to be passed to a 3- or 4-byte address; otherwise, an S9 is used for termination. Normally, there is only one header record, although it is possible for multiple header records to occur.

C.3 S-RECORD CREATION

Dump utilities, debuggers, a VERSAdos resident linkage editor, or cross assemblers and linkers produce S-record format programs. On VERSAdos systems, the build load module (MBLM) utility allows an executable load module to be built from S-records. It has a counterpart utility in BUILDS that allows an S-record file to be created from a load module.

Programs are available for downloading or uploading a file in S-record format from a host system to an 8- or 16-bit microprocessor-based system. A typical S-record-format module is printed or displayed as follows:

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

The module has an S0 record, four S1 records, and an S9 record. The following character pairs comprise the S-record-format module.

S0 Record:

S0—S-record type S0, indicating that it is a header record.
06—Hexadecimal 06 (decimal 6), indicating that six character pairs (or ASCII bytes) follow.
0000—A 4-character, 2-byte address field; zeros in this example.
48—ASCII H
44—ASCII D
52—ASCII R
1B—The checksum.

First S1 Record:

S1—S-record type S1, indicating that it is a code/data record to be loaded/verified at a 2-byte address.
13—Hexadecimal 13 (decimal 19), indicating that 19 character pairs, representing 19 bytes of binary data, follow.
0000—A 4-character, 2-byte address field (hexadecimal address 0000) indicating where the data that follows is to be loaded.

The next 16 character pairs of the first S1 record are the ASCII bytes of the actual program code/data. In this assembly language example, the program's hexadecimal opcodes are sequentially written in the code/data fields of the S1 records.

Opcode	Instruction	
285F	MOVE.L	(A7) +, A4
245F	MOVE.L	(A7) +, A2
2212	MOVE.L	(A2), D1
226A0004	MOVE.L	4(A2), A1
24290008	MOVE.L	FUNCTION(A1), D2
237C	MOVE.L	#FORCEFUNC, FUNCTION(A1)

The rest of this code continues in the remaining S1 record's code/data fields and stores in memory location 0010, etc.

2A—The checksum of the first S1 record.

The second and third S1 records also contain hexadecimal 13 (decimal 19) character pairs and end with checksums 13 and 52, respectively. The fourth S1 record contains 07 character pairs and has a checksum of 92.

S9 Record:

S9—S-record type S9, indicating that it is a termination record.

03—Hexadecimal 03, indicating that three character pairs (3 bytes) follow.

0000—The address field, zeros.

FC—The checksum of the S9 record.

Each printable character in an S-record encodes in hexadecimal (ASCII in this example) representation of the binary bits that transmit. Figure C-2 illustrates the sending of the first S1 record. Table C-2 lists the ASCII code for S-records.

TYPE				RECORD LENGTH				ADDRESS								CODE/DATA								CHECKSUM			
S		1		1		3		0		0		0		0		2		8		5		F	****	2		A	
5	3	3	1	3	1	3	3	3	0	3	0	3	0	3	0	3	2	3	8	3	5	4	6	****	3	2	4	1
0101	0011	0011	0001	0011	0001	0011	0011	0011	0000	0011	0000	0011	0000	0011	0000	0011	0010	0011	1000	0011	0101	0100	0110	****	0011	0010	0100	0001

Figure C-2. Transmission of an S1 Record

Table C-2. ASCII Code

Least Significant Digit	Most Significant Digit							
	0	1	2	3	4	5	6	7
0	NUL	DLE	SP	0	@	P	`	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	¦
D	CR	GS	–	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	DEL