UNITED NATIONS EDUCATIONAL, SCIENTIFIC AND CULTURAL ORGANIZATION
INTERNATIONAL CENTRE FOR THEORETICAL PHYSICS
I.C.T.P., P.O. BOX 586, 34100 TRIESTE, ITALY, CABLE: CENTRATOM TRIESTE

UNITED NATIONS INDUSTRIAL DEVELOPMENT ORGANIZATION

INTERNATIONAL CENTRE FOR SCIENCE AND HIGH TECHNOLOGY
c/o INTERNATIONAL CENTRE FOR THEORETICAL PHYSICS 34100 TRIESTE (ITALY) VIA GRIGNANO, 9 (ADRIATICO PALACE) P.O. BOX 586 TELEPHONE 040-224572 TELEFAX 040-224575 TELEX 460449 APH I

SMR/643 - 7

SECOND COLLEGE ON
MICROPROCESSOR-BASED REAL-TIME CONTROL -
PRINCIPLES AND APPLICATIONS IN PHYSICS
5 - 30 October 1992

RECALL OF THE M6809 ROSY JUNIOR
&
Assembly Language Programming

(Part 2)

C.S. ANG
Department of Electrical Engineering
University of Malaya
Kuala Lumpur
Malaysia

Recall of the M6809, ROSY Junior &

Assembly Language Programming

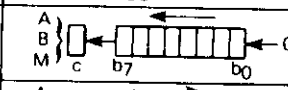
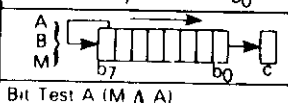
C S Ang
University of Malaya
Kuala Lumpur
Malaysia

Assembly Language Programs

- **Machine language** programs (computer's native tongue) are tedious, error prone, time consuming and inconvenient for people.
- **Assembly language** programs use descriptive mnemonics and symbolic names to ease the problem.
- Many other **assembly directives** are also added to make the task of programming easier.
- However, it is still more tedious than most high level language, including C! Thus, we use assembly language programs in our College only *sparingly*, when it is absolutely necessary.
- We shall recall M6809 assembly language programming so that we can understand and write **Device Descriptor** codes and **Device Drivers** in our experiments. (*The purists will tell you that even that should be done in C!?*)

M6809 Instruction Set Summary

Table D-1. Programming Aid (Continued)

Instruction	Forms	Addressing Modes: Immediate Op	Immediate ~	Immediate #	Direct Op	Direct ~	Direct #	Indexed Op	Indexed ~	Indexed #	Extended Op	Extended ~	Extended #	Inherent Op	Inherent ~	Inherent #	Description	5 H	3 N	2 Z	1 V	0 C
ABX														3A	3	1	B + X → X (Unsigned)	•	•	•	•	•
ADC	ADCA	89	2	2	99	4	2	A9	4+	2+	B9	5	3				A + M + C → A	↕	↕	↕	↕	↕
	ADCB	C9	2	2	D9	4	2	E9	4+	2+	F9	5	3				B + M + C → B	↕	↕	↕	↕	↕
ADD	ADDA	8B	2	2	9B	4	2	AB	4+	2+	BB	5	3				A + M → A	↕	↕	↕	↕	↕
	ADDB	CB	2	2	DB	4	2	EB	4+	2+	FB	5	3				B + M → B	↕	↕	↕	↕	↕
	ADDD	C3	4	3	D3	6	2	E3	6+	2+	F3	7	3				D + M:M + 1 → D	•	↕	↕	↕	↕
AND	ANDA	84	2	2	94	4	2	A4	4+	2+	B4	5	3				A ∧ M → A	•	↕	↕	0	•
	ANDB	C4	2	2	D4	4	2	E4	4+	2+	F4	5	3				B ∧ M → B	•	↕	↕	0	•
	ANDCC	1C	3	2													CC ∧ IMM → CC					?
ASL	ASLA													48	2	1		8	↕	↕	↕	↕
	ASLB													58	2	1		8	↕	↕	↕	↕
	ASL				08	6	2	68	6+	2+	78	7	3					8	↕	↕	↕	↕
ASR	ASRB													47	2	1		8	↕	↕	•	↕
	ASR													57	2	1		8	↕	↕	•	↕
	ASR				07	6	2	67	6+	2+	77	7	3					8	↕	↕	•	↕
BIT	BITA	85	2	2	95	4	2	A5	4+	2+	B5	5	3				Bit Test A (M ∧ A)	•	↕	↕	0	•
	BITB	C5	2	2	D5	4	2	E5	4+	2+	F5	5	3				Bit Test B (M ∧ B)	•	↕	↕	0	•
CLR	CLRA													4F	2	1	0 → A	•	0	1	0	0
	CLRB													5F	2	1	0 → B	•	0	1	0	0
	CLR				0F	6	2	6F	6+	2+	7F	7	3				0 → M	•	0	1	0	0
CMP	CMPA	81	2	2	91	4	2	A1	4+	2+	B1	5	3				Compare M from A	↕	↕	↕	↕	↕
	CMPB	C1	2	2	D1	4	2	E1	4+	2+	F1	5	3				Compare M from B	↕	↕	↕	↕	↕
	CMPD	10	5	4	10	7	3	10	7+	3+	10	8	4				Compare M:M + 1 from D	•	↕	↕	↕	↕
		83			93			A3			B3											
	CMPS	11	5	4	11	7	3	11	7+	3+	11	8	4				Compare M:M + 1 from S	•	↕	↕	↕	↕
		8C			9C			AC			BC											
	CMPU	11	5	4	11	7	3	11	7+	3+	11	8	4				Compare M:M + 1 from U	•	↕	↕	↕	↕
		83			93			A3			B3											
	CMPX	8C	4	3	9C	6	2	AC	6+	2+	BC	7	3				Compare M:M + 1 from X	•	↕	↕	↕	↕
	CMPY	10	5	4	10	7	3	10	7+	3+	10	8	4				Compare M:M + 1 from Y	•	↕	↕	↕	↕
		8C			9C			AC			BC											
COM	COMA													43	2	1	$\overline{A} \rightarrow A$	•	↕	↕	0	1
	COMB													53	2	1	$\overline{B} \rightarrow B$	•	↕	↕	0	1
	COM				03	6	2	63	6+	2+	73	7	3				$\overline{M} \rightarrow M$	•	↕	↕	0	1
CWAI		3C	≥20	2													CC ∧ IMM → CC Wait for Interrupt					7
DAA														19	2	1	Decimal Adjust A	•	↕	↕	0	↕
DEC	DECA													4A	2	1	A − 1 → A	•	↕	↕	↕	•
	DECB													5A	2	1	B − 1 → B	•	↕	↕	↕	•
	DEC				0A	6	2	6A	6+	2+	7A	7	3				M − 1 → M	•	↕	↕	↕	•
EOR	EORA	88	2	2	98	4	2	A8	4+	2+	B8	5	3				A ⊻ M → A	•	↕	↕	0	•
	EORB	C8	2	2	D8	4	2	E8	4+	2+	F8	5	3				B ⊻ M → B	•	↕	↕	0	•
EXG	R1, R2	1E	8	2													R1 → R2[2]	•	•	•	•	•
INC	INCA													4C	2	1	A + 1 → A	•	↕	↕	↕	•
	INCB													5C	2	1	B + 1 → B	•	↕	↕	↕	•
	INC				0C	6	2	6C	6+	2+	7C	7	3				M + 1 → M	•	↕	↕	↕	•
JMP					0E	3	2	6E	3+	2+	7E	4	3				EA[3] → PC	•	•	•	•	•
JSR					9D	7	2	AD	7+	2+	BD	8	3				Jump to Subroutine	•	•	•	•	•
LD	LDA	86	2	2	96	4	2	A6	4+	2+	B6	5	3				M → A	•	↕	↕	0	•
	LDB	C6	2	2	D6	4	2	E6	4+	2+	F6	5	3				M → B	•	↕	↕	0	•
	LDD	CC	3	3	DC	5	2	EC	5+	2+	FC	6	3				M:M + 1 → D	•	↕	↕	0	•
	LDS	10	4	4	10	6	3	10	6+	3+	10	7	4				M:M + 1 → S	•	↕	↕	0	•
		CE			DE			EE			FE											
	LDU	CE	3	3	DE	5	2	EE	5+	2+	FE	6	3				M:M + 1 → U	•	↕	↕	0	•
	LDX	8E	3	3	9E	5	2	AE	5+	2+	BE	6	3				M:M + 1 → X	•	↕	↕	0	•
	LDY	10	4	4	10	6	3	10	6+	3+	10	7	4				M:M + 1 → Y	•	↕	↕	0	•
		8E			9E			AE			BE											
LEA	LEAS							32	4+	2+							EA[3] → S	•	•	•	•	•
	LEAU							33	4+	2+							EA[3] → U	•	•	•	•	•
	LEAX							30	4+	2+							EA[3] → X	•	•	↕	•	•
	LEAY							31	4+	2+							EA[3] → Y	•	•	↕	•	•

Legend:

- OP Operation Code (Hexadecimal)
- ~ Number of MPU Cycles
- # Number of Program Bytes
- + Arithmetic Plus
- − Arithmetic Minus
- • Multiply
- $\overline{M}$ Complement of M
- → Transfer Into
- H Half-carry (from bit 3)
- N Negative (sign bit)
- Z Zero (Reset)
- V Overflow, 2's complement
- C Carry from ALU
- ↕ Test and set if true, cleared otherwise
- • Not Affected
- CC Condition Code Register
- : Concatenation
- V Logical or
- ∧ Logical and
- ⊻ Logical Exclusive or

M6809 Instruction Set Summary

Table D-1. Programming Aid (Continued)

Instruction	Forms	Addressing Modes: Immediate Op	Immediate ~	Immediate #	Direct Op	Direct ~	Direct #	Indexed[1] Op	Indexed ~	Indexed #	Extended Op	Extended ~	Extended #	Inherent Op	Inherent ~	Inherent #	Description	5 H	3 N	2 Z	1 V	0 C
LSL	LSLA													48	2	1	A ← 0	•	↕	↕	↕	↕
	LSLB													58	2	1	B ← 0	•	↕	↕	↕	↕
	LSL				08	6	2	68	6+	2+	78	7	3				M ← 0	•	↕	↕	↕	↕
LSR	LSRA													44	2	1	A → 0	•	0	↕	•	↕
	LSRB													54	2	1	B → 0	•	0	↕	•	↕
	LSR				04	6	2	64	6+	2+	74		3				M → 0	•	0	↕	•	↕
MUL														3D	11	1	A × B → D (Unsigned)	•	•	↕	•	9
NEG	NEGA													40	2	1	$\overline{A}$ + 1 → A	8	↕	↕	↕	↕
	NEGB													50	2	1	$\overline{B}$ + 1 → B	8	↕	↕	↕	↕
	NEG				00	6	2	60	6+	2+	70	7	3				$\overline{M}$ + 1 → M	8	↕	↕	↕	↕
NOP														12	2	1	No Operation	•	•	•	•	•
OR	ORA	8A	2	2	9A	4	2	AA	4+	2+	BA	5	3				A V M → A	•	↕	↕	0	•
	ORB	CA	2	2	DA	4	2	EA	+	2+	FA	5	3				B V M → B	•	↕	↕	0	•
	ORCC	1A	3	2													CC V IMM → CC				7	
PSH	PSHS	34	5+[4]	2													Push Registers on S Stack	•	•	•	•	•
	PSHU	36	5+[4]	2													Push Registers on U Stack	•	•	•	•	•
PUL	PULS	35	5+[4]	2													Pull Registers from S Stack	•	•	•	•	•
	PULU	37	5+[4]	2													Pull Registers from U Stack	•	•	•	•	•
ROL	ROLA													49	2	1	A ← C	•	↕	↕	↕	↕
	ROLB													59	2	1	B ← C	•	↕	↕	↕	↕
	ROL				09	6	2	69	6+	2+	79	7	3				M ← C	•	↕	↕	↕	↕
ROR	RORA													46	2	1	A → C	•	↕	↕	•	↕
	RORB													56	2	1	B → C	•	↕	↕	•	↕
	ROR				06	6	2	66	6+	2+	76	7	3				M → C	•	↕	↕	•	↕
RTI														3B	6/15	1	Return From Interrupt					/
RTS														39	5	1	Return from Subroutine	•	•	•	•	•
SBC	SBCA	82	2	2	92	4	2	A2	4+	2+	B2	5	3				A − M − C → A	8	↕	↕	↕	↕
	SBCB	C2	2	2	D2	4	2	E2	4+	2+	F2	5	3				B − M − C → B	8	↕	↕	↕	↕
SEX														1D	2	1	Sign Extend B into A	•	↕	↕	0	•
ST	STA				97	4	2	A7	4+	2+	B7	5	3				A → M	•	↕	↕	0	•
	STB[8]				D7	4	2	E7	4+	2+	F7	5	3				B → M	•	↕	↕	0	•
	STD				DD	5	2	ED	5+	2+	FD	6	3				D → M M + 1	•	↕	↕	0	•
	STS				10 DF	6	3	10 EF	6+	3+	10 FF	7	4				S → M M + 1	•	↕	↕	0	•
	STU				DF	5	2	EF	5+	2+	FF	6	3				U → M M + 1	•	↕	↕	0	•
	STX				9F	5	2	AF	5+	2+	BF	6	3				X → M M + 1	•	↕	↕	0	•
	STY				10 9F	6	3	10 AF	6+	3+	10 BF	7	4				Y → M M + 1	•	↕	↕	0	•
SUB	SUBA	80	2	2	90	4	2	A0	4+	2+	B0	5	3				A − M → A	8	↕	↕	↕	↕
	SUBB	C0	2	2	D0	4	2	E0	4+	2+	F0	5	3				B − M → B	8	↕	↕	↕	↕
	SUBD	83	4	3	93	6	2	A3	6+	2+	B3	7	3				D − M M + 1 → D	•	↕	↕	↕	↕
SWI	SWI[6]													3F	19	1	Software Interrupt 1	•	•	•	•	•
	SWI2[6]													10 3F	20	2	Software Interrupt 2	•	•	•	•	•
	SWI3[6]													11 3F	20	1	Software Interrupt 3	•	•	•	•	•
SYNC														13	≥4	1	Synchronize to Interrupt	•	•	•	•	•
TFR	R1, R2	1F	6	2													R1 → R2[2]	•	•	•	•	•
TST	TSTA													4D	2	1	Test A	•	↕	↕	0	•
	TSTB													5D	2	1	Test B	•	↕	↕	0	•
	TST				0D	6	2	6D	6+	2+	7D	7	3				Test M	•	↕	↕	0	•

Notes:

1. This column gives a base cycle and byte count. To obtain total count, add the values obtained from the INDEXED ADDRESSING MODE table, in Appendix F.
2. R1 and R2 may be any pair of 8 bit or any pair of 16 bit registers.
   The 8 bit registers are: A, B, CC, DP
   The 16 bit registers are: X, Y, U, S, D, PC
3. EA is the effective address.
4. The PSH and PUL instructions require 5 cycles plus 1 cycle for each **byte** pushed or pulled.
5. 5(6) means: 5 cycles if branch not taken, 6 cycles if taken (Branch instructions).
6. SWI sets I and F bits. SWI2 and SWI3 do not affect I and F.
7. Conditions Codes set as a direct result of the instruction.
8. Value of half-carry flag is undefined.
9. Special Case – Carry set if b7 is SET.

M6809 Instruction Set Summary

Table F-2. Indexed Addressing Mode Data

Type	Forms	Non Indirect				Indirect			
		Assembler Form	Postbyte OP Code	$+_{--}^{\times}$	$+_{\#}$	Assembler Form	Postbyte OP Code	$+_{--}$	$+_{\#}$
Constant Offset From R (twos complement offset)	No Offset	,R	1RR00100	0	0	[,R]	1RR10100	3	0
	5 Bit Offset	n, R	0RRnnnnn	1	0	defaults to 8-bit			
	8 Bit Offset	n, R	1RR01000	1	1	[n, R]	1RR11000	4	1
	16 Bit Offset	n, R	1RR01001	4	2	[n, R]	1RR11001	7	2
Accumulator Offset From R (twos complement offset)	A — Register Offset	A, R	1RR00110	1	0	[A, R]	1RR10110	4	0
	B — Register Offset	B, R	1RR00101	1	0	[B, R]	1RR10101	4	0
	D — Register Offset	D, R	1RR01011	4	0	[D, R]	1RR11011	7	0
Auto Increment/Decrement R	Increment By 1	,R+	1RR00000	2	0	not allowed			
	Increment By 2	,R++	1RR00001	3	0	[,R++]	1RR10001	6	0
	Decrement By 1	,-R	1RR00010	2	0	not allowed			
	Decrement By 2	,--R	1RR00011	3	0	[,--R]	1RR10011	6	0
Constant Offset From PC (twos complement offset)	8 Bit Offset	n, PCR	1XX01100	1	1	[n, PCR]	1XX11100	4	1
	16 Bit Offset	n, PCR	1XX01101	5	2	[n, PCR]	1XX11101	8	2
Extended Indirect	16 Bit Address	—	—	—	—	[n]	10011111	5	2

R = X, Y, U or S X = 00 Y = 01
X = Don't Care U = 10 S = 11

$+_{--}$ and $+_{\#}$ Indicate the number of additional cycles and bytes for the particular variation

Examples of Assembly Language Programs

```
Microware OS-9 Assembler 2.1   10/06/92 00:04:21                    Page 001
PIATEST  - PIA Test Program

00001          * Title:     PIA test program
00002          * Name:      PIATEST
00003          *
00004          * Purpose:   To read from port A and write to port B of PIA0 in ROSY
00005          *
00006          *
00007          * Second College on uP-based Real-time Control
00008          * October 1992, csa
00009
00010                                NAM   PIATEST
00011                                TTL   PIA Test Program
00012                                OPT   M
00013
00014    1000                INITPIA EQU   $1000       assign subroutine address
00015    EC10                OREGA   EQU   $EC10       assign port A address
00016    EC12                OREGB   EQU   $EC12       assign port B address
00017
00018    2000                        ORG   $2000       program entry point
00019 W  2000 BD1000                 JSR   INITPIA     initialize PIA
00020 W  2003 B6EC10         LOOP    LDA   OREGA       read port A
00021 W  2006 B7EC12                 STA   OREGB       store in port B
00022    2009 20F8                   BRA   LOOP        repeat
00023                                END

00000 error(s)
00003 warning(s)
$000B 00011 program bytes generated
$0000 00000 data bytes allocated
$0070 00112 bytes used for symbols
```

```
dump piatest.o

Addr   0 1  2 3  4 5  6 7  8 9  A B  C D  E F  0 2 4 6 8 A C E
----   ---- ---- ---- ---- ---- ---- ---- ----  ---------------
0000   BD10 00B6 EC10 B7EC 1220 F8              =..61.71. x
```

Examples of Assembly Language Programs

```
Microware OS-9 Assembler 2.1   10/06/92 00:07:31                    Page 001
INITPIA  - PIA Initialization Subroutine

00001          * Title:    PIA initialization subroutine
00002          * Name:     INITPIA
00003          *
00004          * Purpose:  To initialize PIA0 in ROSY Jr.
00005          *           Port A as input
00006          *           Port B as output
00007          *
00008          *
00009          * PIA addresses:
00010          *      Port A: OREGA and DDRA at $EC10
00011          *              CRA at $EC11
00012          *      Port B: OREGB and DDRB at $EC12
00013          *              CRB at $EC13
00014          *
00015          * Second College on uP-based Real-time Control
00016          * October 1992, csa
00017
00018                                  NAM    INITPIA
00019                                  TTL    PIA Initialization Subroutine
00020                                  OPT    M
00021    EC10              OREGA       EQU    $EC10       assign addresses
00022    EC10              DDRA        EQU    $EC10
00023    EC11              CRA         EQU    $EC11
00024    EC12              OREGB       EQU    $EC12
00025    EC12              DDPB        EQU    $EC12
00026    EC13              CRB         EQU    $EC13
00027
00028    1000                          ORG    $1000       subroutine entry point
00029    1000 3602                     PSHU   A           save accumulator
00030 W  1002 7FEC11                   CLR    CRA         clear CRA b2 to select DDRA
00031 W  1005 7FEC10                   CLR    DDRA        set port A as input
00032    1008 8604                     LDA    #%00000100  to select OREGA
00033 W  100A B7EC11                   STA    CRA
00034 W  100D 7FEC13                   CLR    CRB         clear CRB b2 to select DDRB
00035    1010 86FF                     LDA    #%11111111  set port B as output
00036 W  1012 B7EC12                   STA    DDRB
00037    1015 8604                     LDA    #%00000100  to select OREGB
00038 W  1017 B7EC13                   STA    CRB
00039    101A 3702                     PULU   A           recover accumulator
00040    101C 39                       RTS
00041                                  END

00000 error(s)
00006 warning(s)
$001D 00029 program bytes generated
$0000 00000 data bytes allocated
$008E 00142 bytes used for symbols
```

Examples of Assembly Language Programs

```
Microware OS-9 Assembler 2.1   10/12/92 00:47:19                    Page 001
piaddrv - An Example of SCF Device Driver

00001          * Name:       piaddrv.a
00002          * Title:      Device driver for the PIA in ROSY Jr.
00003          *
00004          * Purpose:    An example of SCF (Sequential Character File) device driver.
00005          *             The device is a parallel interface adapter (PIA0) in ROSY Jr.
00006          *
00007          * Remarks:
00008          *     This illustrates the implementation of a SCF device driver, which
00009          *     is an OS-9 standard memory module that can be executed in ROSY Jr.
00010          *     running under OS-9.
00011          *
00012          *     As a SCF device driver, six funtions (init, read, write, get
00013          *     status, set status, and terminate) are included in the same
00014          *     module.  We use the first one to initialize the PIA - port A
00015          *     as input, port B as output.  The read and write functions are
00016          *     for reading port A and writing port B respectively.  The
00017          *     remaining three functions are not implemented in this example.
00018          *
00019          * Device driver subroutines:
00020          *     The 6 functions are implemented as subroutines in the SCR
00021          *     device drivers:
00022          *     INIT (device initialization): port A as input, port B as output
00023          *     READ (read character): read a byte from port A
00024          *     WRITE (write character): write a byte to port B
00025          *     GETSTA (get device status): just exit
00026          *     SETSTA (set device status): just exit
00027          *     TERM (terminate device): just exit
00028          *
00029          *
00030          * Others:
00031          *     Second College on Real-time Control
00032          *     October 1992, csa
00033          *
00034                                nam   piaddrv
00035                                ttl   An Example of SCF Device Driver
00036
```

Examples of Assembly Language Programs

```
Microware OS-9 Assembler 2.1   10/12/92 00:47:23                    Page 002
piaddrv - An Example of SCF Device Driver

00037          * Read definitions and symbolic names in pass 1 only
00038          * The "defsfile" contains the following:
00039          *      /r0/defs/os9defs
00040          *      /r0/defs/os9sysdefs.li
00041          *      /r0/defs/os9iodefs
00042          *      /r0/defs/os9rbfdefs
00043          *      /r0/defs/os9scfdefs
00044          *      /r0/defs/systype
00045          *      /r0/defs/sysdefs
00046          * Not all the files are needed in this module, but it is convenient
00047          * to have all of them bundled together in "defsfile".  A suitably
00048          * large memory size (e.g. #20k) must be requested during assembly.
00049                               ifp1
00051                               endc
00052
00053
00054          * Module header starts here:
00055          *      The assembly directive "mod" generates the necessary
00056          *      module header based on six parameters in the statement:
00057          *
00058          *           m.size = a label at the end of the module for
00059          *                    evaluating the module size.
00060          *
00061          *           m.name = a label of the module name string for
00062          *                    evaluating the module name offset.
00063          *
00064          *           DRIVR+OBJCT = the value of type/language byte
00065          *                         needed in the module header.
00066          *                         DRIVR=$E0 for physical device driver,
00067          *                         defined in "os9defs".
00068          *                         OBJCT=$01 for 6809 object code module,
00069          *                         defined in "os9defs".
00070          *
00071          *           REENT+1 = the value of attribute/revision byte
00072          *                     needed in the module header.
00073          *                     REENT=$80 for re-entrant module, defined
00074          *                     in "os9defs".
00075          *                     The revision (lower nibble) is set to 1.
00076          *
00077          *           e.offset = label of entry point to execution code
00078          *                      for evaluating execution offset.
00079          *
00080          *           s.size = label for the static data storage required.
00081          *
00082   0000 87CD0052          mod   m.size,m.name,DRIVR+OBJCT,REENT+1,e.offset,s.size
00083   0004 000EE181
00084   0008 89001600
00085   000C 1D                                        mod m.size,m.name,DRIVR+OBJCT,REENT+1,e
00086
```

```
Microware OS-9 Assembler 2.1   10/12/92 00:47:29                    Page 003
piaddrv - An Example of SCF Device Driver

00087          * The next byte defines the file access mode.
00088          * EXEC. and UPDAT. are defined in "os9def".
00089    000D 07                   fcb   EXEC.+UPDAT.
00090
00091          * The next item is the module name string.
00092          * Bit 7 (sign bit) is set for the last character.
00093    000E 70696164    m.name   fcs   "piaddrv"
00094    0012 6472F6                                 m.name fcs "piaddrv"
00095
00096          * Next the revision number.  Set to 1 for the first time
00097    0015 01                   fcb   1           revision number
00098
00099          * Define storage space required.
00100          * V.SCF, defined in "os9scfdefs" is the total SCF manager static
00101          * overhead.
00102          * "org" in OS-9 mode changes the value of the 'data' location
00103          * counter.  It cannot have a label.  Thus an "equ' is needed
00104          * in the second line to assign V.SCF to s.size, the static
00105          * storage size required.
00106 D  001D                       org   V.SCF       end of the SCF definition
00107 D  001D             s.size   equ   .           now assign it
00108
00109          * The execution code starts here:
00110 W  0016 16000F     e.offset lbra  INIT        branch to device initialization
00111 W  0019 160021              lbra  READ        branch to read a byte from port A
00112 W  001C 160024              lbra  WRITE       branch to write a byte to port B
00113 W  001F 160028              lbra  GETSTA      branch to an island then exit
00114 W  0022 160027              lbra  SETSTA      branch to an island then exit
00115 W  0025 160026              lbra  TERM        exit
00116
00117          * INIT subroutine to initialize PIA
00118          * Refer to page 7-10 of OS-9 System Programmer's Manual (SPM) for
00119          * details.
00120          * Input:  (y) = address of device descriptor module
00121          *         (u) = address of device static storage
00122    0028 AE41       INIT      ldx   V.PORT,u    get PIA base address
00123    002A 6F01                 clr   1,x         clear b2 of CRA to select DDRA
00124    002C 6F84                 clr   ,x          clear DDRA, thus port A is input
00125    002E 8604                 lda   #%00000100  set b2 of CRA to select ORA
00126    0030 A701                 sta   1,x
00127    0032 6F03                 clr   3,x         clear b2 of CRB
00128    0034 86FF                 lda   #%11111111  set port B as output
00129    0036 A702                 sta   2,x
00130    0038 8604                 lda   #%00000100  set b2 of CRB to select ORB
00131    003A A703                 sta   3,x
00132    003C 39                   rts               return
00133
00134          * READ subroutine to read a byte from port A
00135          * Refer to SPM page 7-11 for details.
00136          * Input: (y) = address of path descriptor
00137          *        (u) = address of device static storage
00138          * Output: (a) = character read
00139          * Error:  (cc) = C bit set
00140          *         (b)  = error code
00141    003D A6D801     READ      lda   [V.PORT,u] read from port A into accumulator A
00142    0040 1CFE                 andcc #%11111110  no error
00143    0042 39                   rts               return
00144
```

Examples of Assembly Language Programs

```
Microware OS-9 Assembler 2.1   10/12/92 00:47:36                    Page 004
piaddrv - An Example of SCF Device Driver

00145         * WRITE subroutine to write a byte to port B
00146         * Refer to SPM page 7-12 for details
00147         * Input:  (a) = character to write
00148         *         (y) = address of path descriptor
00149         *         (u) = address of device static storage
00150         * Output: none
00151         * Error:  (cc) = C bit set
00152         *         (b) = error code
00153   0043 AE41       WRITE    ldx   V.PORT,u   get PIA base address
00154   0045 A702                sta   2,x        write to port B
00155   0047 1CFE                andcc #%11111110 no error
00156   0049 39                  rts
00157
00158
00159         * GETSTA subroutine - just branch to exit
00160   004A 2002       GETSTA   bra   TERM
00161         * SETSTA subroutine - just branch to exit
00162   004C 2000       SETSTA   bra   TERM
00163
00164         * TERM subroutine - just exit
00165   004E 39         TERM     rts              exit
00166
00167         * "emod" is the end of module directive.  It outputs the 3-byte CRC.
00168   004F 1717DF              emod             end of module and CRC
00169
00170   0052             m.size   equ   *          program size is address of last byte + 1

00000 error(s)
00006 warning(s)
$0052 00082 program bytes generated
$0000 00000 data bytes allocated
$25D2 09682 bytes used for symbols
```

Examples of Assembly Language Programs

```
Microware OS-9 Assembler 2.1   10/11/92 18:00:19                    Page 001
piaddes - An example of a simple device descriptor

00001          * Name:       piaddes.a
00002          * Title:      Device descriptor for the PIA in ROSY Jr.,
00003          *             to be used by device driver "piaddrv".
00004          *
00005          * Purpose:
00006          *     An example device descriptor for the PIA in ROSY Jr.
00007          *     This will be used by the SCF device driver (piaddrv) for
00008          *     getting the hardware addresses and initialization parameters.
00009          *     In this case, only the base address ($EC10) of the PIA in
00010          *     the ROSY Jr. is needed.
00011          *
00012          * Remarks:
00013          *     In general, a device descriptor module is a small, non-
00014          *     executable module that provides information that associates
00015          *     a specific device with its:
00016          *                     hardware controller address(es),
00017          *                     device driver name,
00018          *                     logical name,
00019          *                     file manager name, and
00020          *                     initialization parameters.
00021          *
00022          *     A device descriptor module for a SCF-type device may thus
00023          *     contain a list of control character definitions.  Refer to
00024          *     SPM page 7-6 for such a list.  However, we do not need any
00025          *     control character definition in our simple example of
00026          *     turnig our PIA into a very expensive switch!
00027          *
00028          *     Thus, only the PIA register base address ($EC10), the descriptor's
00029          *     own name (piaddes), the file manager name (SCF) and the device
00030          *     driver name (piaddrv) are passed.
00031          *
00032          * Others:
00033          *     Second College on Real-time Control
00034          *     October 1992, csa
00035
00036                          nam   piaddes
00037
00038                          ttl   An example of a simple device descriptor
00039
```

```
Microware OS-9 Assembler 2.1   10/11/92 18:00:23                    Page 002
piaddes - An example of a simple device descriptor

00040         * Read "defsfile" for symbolic names and definitions in pass 1 only.
00041         * The line between the conditional assembly directives "ifp1" and
00042         * "endc" is "use defsfile".  For some unknown reasons, I cannot get
00043         * it printed on the assembly listing.  Please refer to the source
00044         * listing if necessary.
00045                               ifp1              conditional assembly directive
00047                               endc              end of conditional assembly
00048
00049         * Module header starts here:
00050         * Please refer to "piaddrv.a", the device driver listing for
00051         * explanation if this is the first time you see a memory module
00052         * being defined.  I am tired of typing the same stuff using the
00053         * OS-9 EDITOR!!! (Must use SLED, but don't tell Rinus!?)
00054   00F1            type      set   DEVIC+OBJCT device descriptor, 6809 object code
00055   0081            revs      set   REENT+1     re-entrant module, 1=revision number
00056   EC10            piabase   set   $EC10       PIA base address
00057   0000 87CD0026             mod   len,name,type,revs,fm.name,drv.name
00058   0004 0012F181
00059   0008 F1001900
00060   000C 1C                                     mod len,name,type,revs,fm.name,drv.name
00061
00062         * Next is the mode byte in the header of the module.
00063   000D 03                   fcb   UPDAT.
00064
00065         * Then come 3 bytes of device controller's absolute (yes, we manage
00066         * to get an 'absolute address' finally.) physical address.
00067         * Since our M6809 only has a 2-byte address for the PIA, the first
00068         * byte is set to $FF.  The base address of the PIA in ROSY Jr.,
00069         * which is $EC10, then follows.
00070   000E FF                   fcb   $FF
00071   000F EC10                 fdb   piabase     piabase is the PIA base address set pre
00072
00073         * The next byte is the initialization table size.  Since we do not
00074         * have such a table it is set to 0.
00075   0011 00                   fcb   0           no init table
00076
00077         * Here would have been the initialization table if there is one.
00078
00079         * The next three items are the names needed by the module.
00080         * Bit 7 (sign bit) of the last character of each name string is set
00081         * as a delimiter.
00082   0012 70696164   name      fcs   "piaddes"   device name
00083   0016 6465F3                                 name fcs "piaddes"device name
00084   0019 5343C6     fm.name   fcs   "SCF"       file manager
00085   001C 70696164   drv.name  fcs   "piaddrv"   device driver
00086   0020 6472F6                                 drv.name fcs "piaddrv"device driver
00087
00088   0023 D66A33               emod              end of module and produce 3-byte CRC
00089
00090   0026            len       EQU   *           length of module

00000 error(s)
00000 warning(s)
$0026 00038 program bytes generated
$0000 00000 data bytes allocated
$25A5 09637 bytes used for symbols
```

Examples of Assembly Language Programs

```
dump piaddrv

Addr   0 1  2 3  4 5  6 7  8 9  A B  C D  E F  0 2 4 6 8 A C E
----   ---- ---- ---- ---- ---- ---- ---- ----  ----------------
0000   87CD 0052 000E E181 8900 1600 1D07 7069  .M.R..a.......pi
0010   6164 6472 F601 1600 0F16 0021 1600 2416  addrv.......!..$.
0020   0028 1600 2716 0026 AE41 6F01 6F84 8604  .(..'..&.Ao.o....
0030   A701 6F03 86FF A702 8604 A703 39A6 D801  '.o...'...'.9&X.
0040   1CFE 39AE 41A7 021C FE39 2002 2000 3917  ..9.A'...9 . .9.
0050   17DF                                     '_


dump piaddes

Addr   0 1  2 3  4 5  6 7  8 9  A B  C D  E F  0 2 4 6 8 A C E
----   ---- ---- ---- ---- ---- ---- ---- ----  ----------------
0000   87CD 0026 0012 F181 F100 1900 1C02 FFEC  .M.&..q.q.......l
0010   1000 7069 6164 6465 F353 43C6 7069 6164  ..piaddesSCFpiad
0020   6472 F6D6 7B37                           drvV{7
```

```
/*****************************************************************
* Name:       piatest.c
* Title:      A simple PIA test program
*
* Purpose:
*     This is a small C program written to illustrate the use of
*     non-standard device driver and device descriptor.
*
* Remarks:
*     This is a solution to the problem given on page 10 of csa's
*     notes on "Recall of the M6809", implemented on OS-9.
*
*     The PIA is treated as a sequential character file device and
*     it is accessed like any other SCF device, except that in
*     this case the device descriptor and device driver are both
*     user written.
*
*     The program first opens the file "piaddes" for read and
*     write.  "piaddes" is actually a device descriptor which
*     provides the necessary information to operate the PIA in
*     ROSY Jr.  This includes the physical address of the PIA,
*     the name of the device driver "piaddrv" and the name of
*     the file manager "SCF".
*
*     It then reads continuously from the device and writes the
*     received byte back to the device.  It is the device driver
*     that actually performs the read and write functions.  It
*      reads from port A and writes to port B.
*
*     By treating the PIA as a SCF device, the user is now shielded
*     from the nitty gritty of the PIA register control, and can
*     write in the 'comfort' of C environment! (Ha!Ha!Ha!)
*
*     To execute this program, three modules must be in memory -
*     SCF (the file manager), piaddrv (the device driver) and
*     piaddes (the device descriptor).  Since the SCF is normally
*     resident in the memory, only the last two need be loaded.
*
* Others:
*     Second College on Real-time Control
*     October 1992, csa
*****************************************************************/

#include <stdio.h>
#include <errno.h>
#define FILE_NAM "/piaddes"      /* name of device descriptor */
#define RW 3                     /* 3 for read & write mode */

main() {
  char buffer;    /* buffer for value from port A */
  int path_no;    /* path number return when "piaddrv" is opened */
  int byte_count=1;   /* number of byte to read, set to 1 */

  printf("Piatest running ...   hit Ctrl-C to abort\n");

  if ((path_no = open(FILE_NAM, RW)) == -1) exit(errno);

  while(1) {
    read(path_no, &buffer, byte_count);
    write(path_no, &buffer, byte_count);
  }
}
```