6500 CHIP FAMILY SUPPLEMENT

TO THE

CROSS ASSEMBLER MANUAL

CAS-2006-00

TABLE OF CONTENTS

LIST OF TABLES

This guide is a supplement to the Emulogic Relocatable Macro Cross Assembler Manual, providing specific assembler information for writing software programs to run on the 6500 microprocessor series. This supplement applies to the following chips:

* 6502	* 6505	* 6512
* 6503	* 6506	* 6513
* 6504	* 6507	* 6515

This supplement includes:

* a summary of the 6500 instruction set,

* procedures for running the 6500 cross assembler, and

* sample 6500 cross assembler output listings

6500 PERMANENT SYMBOL TABLE

The following is a summary of the mnemonics for the operation (op) codes included in the 6500 series instruction set. They are stored in the Permanent Symbol Table and are automatically recognized by the Emulogic relocatable macro cross assembler. References to, and operations with the registers within the microprocessor are legal. For a detailed description of the 6500 op codes, refer to the Synertek SY6500/MCS6500 Microcomputer Family Programming Manual.

Instruction operands are represented herein as follows:

Operand	Meaning
ii	Immediate operand (byte)
^^	8-bit relative branch address
aa	8-bit address variable (zero page)
aaaa	16-bit absolute address
x	X index register
y	Y index register
A	Accumulator

Programming Notes:

(1) Address references in zero page must be predefined.

(2) The use of complex forward references should be avoided as they may result in phase errors.

TABLE S-1. 6500 INSTRUCTION SET SUMMARY

MNEMONIC	OPERANDS	DESCRIPTION	EXAMPLE
ADC	#ii	Add immediate to accumulator with carry	ADC #20
ADC	aa	Add memory to accumulator with carry	ADC 3F
ADC	aaaa	Add memory to accumulator with carry	ADC 0FFF
ADC	aa,x	Add memory indexed to accumulator with carry	ADC 3,X
ADC	aaaa,x	Add memory indexed to accumulator with carry	ADC TAG,X
ADC	aaaa,y	Add memory indexed to accumulator with carry	ADC 245,Y
ADC	(aa,x)	Add memory indexed indirect to accumulator with carry	ADC (4,X)
ADC	(aa),y	Add memory indirect indexed to accumulator with carry	ADC (7),Y
AND	#ii	AND immediate with accumulator	AND #125.
AND	aa	AND memory with accumulator	AND 10
AND	aaaa	AND memory with accumulator	AND 258.
AND	aa,x	AND memory indexed with accumulator	AND 5A,X
AND	aaaa,x	AND memory indexed with accumulator	AND 300.,X
AND	aaaa,y	AND memory indexed with accumulator	AND TAG,Y
AND	(aa,x)	AND memory indexed indirect with accumulator	AND (0FE,X)
AND	(aa),y	AND memory indirect indexed with accumulator	AND (25.),Y
ASL	A	Shift left accumulator one bit	ASL A
ASL	aa	Shift left memory one bit	ASL 254.
ASL	aaaa	Shift left memory one bit	ASL TAG

Table S-1. 6500 Instruction Set Summary (contd)

MNEMONIC	OPERANDS	DESCRIPTION	EXAMPLE
ASL	aa,x	Shift left memory indexed one bit	ASL 0A4,X
ASL	aaaa,x	Shift left memory indexed one bit	ASL 1452,X
BCC	^^	Branch on carry clear	BCC 12
BCS	^^	Branch on carry set	BCS TAG
BEQ	^^	Branch on result zero	BEQ 34.
BIT	aa	Test bits in memory with accumulator	BIT 0FF
BIT	aaaa	Test bits in memory with accumulator	BIT 257.
BMI	^^	Branch on result minus	BMI 0A
BNE	^^	Branch on result not zero	BNE 123.
BPL	^^	Branch on result plus	BPL TAG1
BRK		Force break	BRK
BVC	^^	Branch on overflow clear	BVC 24
BVS	^^	Branch on overflow set	BVS 0AB
CLC		Clear carry flag	CLC
CLD		Clear decimal load	CLD
CLI		Clear interrupt disable bit	CLI
CLV		Clear overflow flag	CLV
CMP	#ii	Compare immediate to accumulator	CMP #250.
CMP	aa	Compare memory to accumulator	CMP 86
CMP	aaaa	Compare memory to accumulator	CMP 257.
CMP	aa,x	Compare memory indexed to accumulator	CMP 4,X
CMP	aaaa,x	Compare memory indexed to accumulator	CMP 0FFFF,X

Table S-1. 6500 Instruction Set Summary (contd)

MNEMONIC	OPERANDS	DESCRIPTION	EXAMPLE
CMP	aaaa,y	Compare memory indexed to accumulator	CMP TAG,Y
CMP	(aa,x)	Compare memory indexed indirect to accumulator	CMP (255.,X)
CMP	(aa),y	Compare memory indirect indexed to accumulator	CMP (6F),Y
CPX	#ii	Compare immediate and index X	CPX #77
CPX	aa	Compare memory and index X	CPX 45
CPX	aaaa	Compare memory and index X	CPX 284
CPY	#ii	Compare immediate and index Y	CPY #2
CPY	aa	Compare memory and index Y	CPY 7F
CPY	aaaa	Compare memory and index Y	CPY 0FFA
DEC	aa	Decrement memory by one	DEC 80
DEC	aaaa	Decrement memory by one	DEC TAG
DEC	aa,x	Decrement memory indexed by one	DEC 7F,X
DEC	aaaa,x	Decrement memory indexed by one	DEC 3FA,X
DEX		Decrement index X by one	DEX
DEY		Decrement index Y by one	DEY
EOR	#ii	Exclusive OR immediate with accumulator	EOR #9
EOR	aa	Exclusive OR memory with accumulator	EOR 0F7
EOR	aaaa	Exclusive OR memory with accumulator	EOR 100
EOR	aa,x	Exclusive OR memory indexed with accumulator	EOR 25,X
EOR	aaaa,x	Exclusive OR memory indexed with accumulator	EOR TAG,X
EOR	aaaa,y	Exclusive OR memory indexed with accumulator	EOR 200,Y

Table S-1. 6500 Instruction Set Summary (contd)

MNEMONIC	OPERANDS	DESCRIPTION	EXAMPLE
EOR	(aa,x)	Exclusive OR memory indexed indirect with accumulator	EOR (5,X)
EOR	(aa),y	Exclusive OR memory indirect indexed with accumulator	EOR (0FF),Y
INC	aa	Increment memory by one	INC 88
INC	aaaa	Increment memory by one	INC 986
INC	aa,x	Increment memory indexed by one	INC 35,X
INC	aaaa,x	Increment memory indexed by one	INC 458,X
INX		Increment index X by one	INX
INY		Increment index Y by one	INY
JMP	aaaa	Jump to new location	JMP 6
JMP	(aaaa)	Jump to new location indirect	JMP (TAG)
JSR	aaaa	Jump to new location saving return address	JSR 101
LDA	#ii	Load accumulator with immediate	LDA #55
LDA	aa	Load accumulator with memory	LDA 99
LDA	aaaa	Load accumulator with memory	LDA 105
LDA	aa,x	Load accumulator with memory indexed	LDA 0FF,X
LDA	aaaa,x	Load accumulator with memory indexed	LDA TAG,X
LDA	aaaa,y	Load accumulator with memory indexed	LDA 3FF,Y
LDA	(aa,x)	Load accumulator with memory indexed indirect	LDA (0DD,X)
LDA	(aa),y	Load accumulator with memory indirect indexed	LDA (55),Y
LDX	#ii	Load index X with immediate data	LDX #123.
LDX	aa	Load index X with memory	LDX 255.
LDX	aaaa	Load index X with memory	LDX 100

Table S-1. 6500 Instruction Set Summary (contd)

MNEMONIC	OPERANDS	DESCRIPTION	EXAMPLE
LDX	aa,y	Load index X with memory indexed	LDX 0AB,Y
LDX	aaaa,y	Load index X with memory indexed	LDX 105,Y
LDY	#ii	Load index Y with immediate data	LDY #34
LDY	aa	Load index Y with memory	LDY 0FA
LDY	aaaa	Load index Y with memory	LDY 555
LDY	aa,x	Load index Y with memory indexed	LDY 2,X
LDY	aaaa,x	Load index Y with memory indexed	LDY 4F5,X
LSR	A	Shift right accumulator one bit	LSR A
LSR	aa	Shift right memory one bit	LSR 55
LSR	aaaa	Shift right memory one bit	LSR 375
LSR	aa,x	Shift right memory indexed one bit	LSR 41,X
LSR	aaaa,x	Shift right memory indexed one bit	LSR TAG,X
NOP		No operation	NOP
ORA	#ii	OR immediate with accumulator	ORA #154.
ORA	aa	OR memory with accumulator	ORA 0F1
ORA	aaaa	OR memory with accumulator	ORA TAG1
ORA	aa,x	OR memory indexed with accumulator	ORA 56,X
ORA	aaaa,x	OR memory indexed with accumulator	ORA 678,X
ORA	aaaa,y	OR memory indexed with accumulator	ORA 34,Y
ORA	(aa,x)	OR memory indexed indirect with accumulator	ORA (45,X)
ORA	(aa),y	OR memory indirect indexed with accumulator	ORA (77),Y
PHA		Push accumulator on stack	PHA

Table S-1. 6500 Instruction Set Summary (contd)

MNEMONIC	OPERANDS	DESCRIPTION	EXAMPLE
PHP		Push processor status on stack	PHP
PLA		Pull accumulator from stack	PLA
PLP		Pull processor status from stack	PLP
ROL	A	Rotate left accumulator one bit	ROL A
ROL	aa	Rotate left memory one bit	ROL 23
ROL	aaaa	Rotate left memory one bit	ROL 0FFF
ROL	aa,x	Rotate left memory indexed one bit	ROL 34,X
ROL	aaaa,x	Rotate left memory indexed one bit	ROL TAG,X
ROR	A	Rotate right accumulator one bit	ROR A
ROR	aa	Rotate right memory one bit	ROR 63
ROR	aaaa	Rotate right memory one bit	ROR 1627
ROR	aa,x	Rotate right memory indexed one bit	ROR 3,X
ROR	aaaa,x	Rotate right memory indexed one bit	ROR 0CA4,X
RTI		Return from interrupt	RTI
RTS		Return from subroutine	RTS
SBC	#ii	Subtract immediate from accumulator with borrow	SBC #0FF
SBC	aa	Subtract memory from accumulator with borrow	SBC 65.
SBC	aaaa	Subtract memory from accumulator with borrow	SBC 250
SBC	aa,x	Subtract memory indexed from accumulator with borrow	SBC 0ED,X
SBC	aaaa,x	Subtract memory indexed from accumulator with borrow	SBC 1000,X
SBC	aaaa,y	Subtract memory indexed from accumulator with borrow	SBC 8219,Y
SBC	(aa,x)	Subtract memory indexed indirect from accumulator with borrow	SBC (88,X)

Table S-1. 6500 Instruction Set Summary (contd)

MNEMONIC	OPERANDS	DESCRIPTION	EXAMPLE
SBC	(aa),y	Subtract memory indirect indexed from accumulator with borrow	SBC (0DC),Y
SEC		Set carry flag	SEC
SED		Set decimal mode	SED
SEI		Set interrupt disable status	SEI
STA	aa	Store accumulator in memory	STA 2
STA	aaaa	Store accumulator in memory	STA 258.
STA	aa,x	Store accumulator in memory indexed	STA 0BA,X
STA	aaaa,x	Store accumulator in memory indexed	STA TAG,X
STA	aaaa,y	Store accumulator in memory indexed	STA 56,Y
STA	(aa,x)	Store accumulator in memory indexed indirect	STA (65,X)
STA	(aa),y	Store accumulator in memory indirect indexed	STA (52),Y
STX	aa	Store index X in memory	STX 0FF
STX	aaaa	Store index X in memory	STX TAG1
STX	aa,y	Store index X in memory indexed	STX 79,Y
STY	aa	Store index Y in memory	STY 45
STY	aaaa	Store index Y in memory	STY 376
STY	aa,x	Store index Y in memory indexed	STY 123.,X
TAX		Transfer accumulator to index X	TAX
TAY		Transfer accumulator to index Y	TAY
TSX		Transfer stack pointer to index X	TSX
TXA		Transfer index X to accumulator	TXA
TXS		Transfer index X to stack pointer	TXS
TYA		Transfer index Y to accumulator	TYA

CALLING THE 6500 CROSS ASSEMBLER

To call the 6500 cross assembler from the system device, enter the following command in response to the RT-11 keyboard monitor prompt:

```
.RUN X6500 CR
```

When the cross assembler responds with an asterisk (*), it is ready to accept command string input and to perform an assembly.

TERMINATING THE 6500 CROSS ASSEMBLER

If you have typed

```
.RUN X6500 CR
```

and received the asterisk prompt but have not yet entered the command string, you can terminate 6500 cross assembler control and return to the keyboard monitor by typing

```
^C
```

If you have completed command string input and started an assembly, you can halt the assembly process at any time by typing

```
^C^C
```

This returns control to the RT-11 keyboard monitor, and a system monitor prompt (.) will appear on the terminal screen.

SAMPLE 6500 ASSEMBLY LISTINGS

The remainder of this supplement consists of sample output listings from the 6500 cross assembler.

```
                        .RADIX  16
                        .ASECT
                    ;
                    ;       DIRECT ASSIGNMENT OF LABELS
                    ;
                        PC=32
                        SEMI=3B
                        DE1=0EC18
                        PACK=0EA84
                        PHXY=0EB9E
                        PLXY=0EBAC
                        COL0=0F2E1
                        COL1=0F321
                        COL2=0F361
                        COL3=0F3A1
                        COL4=0F3E1
                        T2L=0A808
                        MOTON=0C
                        MOTOFF=0E
                        DRB=0A000
                        DRA=0A001
                        DDRB=0A002
                        DDRA=0A003
                        T1L=0A004
                        T1CH=0A005
                        T1H=0A007
                        ACR=0A00B
                        PCR=0A00C
                        IFR=0A00D
                        .=190
0190    00              SAVA:   .BYTE
0191    00              EQFL:   .BYTE
0192    00              CRFL:   .BYTE
0193    00              PBPTR:  .BYTE
0194    00              PBUF:   .BYTE
                        .=200
                    ;
                    ;       ENTRY & INITIALIZATION
                    ;
0200    08          PRINT:  PHP             ;SAVE PROCESSOR STATUS
0201    78                  SEI             ;DISABLE INTERRUPT DURING PRINT
0202    A9  D0              LDA     #0D0
0204    8D  04  A0          STA     T1L
0207    A9  0C              LDA     #MOTON
0209    8D  0C  A0          STA     PCR     ;START MOTOR
020C    2C  00  A0  PR1:    BIT     DRB     ;TEST LIMIT SWITCHES
020F    50  53              BVC     RMAR
0211    30  F9              BMI     PR1
                    ;
                    ;       LEFT TO RIGHT PRINT
                    ;
0213    20  CF  02  LMAR:   JSR     DEBDEL  ;DEBOUNCE DELAY
0216    A0  00              LDY     #0
0218    2C  00  A0  LM1:    BIT     DRB
021B    10  FB              BPL     LM1     ;WAIT TO CLEAR MARGIN
021D    A9  01              LDA     #1
021F    8D  05  A0          STA     T1CH    ;START DOT RIMER(200)
```

```
0222  B9 94 01  LM2:   LDA  PBUF,Y   ;LOAD WITH CHARACTER
0225  29 3F            AND  #3F
0227  AA               TAX
0228  A9 20            LDA  #20
022A  99 94 01         STA  PBUF,Y   ;REPLACE WITH BLANK
022D  BD E1 F2         LDA  COL0,X
0230  20 A6 02         JSR  OUTDOT   ;OUTPUT COLUMN 0
0233  BD 21 F3         LDA  COL1,X
0236  20 A6 02         JSR  OUTDOT   ;OUTPUT COLUMN 1
0239  BD 61 F3         LDA  COL2,X
023C  20 A6 02         JSR  OUTDOT   ;OUTPUT COLUMN 2
023F  BD A1 F3         LDA  COL3,X
0242  20 A6 02         JSR  OUTDOT   ;OUTPUT COLUMN 3
0245  BD E1 F3         LDA  COL4,X
0248  20 A6 02         JSR  OUTDOT   ;OUTPUT COLUMN 4
024B  A9 00            LDA  #0       ;INSERT 1 SPACE BETWEEN CHARACTERS
024D  20 A6 02         JSR  OUTDOT
0250  C8               INY
0251  C0 48            CPY  #72.     ;END OF LINE?
0253  90 CD            BCC  LM2      ;IF NOT, GET MORE CHARACTERS
                ;
                ;      EXIT ROUTINE
                ;
0255  A9 FF     PRXIT: LDA  #0FF
0257  8D 08 A8         STA  T2L
025A  20 18 EC         JSR  DE1
025D  A9 0E            LDA  #MOTOFF
025F  8D 0C A0         STA  PCR      ;MOTOR OFF
0262  28               PLP           ;RESTORE PROCESSOR STATUS
0263  60               RTS
                ;
                ;      RIGHT TO LEFT PRINT
                ;
0264  20 CF 02  RMAR:  JSR  DEBDEL
0267  A0 47            LDY  #71.     ;RIGHT BUFFER LIMIT
0269  2C 00 A0  RM1:   BIT  DRB
026C  50 FB            BVC  RM1
026E  A9 01            LDA  #1
0270  8D 05 A0         STA  T1CH
0273  B9 94 01  RM2:   LDA  PBUF,Y
0276  29 3F            AND  #3F
0278  AA               TAX
0279  A9 20            LDA  #20
027B  99 94 01         STA  PBUF,Y
027E  BD E1 F3         LDA  COL4,X
0281  20 A6 02         JSR  OUTDOT
0284  BD A1 F3         LDA  COL3,X
0287  20 A6 02         JSR  OUTDOT
028A  BD 61 F3         LDA  COL2,X
028D  20 A6 02         JSR  OUTDOT
0290  BD 21 F3         LDA  COL1,X
0293  20 A6 02         JSR  OUTDOT
0296  BD E1 F2         LDA  COL0,X
0299  20 A6 02         JSR  OUTDOT
029C  A9 00            LDA  #0
029E  20 A6 02         JSR  OUTDOT
02A1  88               DEY
```

```
115  02A2  10  CF          BPL   RM2
116  02A4  30  AF          BMI   PRXIT
117                    ;
118                    ;     HERE TO OUTPUT 1 COLUMN OF DOTS
119                    ;
120  02A6  49  FF      OUTDOT: EOR   #0FF    ;INVERT FOR OUTPUT
121  02A8  2C  0D  A0  OD1:    BIT   IFR
122  02AB  50  FB          BVC   OD1     ;WAIT FOR INTER-DOT TIMEOUT
123  02AD  8D  01  A0      STA   DRA     ;OUTPUT DOTS
124  02B0  A9  05          LDA   #5
125  02B2  8D  07  A0      STA   T1H     ;LOAD INTER-DOT TIME
126  02B5  A9  86          LDA   #86
127  02B7  8D  04  A0      STA   T1L
128  02BA  A9  FF          LDA   #0FF
129  02BC  2C  0D  A0  OD2:    BIT   IFR
130  02BF  50  FB          BVC   OD2     ;WAIT FOR DOT TIMEOUT
131  02C1  8D  01  A0      STA   DRA     ;OFF
132  02C4  A9  01          LDA   #1
133  02C6  8D  07  A0      STA   T1H
134  02C9  A9  D0          LDA   #0D0
135  02CB  8D  04  A0      STA   T1L
136  02CE  60              RTS
137                    ;
138                    ;     DELAY ROUTINE
139                    ;
140  02CF  A9  10      DEBDEL: LDA   #10     ;DEBOUNCE DELAY
141  02D1  8D  08  A8      STA   T2L
142  02D4  A9  27          LDA   #27
143  02D6  4C  18  EC      JMP   DE1
144                    ;
145                    ;     INITIALIZATION ROUTINE
146                    ;
147  02D9  A9  47      DRI:    LDA   #71.
148  02DB  A9  20          LDA   #20
149  02DD  9D  94  01  DRI1:   STA   PBUF,X  ;CLEAR BUFFER
150  02E0  CA              DEX
151  02E1  10  FA          BPL   DRI1
152  02E3  A9  00          LDA   #0
153  02E5  8D  93  01      STA   PBPTR
154  02E8  8D  92  01      STA   CRFL
155  02EB  8D  91  01      STA   EQFL
156  02EE  8E  01  A0      STX   DRA
157  02F1  8E  03  A0      STX   DDRA
158  02F4  A9  40          LDA   #40
159  02F6  8D  0B  A0      STA   ACR     ;T1 FREE RUN
160  02F9  60              RTS
161                    ;
162                    ;     DRIVER ROUTINE
163                    ;
164  02FA  90  DD      DRIVER: BCC   DRI     ;CHECK FOR INITIALIZATION
165  02FC  68              PLA           ;GET CHARACTER TO BE PRINTED
166  02FD  20  9E  EB      JSR   PHXY
167  0300  8D  90  01      STA   SAVA
168  0303  29  7F          AND   #7F
169  0305  C9  0D          CMP   #0D     ;CARRIAGE RETURN?
170  0307  D0  0E          BNE   DR1
171  0309  0E  92  01      ASL   CRFL    ;YES
```

```
172  030C  90 03          BCC   CR1     ;FLAG SET?
173  030E  20 7A 03       JSR   PLINE   ;YES,PRINT LINE
174  0311  38        CR1: SEC           ;SET CARRY FLAG
175  0312  6E 92 01       ROR   CRFL    ;SET CARRIAGE RETURN FLAG
176  0315  D0 36          BNE   DRXIT
177  0317  C9 3D     DR1: CMP   #3D     ;IS THERE AN ' = '?
178  0319  D0 1A          BNE   DR3
179  031B  0E 92 01       ASL   CRFL    ;YES
180  031E  90 0E          BCC   DR2
181  0320  20 00 02       JSR   PRINT   ;PRINT LINE
182  0323  A9 00          LDA   #0
183  0325  8D 93 01       STA   PBPTR   ;ZERO BUFFER POINTER
184  0328  38             SEC
185  0329  6E 91 01       ROR   EQFL    ;SET EQUAL FLAG
186  032C  D0 1F          BNE   DRXIT
187  032E  0E 91 01  DR2: ASL   EQFL    ;CRFL NOT SET, TEST EQFL
188  0331  90 35          BCC   STUFF   ;PUT ' = ' IN BUFFER IF FIRST
189  0333  B0 18          BCS   DRXIT   ;IGNORE IF SECOND
190  0335  C5 3B     DR3: CMP   SEMI    ;SEMICOLON?
191  0337  D0 1B          BNE   DR5
192  0339  0E 92 01       ASL   CRFL    ;YES
193  033C  AE 93 01       LDX   PBPTR
194  033F  E0 0C          CPX   #12.    ;START OF LINE?
195  0341  F0 25          BEQ   STUFF
196  0343  A2 1E     DR4: LDX   #30.    ;NO
197  0345  EC 93 01       CPX   PBPTR   ;TAB TO COLUMN 30
198  0348  90 03          BCC   DRXIT
199  034A  8E 93 01       STX   PBPTR
200  034D  20 AC EB  DRXIT: JSR  PLXY
201  0350  AD 90 01       LDA   SAVA
202  0353  60             RTS
203  0354  0E 92 01  DR5: ASL   CRFL    ;NOT CARRIAGE RETURN,EQUAL OR SEMICOLON
204  0357  90 0F          BCC   STUFF   ;LOAD
205  0359  A2 0C          LDX   #12.
206  035B  EC 93 01       CPX   PBPTR   ;CHECK FOR BEYOND COLUMN 12
207  035E  90 05          BCC   DR6
208  0360  8E 93 01       STX   PBPTR   ;TAB TO COLUMN 12
209  0363  B0 03          BCS   STUFF   ;LOAD
210  0365  20 7A 03  DR6: JSR   PLINE   ;PRINT LINE
211  0368  AD 90 01 STUFF: LDA  SAVA    ;GET CHARACTER
212  036B  AE 93 01       LDX   PBPTR   ;GET BUFFER POINTER
213  036E  E0 48          CPX   #72.    ;CHECK FOR FULL
214  0370  B0 DB          BCS   DRXIT
215  0372  9D 94 01       STA   PBUF,X  ;NO,PUT CHARACTER IN BUFFER
216  0375  EE 93 01       INC   PBPTR   ;INCREMENT FOR ANOTHER
217  0378  D0 D3          BNE   DRXIT
218  037A  20 00 02 PLINE: JSR  PRINT
219  037D  A2 00          LDX   #0
220  037F  A5 33          LDA   PC+1    ;PC UPPER
221  0381  20 8F 03       JSR   CONVT
222  0384  A5 32          LDA   PC      ;PC LOWER
223  0386  20 8F 03       JSR   CONVT
224  0389  A9 0C          LDA   #12.
225  038B  8E 93 01       STX   PBPTR   ;SET COLUMN POINTER
226  038E  60             RTS
227                  ;
228                  ;     HEX TO ASCII CONVERSION AND LOAD
```

```
  229   038F    48                  CONVT:  PHA
  230   0390    4A                          LSR     A
  231   0391    4A                          LSR     A
  232   0392    4A                          LSR     A
  233   0393    4A                          LSR     A
  234   0394    20    9A    03              JSR     CONV
  235   0397    68                          PLA
  236   0398    29    0F                    AND     #0F
  237   039A    18                  CONV:   CLC             ;CLEAR CARRY FLAG
  238   039B    69    30                    ADC     #30
  239   039D    C9    3A                    CMP     #3A
  240   039F    90    02                    BCC     CONV1
  241   03A1    69    06                    ADC     #6
  242   03A3    9D    94    01      CONV1:  STA     PBUF,X
  243   03A6    E8                          INX
  244   03A7    60                          RTS
  245           0001                        .END
```

```
.MAIN.  X6500  V1.01 16-MAY-82 PAGE 1-5
SYMBOL TABLE

ACR    =  A00B        DDRB   =  A002        DR4     0343        OUTDOT  02A6        FR1     020C
COL0   =  F2E1        DEBDEL    02CF        DR5     0354        PACK   =  EA84        RMAR    0264
COL1   =  F321        DE1    =  EC18        DR6     0365        PBPTR      0193        RM1     0269
COL2   =  F361        DRA    =  A001        EQFL    0191        PBUF       0194        RM2     0273
COL3   =  F3A1        DRB    =  A000        IFR    =  A00D      PC     =   0032        SAVA    0190
COL4   =  F3E1        DRI       02D9        LMAR    0213        PCR    =   A00C        SEMI   =  003B
CONV      039A        DRIVER    02FA        LM1     0218        PHXY   =   EB9E        STUFF      0368
CONVT     038F        DRI1      02DD        LM2     0222        PLINE      037A        T1CH   =  A005
CONV1     03A3        DRXIT     034D        MOTOFF=    000E     PLXY   =   EBAC        T1H    =  A007
CRFL      0192        DR1       0317        MOTON  =   000C     PRINT      0200        T1L    =  A004
CR1       0311        DR2       032E        OD1        02A8     PRXIT      0255        T2L    =  A808
DDRA   =  A003        DR3       0335        OD2        02BC

. ABS.    03A8      00
          0000      01
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  288 WORDS  ( 2 PAGES)
DYNAMIC MEMORY AVAILABLE FOR  74 PAGES
,DY1:TST65=DY1:TST65
```

ADDENDUM TO: 6500 CHIP FAMILY SUPPLEMENT TO THE CROSS ASSEMBLER MANUAL OCTOBER 14, 1983

The ability to reference the low and/or high byte of a word has been added to the instruction set of the EMULOGIC 6500 cross assembler. This added capability has been provided for all 8-bit immediate data instructions. A coding example is provided in the following table.

MNEMONIC/OPERAND		DESCRIPTION	EXAMPLE
ADC	#nn(L)	Add with carry the low order byte of the immedaite data to the accumulator	ADC #TAG(L)
ADC	#nn(H)	Add with carry the high order byte of the immediate data to the accumulator	ADC #TAG(H)
AND	#nn(L)	Logical AND the low order byte of the immediate data and the accumulator	AND #TAG(L)
AND	#nn(H)	Logical AND the high order byte of the immediate data and the accumulator	AND #TAG(H)
CMP	#nn(L)	Compare the low order byte of the immediate data with the accumulator	CMP #TAG(L)
CMP	#nn(H)	Compare the high order byte of the immediate data with the accumulator	CMP #TAG(H)
CPX	#nn(L)	Compare the low order byte of the immediate data with Index X	CPX #TAG(L)
CPX	#nn(H)	Compare the high order byte of the immediate data with Index X	CPX #TAG(H)
CPY	#nn(L)	Compare the low order byte of the immediate data with Index Y	CPY #TAG(L)
CPY	#nn(H)	Compare the high order byte of the immediate data with Index Y	CPY #TAG(H)
EOR	#nn(L)	Exclusive OR the low order byte of the immedaite data and the accumulator	EOR #TAG(L)
EOR	#nn(H)	Exclusive OR the high order byte of the immediate data and the accumulator	EOR #TAG(H)
LDA	#nn(L)	Load the accumulator with the low order byte of the immediate data	LDA #TAG(L)
LDA	#nn(H)	Load the accumulator with the high order byte of the immediate data	LDA #TAG(H)
LDX	#nn(L)	Load Index X with the low order byte of the immediate data	LDX #TAG(L)
LDX	#nn(H)	Load Index X with the high order byte of the immediate data	LDX #TAG(H)
LDY	#nn(L)	Load Index Y with the low order byte of the immediate data	LDY #TAG(L)
LDY	#nn(H)	Load Index Y with the high order byte of the immediate data	LDY #TAG(H)
ORA	#nn(L)	Logical OR the low order byte of the immediate data with the accumulator	ORA #TAG(L)

MNEMONIC/OPERAND		DESCRIPTION	EXAMPLE
ORA	#nn(H)	Logical OR the high order byte of the immediate data with the accumulator	ORA #TAG(H)
SBC	#nn(L)	Subtract the low order byte of the immediate data from the accumulator with borrow	SBC #TAG(L)
SBC	#nn(H)	Subtract the high order byte of the immediate data from the accumulator with borrow.	SBC #TAG(H)

TAG = absolute reference, relocatable reference, or global reference

Additionally, the ability to force absolute and absolute indexed addressing modes (for those intructions which it is legal) has been added to the EMULOGIC 6500 cross assembler. The instructions for which this is legal are described in the table below.

MNEMONIC/OPERAND		DESCRIPTION	EXAMPLE
ADC	OPER	Add with carry memory to accumulator	ADC TAG(A)
ADC	OPER,X	Add with carry memory to accumulator indexed by X	ADC TAG(A),X
ADC	OPER,Y	Add with carry memory to accumulator indexed by Y	ADC TAG(A),Y
AND	OPER	Logical AND memory with accumulator	AND TAG(A)
AND	OPER,X	Logical AND memory with accumulator indexed by X	AND TAG(A),X
AND	OPER,Y	Logical AND memory with accumulator indexed by Y	AND TAG(A),Y
ASL	OPER	Shift left one bit (memory or accumulator)	ASL TAG(A)
ASL	OPER,X	Shift left one bit (memory or accumulator) indexed by X	ASL TAG(A),X
BIT	OPER	Test bits in memory with accumulator	BIT TAG(A)
CMP	OPER	Compare memory with accumulator	CMP TAG(A)
CMP	OPER,X	Compare memory with accumulator indexed by X	CMP TAG(A),X
CMP	OPER,Y	Compare memory with accumulator indexed by Y	CMP TAG(A),Y
CPX	OPER	Compare memory and index X	CPX TAG(A)
CPY	OPER	Compare memory and index Y	CPY TAG(A)
DEC	OPER	Decrement memory by one	DEC TAG(A)
DEC	OPER,X	Decrement memory by one indexed by X	DEC TAG(A),X

MNEMONIC/OPERAND		DESCRIPTION	EXAMPLE
EOR	OPER	Exclusive OR memory with accumulator	EOR TAG(A)
EOR	OPER,X	Exclusive OR memory with accumulator indexed by X	EOR TAG(A),X
EOR	OPER,Y	Exclusive OR memory with accumulator indexed by Y	EOR TAG(A),Y
INC	OPER	Increment memory by one	INC TAG(A)
INC	OPER,X	Increment memory by one indexed by X	INC TAG(A),X
JMP	OPER	Jump to new location	JMP TAG(A)
JSR	OPER	Jump to new location saving return address	JSR TAG(A)
LDA	OPER	Load the accumulator with memory	LDA TAG(A)
LDA	OPER,X	Load the accumulator with memory indexed by X	LDA TAG(A),X
LDA	OPER,Y	Load the accumulator with memory indexed by Y	LDA TAG(A),Y
LDX	OPER	Load Index X with memory	LDX TAG(A)
LDX	OPER,Y	Load Index X with memory indexed by Y	LDX TAG(A),Y
LDY	OPER	Load Index Y with memory	LDY TAG(A)
LDY	OPER,X	Load Index Y with memory indexed by X	LDY TAG(A),X
LSR	OPER	Shift right one bit	LSR TAG(A)
LSR	OPER,X	Shift right one bit indexed by X	LSR TAG(A),X
ORA	OPER	Logical OR memory with accumulator	ORA TAG(A)
ORA	OPER,X	Logical OR memory with accumulator indexed by X	ORA TAG(A),X
ORA	OPER,Y	Logical OR memory with accumulator indexed by Y	ORA TAG(A),Y
ROL	OPER	Rotate one bit left	ROL TAG(A)
ROL	OPER,X	Rotate one bit left indexed by X	ROL TAG(A),X
ROR	OPER	Rotate one bit right	ROR TAG(A)
ROR	OPER,X	Rotate one bit right indexed by X	ROR TAG(A),X
SBC	OPER	Subtract memory from accumulator with borrow	SBC TAG(A)
SBC	OPER,X	Subtract memory from accumulator with borrow indexed by X	SBC TAG(A),X
SBC	OPER,Y	Subtract memory from accumulator with borrow indexed by Y	SBC TAG(A),Y
STA	OPER	Store accumulator in memory	STA TAG(A)

MNEMONIC/OPERAND		DESCRIPTION	EXAMPLE
STA	OPER,X	Store accumulator in memory indexed by X	STA TAG(A),X
STA	OPER,Y	Store accumulator in memory indexed by Y	STA TAG(A),Y
STX	OPER	Store index X in memory	STX TAG(A)
STY	OPER	Store index Y in memory	STY TAG(A)

TAG = absolute reference, relocatable reference, or global reference

In the preceding tables, TAG references may be one of three types.

ABSOLUTE REFERENCE:

When defining/accessing defined data, all references are resolved at assembly time and are displayed in the assembly listing.

GLOBAL REFERENCE:

When defining/accessing global data, the list file outputs a zero byte, followed by a "G". The linker will select from its symbol table the global variable and resolve the low or high byte.

RELOCATABLE REFERENCE:

When defining/accesing relocatable data, the assembler list always shows the low byte of the reference, followed by a " ′ " mark. However, the entire relocation constant is transferred to the linker. The linker (ELINK2.SAV) will locate the actual value and select the appropriate low or high byte.

The ability to force zero page and zero page indexed addressing modes has been implemented (for those instructions which it is legal). The instructions are described in the table below.

MNEMONIC/OPERAND		DESCRIPTION	EXAMPLE
ADC	ZPAGE	Add with carry memory to accumulator	ADC TAG(Z)
ADC	ZPAGE,X	Add with carry memory to accumulator indexed by X	ADC TAG(Z),X
AND	ZPAGE	Logical AND memory with accumulator	AND TAG(Z)
AND	ZPAGE,X	Logical AND memory with accumulator indexed by X	AND TAG(Z),X
ASL	ZPAGE	Shift left one bit (memory or accumulator)	ASL TAG(Z)
ASL	ZPAGE,X	Shift left one bit (memory or accumulator) indexed by X	ASL TAG(Z),X
BIT	ZPAGE	Test bits in memory with accumulator	BIT TAG(Z)
CMP	ZPAGE	Compare memory with accumulator	CMP TAG(Z)
CMP	ZPAGE,X	Compare memory with accumulator indexed by X	CMP TAG(Z),X
CPX	ZPAGE	Compare memory and index X	CPX TAG(Z)
CPY	ZPAGE	Compare memory and index Y	CPY TAG(Z)
DEC	ZPAGE	Decrement memory by one	DEC TAG(Z)
DEC	ZPAGE,X	Decrement memory by one indexed by X	DEC TAG(Z),X
EOR	ZPAGE	Exclusive OR memory with accumulator	EOR TAG(Z)
EOR	ZPAGE,X	Exclusive OR memory with accumulator indexed by X	EOR TAG(Z),X
INC	ZPAGE	Increment memory by one	INC TAG(Z)
INC	ZPAGE,X	Increment memory by one indexed by X	INC TAG(Z),X
LDA	ZPAGE	Load the accumulator with memory	LDA TAG(Z)
LDA	ZPAGE,X	Load the accumulator with memory indexed by X	LDA TAG(Z),X
LDX	ZPAGE	Load Index X with memory	LDX TAG(Z)
LDX	ZPAGE,Y	Load Index X with memory indexed by Y	LDX TAG(Z),Y
LDY	ZPAGE	Load Index Y with memory	LDY TAG(Z)
LDY	ZPAGE,X	Load Index Y with memory indexed by X	LDY TAG(Z),X
LSR	ZPAGE	Shift right one bit	LSR TAG(Z)
LSR	ZPAGE,X	Shift right one bit with indexed by X	LSR TAG(Z),X

MNEMONIC/OPERAND		DESCRIPTION	EXAMPLE
ORA	ZPAGE	Logical OR memory with accumulator	ORA TAG(Z)
ORA	ZPAGE,X	Logical OR memory with accumulator indexed by X	ORA TAG(Z),X
ROL	ZPAGE	Rotate one bit left	ROL TAG(Z)
ROL	ZPAGE,X	Rotate one bit left indexed by X	ROL TAG(Z),X
ROR	ZPAGE	Rotate one bit right	ROR TAG(Z)
ROR	ZPAGE,X	Rotate one bit right indexed by X	ROR TAG(Z),X
SBC	ZPAGE	Subtract memory from accumulator with borrow	SBC TAG(Z)
SBC	ZPAGE,X	Subtract memory from accumulator with borrow indexed by X	SBC TAG(Z),X
STA	ZPAGE	Store accumulator in memory	STA TAG(Z)
STA	ZPAGE,X	Store accumulator in memory indexed by X	STA TAG(Z),X
STX	ZPAGE	Store index X in memory	STX TAG(Z)
STX	ZPAGE,Y	Store index X in memory indexed by Y	STX TAG(Z),Y
STY	ZPAGE	Store index Y in memory	STY TAG(Z)
STY	ZPAGE,X	Store index Y in memory indexed by X	STY TAG(Z),X

TAG = absolute reference, relocatable reference, or global reference

If TAG is absolute and greater than FF, an error will be indicated at assembly time.

If TAG is relocatable or global and greater than FF, an error will be indicated at link time.

The following lists additional information for programming with the cross assembler.

1.) If address zero is referenced, the instruction will be assembled with the extended addressing mode instead of the direct addressing mode. To have the instruction assemble address zero as only a byte, you must use the force zero page syntax.

2.) Addressing references in zero page must be predefined.

3.) The use of complex forward references should be avoided as they may result in phase errors. However, when a complex forward reference is made, it can be forced absolute, thereby avoiding phasing errors.

ex.) LDX TAG+3(A)

where TAG is a forward reference.

EMULOGIC, INC.
3 Technology Way
Norwood, MA 02062-3978
Tel: (617) 329-1031
Telex: 710-336-5908

EMULOGIC®

6502

User's Guide Supplement

Order Number: CSU-3006-01

6502 USER'S GUIDE SUPPLEMENT

This document supplements the ECL-3211 System User's Guide by providing operational information specific to the emulation of 6502 and compatible microprocessors. This document describes special set-up procedures, conditions, and limitations to be noted when emulating the 6502. It is assumed here that the reader has read the User's Manual and is already familiar with the details of the 6502. Ready access to the technical literature is a plus.

This supplement covers five general areas.

*** INSTALLATION ***

System installation instructions will be found in the User's Manual.

*** INITIALIZATION ***

Type on the keyboard "RUN L01500" to load the Emulation Software into the ECL-3211. (The "RUN" command is discussed in the User's Guide.) Note that a user can use the Operating System's RENAME function to give the file a name the user would prefer. Additionally, a Command File can be created which can invoke L01500.

There are no special initialization instructions for the 6502.

SYSTEM DISPLAY

These are seen on the top half of the display when using the Emulation Software. All of these registers and flags can be loaded with user prefered values with the SET Command or ALTER mode as described in the User Manual or HELP file.

***	*** DESCRIPTION ***	
PC	Program Counter	16 bits/4 hex digits
X	Index Register X	8 bits/2 hex digits
S	Status Register	8 bits/2 hex digits
A	Accumulator	8 bits/2 hex digits
Y	Index Register Y	8 bits/2 hex digits
P	Stack Pointer	8 bits/2 hex digits
N	Negative Result (Sign)	Status bit 7
V	Overflow	Status bit 6
-	--	--
B	Break	Status bit 4
D	Decimal Mode (BCD)	Status bit 3
I	Interrupt enable/disable	Status bit 2
Z	Zero	Status bit 1
C	Carry	Status bit 0

TRACE DISPLAY

Note: Low=0 High=1 Don't Care=X

"1" and "0" refer to ELECTRICAL, NOT logical levels; though for ECL-3211 functions logical and electrical coincide.

These are seen when examining the Trace.

***	*** DESCRIPTION ***
IQ	Interrupt Request-L
NM	Non-Maskable Interrupt-L
RS	Reset-L
RY	Ready
RD	Read/Write-L
SO	Set Overflow
BA	Bus Available; generated by the ECL-3211, a "0" indicates that the Data Bus is Tristate.
SY	Sync

BREAKPOINT DISPLAY Note: Low=0 High=1 Don't care=X

These are seen when examining or setting Breakpoints.

E0-E7	Pod External Input 0-7
SW1	Logical Switch 1 External Trigger 1
SW2	Logical Switch 2 External Trigger 2
SW3	Logical Switch 3
SW4	Logical Switch 4
ROM	ROM access; "1" means trigger on a read from an address designated as ROM.
SYNC	SYNC; A "1" indicates the fetch of the first byte of an Op Code as a Breakpoint Condition.
CO1	"1" selects Counter 1 expired
CO2	"1" selects Counter 2 expired
ADDR	Program Counter; 16 bits
DATA	Data; 8 bits
IRQ	Interrupt Request-L
NMI	Non-Maskable Interrupt-L
RES	Reset-L
RDY	Ready
READ	Read/Write-L
SO	Set Overflow
BA	Bus Available; a "0" selects as a Breakpoint condition the Data Bus being Tristate.
PH=JMP	The 6502 Pod performs Phantom Jumps as a Breakpoint Action.

*** UNIQUE FEATURES

L01500

The file name for the Emulation Software is L01500. It is accessed through the Operating System hosted by the ECL-3211's CPU.

RESET

The ECL-3211's RESET command resets the 6502 Pod only, and does not reset the Target. A Reset generated by the Target has effect during emulation only.

NO TARGET

Not having the 6502 Pod deployed in a target will not affect the operation of the Emulator in any way, assuming the user does not try to access resources in the Target.

MAX FREQ

The maximum frequency of operation is 2 Megahertz for both Target and ECL-3211 memory.

DEC INTERNAL

The 6502 Pod cannot operate using the bank of memory termed in the User's Guide as DEC Internal.

PHANTOMS

The 6502 Pod performs Phantom Jumps. Naturally, provision must be made to return to the original code path if that is desired by the user.

There are two important qualifications to their use:

1) The instruction immediately preceding the Phantom Jump must perform a Prefetch. This means that a Phantom Jump cannot be inserted after a 2 or 3 byte intruction.

2) The address desired as a Breakpoint condition must be defined as an address value (ADDR) rather than a Program Counter value (PC).

To illustrate, consider the following examples of defining Breakpoint 4 as a Phantom Jump to address 5050:

Given this code segment--

ADDR	INSTR	DATA
****	*****	****
.	.	.
.	.	.
A500	DEX	CA
A501	LDX OFF	A2
A502		FF
A503	INX	E8
.	.	.
.	.	.

a) Typing "BR 4 PH=5050/ADDR=A501" will be successful. The Phantom Jump is being inserted by the Prefetch of a single byte instruction at a location defined as ADDR rather than PC (Program Counter).

b) Typing "BR 4 PH=5050/PC=A501" will fail. The address where the Phantom Jump is intended to be inserted is defined as PC, a Program Counter value.

c) Typing "BR 4 PH=5050/ADDR=A503" will fail. The preceding instruction, LDX, is a 2 byte instruction and does not Prefetch.

TRACE DATA CAPTURE

If the Trace has been turned on, it takes a "snapshot" of conditions during each Machine Cycle when the conditions are valid. For example, the Data bus is sampled when it contains valid Data. Address information is sampled when there is a valid Address on the bus. Control signals are sampled at the same time as the Data unless they must be sampled at a different point in the Machine Cycle. (The Trace is turned on by defining a Breakpoint with conditions that will be met and an Action statement including Set Trace, as described in the User's Guide and HELP file.)

Instructions are disassembled in the Trace as they appeared on the Data bus when they were fetched.

Note that the External Inputs are not sampled simultaneously in a Machine Cycle. External Inputs 0-3 are sampled during the valid address time of a Machine Cycle and External Inputs 4-7 are sampled during valid data time.

BREAKPOINT ACTION

Defined Breakpoint conditions are tested and resolved prior to the end of the Machine Cycle. Any Breakpoint Actions for a Breakpoint with conditions that have been met in a Machine Cycle commence at the completion of that Machine Cycle.

CLOCK

The Emulator provides two sources of Clock signals for the 6502 Pod, the ECL-3211 and the Target circuit. Internal Clock has a guarantee of 100 Kilohertz resolution.

--- External ---

External Clock is the mode in which the Target Circuit provides the clock. Since it is buffered in the Pod with TTL logic, the clock signal must be TTL driven or equivalent. Do not clock the Pod with a Crystal/RC Network circuit.

Type "FREQ EXT" to select this mode.

--- Internal ---

Internal Clock is the mode in which the 6502 Pod is clocked by the Emulator. The clocking signal taken from the Target is not used. $\phi 1$ (pin 3) and $\phi 2$ (pin 39) are still active.

Type "FREQ xxxx" to select the Internal Clock mode. "xxxx" is the value of the frequency in units of Kilohertz. There is no need to specify "Internal" at any point.

NOT EMULATING

When the ECL-3211 is not in Emulation mode, the signals from the 6502 Pod to the Target have the following status:

A0-A15	Active
D0-D7	Tri-state
SYNC	Active
SO	Active
Ø2	Active
Ø1	Active
Ø0	Active; ignored if Clock Internal selected
IRQ-L	High; ignored by Pod
NMI-L	High; ignored by Pod
RES-L	High; ignored by Pod
RDY	High; ignored by Pod
R/W-L	High; ignored by Pod

*** ELECTRICAL (DC) CHARACTERISTIC

Signal	Buffer Type	Output Drive		Input Load		Delay, additional	Termination, pull-up R
	74xxx	High mA	Low mA	High mA	Low mA	nSec typical	ohms
A0-A15	LS245	-15.0	24.0	--	--	12	--
D0-D7	LS245	-15.0	24.0	0.02	0.2	8	--
SYNC	LS04	-0.4	8.0	--	--	13	--
R/W-L	F04	-1.0	2.0	--	--	7	--
ϕ2	LS04	-0.4	8.0	--	--	13	--
ϕ1	LS04	-0.4	8.0	--	--	20	--
SO	LS04	--	--	0.02	-0.4	20	--
ϕ0	LS00	--	--	0.02	-0.4	20	--
IRQ-L	LS32	--	--	0.02	-0.4	14	--
NMI-L	LS32	--	--	0.02	-0.4	14	--
RES-L	LS32	--	--	0.02	-0.4	24	--
RDY	LS32	--	--	0.02	-0.4	14	--
HALT-L	LS32	--	--	0.02	-0.4	14	--