KUE-CHIP2 教育用ボード
リファレンスマニュアル

神原弘之 (京都高度技術研究所)
越智裕之，澤田宏，浜口清治（京都大学工学部情報工学教室）
岡田和久（京都大学工学部電子工学教室）
上嶋明（立命館大学理工学部情報工学教室）
安浦寛人（九州大学大学院総合理工学研究科）

1993 年 6 月 25 日 (金)
Version 1.11

目次

第 1 章

KUE-CHIP2 教育用ボードの概要

KUE-CHIP2(Kyoto University Education-CHIP 2) の動作を調べる実験は、KUE-CHIP2 を搭載した KUE-CHIP2 教育用ボードの上で行なう。 KUE-CHIP2 教育用ボードは図 1.1に示すように観測や実験のためのスイッチ、表示用 LED、コネクタを備えている。また、 KUE-CHIP2 のほかに、これらのユーザーインタフェースのための多くの IC も搭載している。

KUE-CHIP2 教育用ボードは、実験を円滑に行なうため、ボード上に 512 バイトのメモリバンクを 16 個搭載している。 MEM スイッチが INT に選択された時は KUE-CHIP2 内蔵のメモリ (512 バイト) が使われ、 EXT に選択された時はボード上の 16 バンクの内の一つが用いられる。これらのメモリバンクはすべて独立で、ADDRESS C ~ 9 スイッチによってその 1 つを選べるようになっている。即ち、KUE-CHIP2 のメモリアドレスは 9 ビットであるが、ボード上では 13 ビットのアドレスバスが使われる (第 2章において、この機能に関連する項目には * を付けてある)。この外部メモリバンクの他に 16 バンク (1 バンクは 512 バイト) の ROM も搭載できる。 ROM に保存してあるプログラムやデータを内部メモリバンクや外部メモリバンクへコピーするのに便利な機能も用意している (この機能に関連する項目には ** を付けてある)。

また、KUE-CHIP2 教育用ボードには、KUE-CHIP2 と同じ外部仕様の回路をボードの外部に製作し、その動作を KUE-CHIP2 教育用ボードのユーザーインタフェースを利用して観測するための機能も付加されている (この機能に関連する項目には *** を付けてある)。

上に挙げたような KUE-CHIP2 教育用ボードのさまざまな機能の詳細は第 2章で述べるが、KUE-CHIP2 教育用ボードを初めて使用される方には、まず第 3章を参考に、 KUE-CHIP2 および、KUE-CHIP2 教育用ボードの操作法に馴染まれることをお勧めする。また、いくつかの実用的なプログラムをサンプルとして第 4章にあげておく。

表 1.1: KUE-CHIP2 教育用ボードの諸元

クロック	0.033Hz ~ 1.0MHz 可変 (16 段階)
表示	7 セグメント LED3 桁 + 9 ビット LED(アドレス)
	7 セグメント LED2 桁 + 8 ビット LED(データ)
入力	8 ビットトグルスイッチ
コネクタ	8 ビットパラレル入力 · 出力ポート
	KUE-CHIP2 入力 · 出力信号 ***
電源	5V 1A 以下
寸法	220 × 300 × 60(mm)

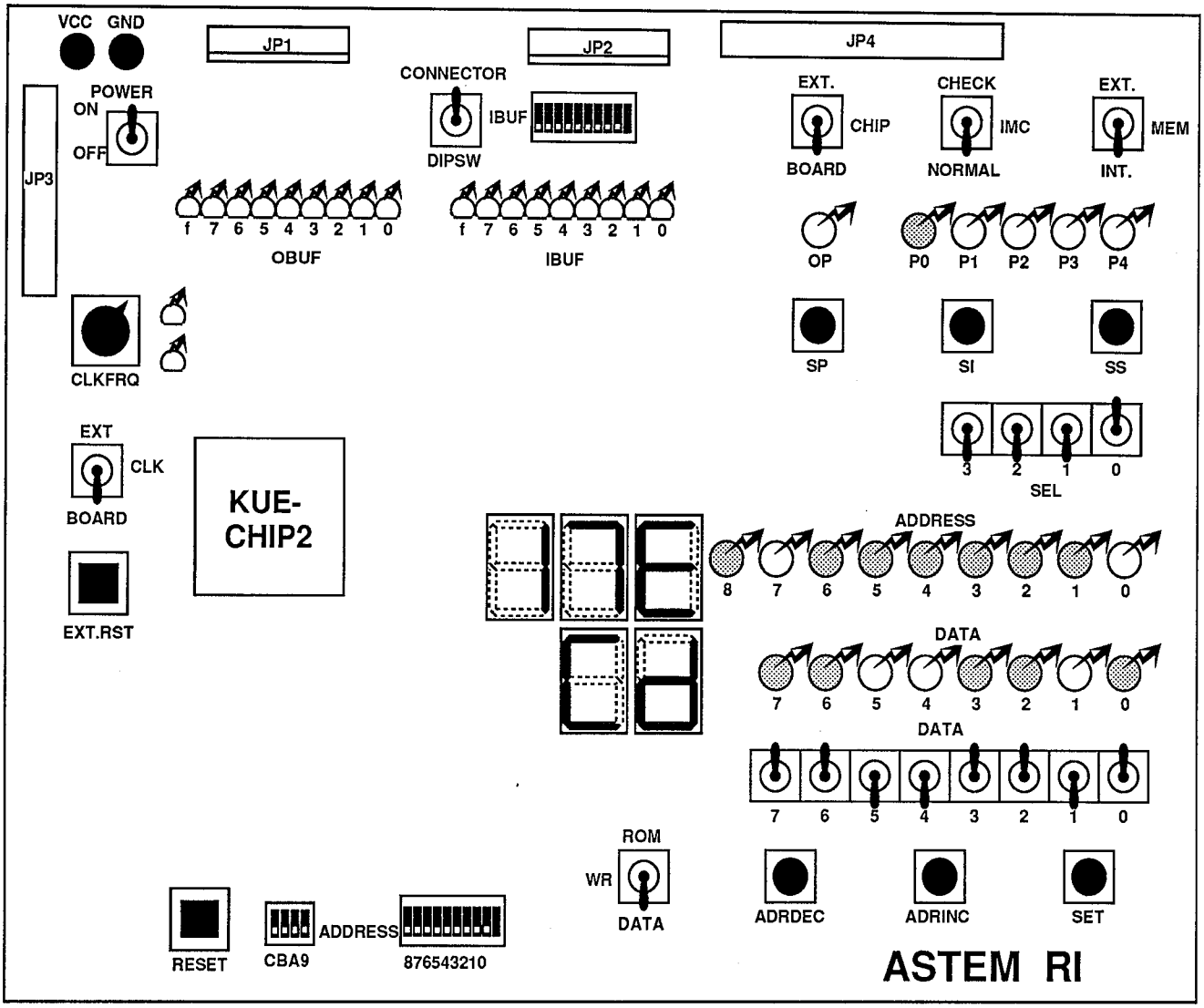


図 1.1: KUE-CHIP2 教育用ボード

第 2 章

KUE-CHIP2 教育用ボードの機能

この章では、KUE-CHIP2 ボードの機能を大きく表示機能、スイッチの機能、ターミナルの機能に分け、それぞれについて詳細を述べる。

2.1 表示機能

KUE-CHIP2 教育用ボードは KUE-CHIP2 の内部状態や外部回路の状態を観測するために以下のような表示用 LED(ランプ) を備えている。また、各 LED のボード上における配置は図 2.1を参照されたい。

- OP (LED)

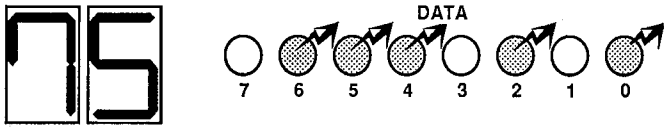
点灯	プログラム、命令を実行中

- P0~P4 (LED×5)

点灯	実行クロックフェーズ

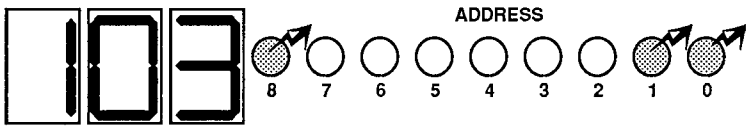
- DATA 7~0 (LED×8 と 7segment×2)

  SEL 3~0 で指定されたメモリ、レジスタ、フラグなどの値を 2 進 (LED)、16 進 (7seg) で表示する



- ADDRESS 8~0 (LED×9 と 7segment×3) *

  アドレスバス:AB<8:0> の値を 2 進 (LED)、16 進 (7seg) で表示する



- IBUF (LED×9)

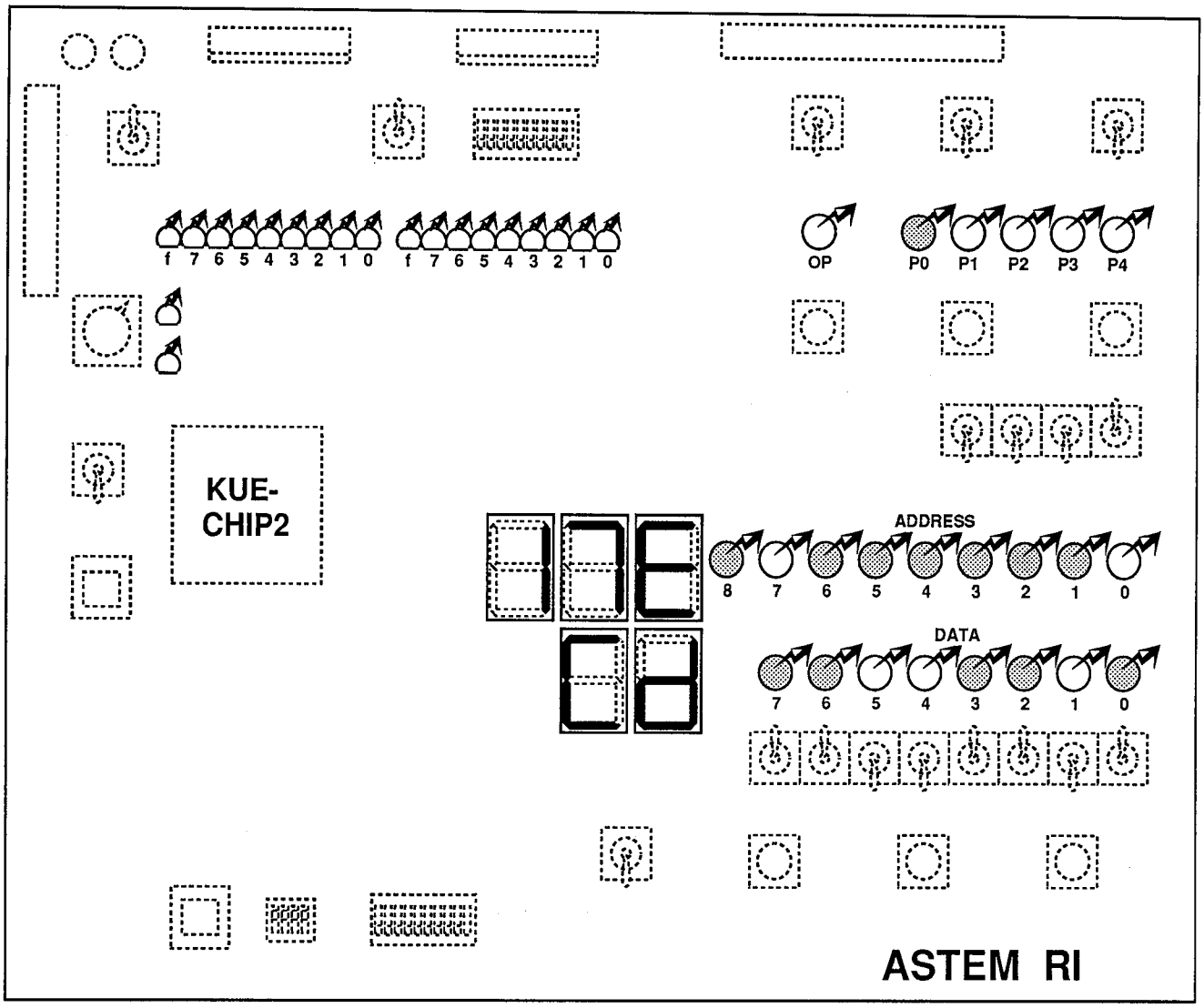
IBUF

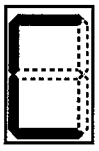
f	入力ポート IBUF のフラグ	
7	入力ポート IBUF のデータの	最上位ビット (MSB)
6		第 6 ビット
5		第 5 ビット
4		第 4 ビット
3		第 3 ビット
2		第 2 ビット
1		第 1 ビット
0		最下位ビット (LSB)

- OBUF (LED×9)

OBUF

f	出力ポート OBUF のフラグ	
7	出力ポート OBUF のデータの	最上位ビット (MSB)
6		第 6 ビット
5		第 5 ビット
4		第 4 ビット
3		第 3 ビット
2		第 2 ビット
1		第 1 ビット
0		最下位ビット (LSB)





7seg LED

LED

LED small

図 2.1: KUE-CHIP2 教育用ボードの LED

2.2 スイッチの機能

以下に KUE-CHIP2 教育用ボード上のスイッチの機能について説明する。また、各スイッチのボード上における配置は図 2.2を参照されたい。

- POWER (トグル SW)

ON	電源を投入する
OFF	電源を切る

- RESET (プッシュ SW)

PUSH	KUE-CHIP2 内部のすべてのレジスタ、カウンタ、フラグ、その他のフリップフロップ の値を 0 にリセットする IBUF、OBUF のデータとフラグを 0 にリセットする

- CLKFRQ (16 接点ロータリー SW)

  KUE-CHIP2 の動作するクロック周波数を設定する

CLKFLQ ==	0	1	MHz
	1	333	kHz
	2	100	kHz
	3	33	kHz
	4	10	kHz
	5	3	kHz
	6	1	kHz
	7	333	Hz
	8	100	Hz
	9	33	Hz
	10	10	Hz
	11	3.3	Hz
	12	1	Hz
	13	0.33	Hz
	14	0.10	Hz
	15	0.033	Hz

- CLK (トグル SW) ***

EXT	ボード外の KUE-CHIP2 だけにコネクタ (JP3) を通してクロックを供給する
中立	ボード上の KUE-CHIP2 ならびにコネクタの両方にクロックを供給する
BOARD	ボード上の KUE-CHIP2 だけにクロックを供給する

- EXT.RST(プッシュ SW) ***

PUSH	ボード外の KUE-CHIP2 をリセットする

- ADDRESS C~9 (4 ビットディップ SW) *

アドレスバスの上位 4 ビットを指定する。第 C ビットの変化は 7segment に表示されない。

C	アドレスバスの	第 C ビット (MSB)	
B		第 B ビット	
A		第 A ビット	
9		第 9 ビット	を指定する

- ADDRESS 8~0 (9 ビットディップ SW)

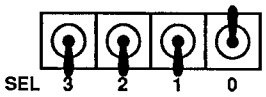
IMC = Check の時のアドレスバスの下位 9 ビットを指定する

8	アドレスバスの	第 8 ビット
7		第 7 ビット
6		第 6 ビット
5		第 5 ビット
4		第 4 ビット
3		第 3 ビット
2		第 2 ビット
1		第 1 ビット
0		第 0 ビット (LSB) を指定する。

・注 ディップ SW は 10 ビットの部品が用いられており、一番右端のビットは No Connection になっている

- SEL 3~0 (4 ビットトグル SW)

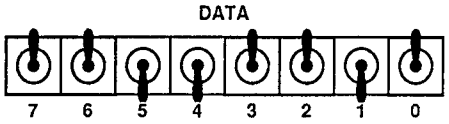
  観測、書き込みを行なうメモリ、カウンタ、レジスタ、フラグまたは、観測する制御信号線を指定する



SEL =	0	メモリのプログラム領域 (000H から 0FFH まで)
	1	メモリのデータ領域 (100H から 1FFH まで)
	2	PC
	3	FLAG
	4	ACC
	5	IX
	6	DBi
	7	DBo
	8	MAR
	9	IR
	A	制御線 imem(0:2), mar(0:4)
	B	制御線 pc(0:2), ir(0:1), ios(0:1), flag_ob
	C	制御線 obc(0:1), cfset(0:2), vfset(0:2)
	D	制御線 phasecut, halt, nfset(0:2), zfset(0:2)
	E	制御線 alu, alu_ope(0:3), reg(0:2)
	F	制御線 acc(0:3), ix(0:3)

- DATA 7~0 (8 ビットトグル SW)

  メモリ、カウンタ、レジスタに書き込むデータの値を設定する



DATA7	書き込むデータの	第 7 ビット (MSB) を指定する
DATA6		第 6
DATA5		第 5
DATA4		第 4
DATA3		第 3
DATA2		第 2
DATA1		第 1
DATA0		第 0 ビット (LSB) を指定する

- WR (トグル SW) **

  SET スイッチにより書き込むデータを指定

ROM	外部 ROM(U44) の (アドレスバスで指定された番値の) 値
DATA	8 ビットトグルスイッチ:DATA 0~7 の値

- ADRDEC (プッシュ SW)

PUSH	メモリアドレスレジスタ:MAR を 1 ディクリメントする

- ADRINC (プッシュ SW)

PUSH	メモリアドレスレジスタ:MAR を 1 インクリメントする

- SET (プッシュ SW) **

SEL で指定したメモリ、カウンタ、レジスタにトグルスイッチ WR に従い値をセットする

PUSH	WR = DATA	トグルスイッチ DATA 0~7 の値 (00H~FFH) を書き込む
	WR = ROM	(アドレスバスで指定された番値の) 外部 ROM(U44) の値 (00H~FFH) を書き込む

- SP (プッシュ SW)

PUSH	1 クロックフェーズだけ命令を実行し、停止する

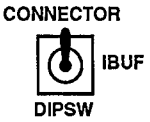
- SI (プッシュ SW)

PUSH	1 命令だけプログラムを実行し、停止する

- SS (プッシュ SW)

PUSH	停止中	プログラムを実行し、HALT 命令で停止する
	プログラムの実行中	現在実行中の命令を実行し、その次の命令の前で停止する

- IBUF (トグル SW)



CONNECTOR	入力ポート IBUF は、コネクタ JP2 からのデータ により設定
DIPSW	入力ポート IBUF は、IBUF(9 ビットディップ SW) により設定

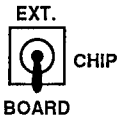
- IBUF(9 ビットディップ SW)

IBUF(トグル SW) = DIPSW の時、入力ポート IBUF のデータをセットする

f	入力ポート IBUF の	フラグをセットする
7		最上位ビット (MSB) をセットする
6		第 6
5		第 5
4		第 4
3		第 3
2		第 2
1		第 1
0		最下位ビット (LSB) をセットする

・注 ディップ SW は 10 ビットの部品が用いられており、一番右端のビットは No Connection になっている

- CHIP (トグル SW) ***



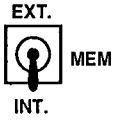
EXT	ボード外の (コネクタ JP3 と JP4 に繋がっている)KUE-CHIP2 を選択して使用
BOARD	ボード上の KUE-CHIP2 を選択して使用

- IMC (トグル SW)

アドレスバス (12 ビット幅) の下位 9 ビットの指定方法を選択する

CHECK	アドレスバスの下位 9 ビットは ADDRESS 8~0 ディップスイッチで与えられる
NORMAL	アドレスバスの下位 9 ビットは KUE-CHIP2 の MAR(メモリアドレスレジスタ) で与えられる

- MEM (トグル SW)



INT.	KUE-CHIP2 の内部メモリ (512bytes RAM) を使用する
EXT.	ボード上の外部メモリ (512bytes×16 RAM) を使用する

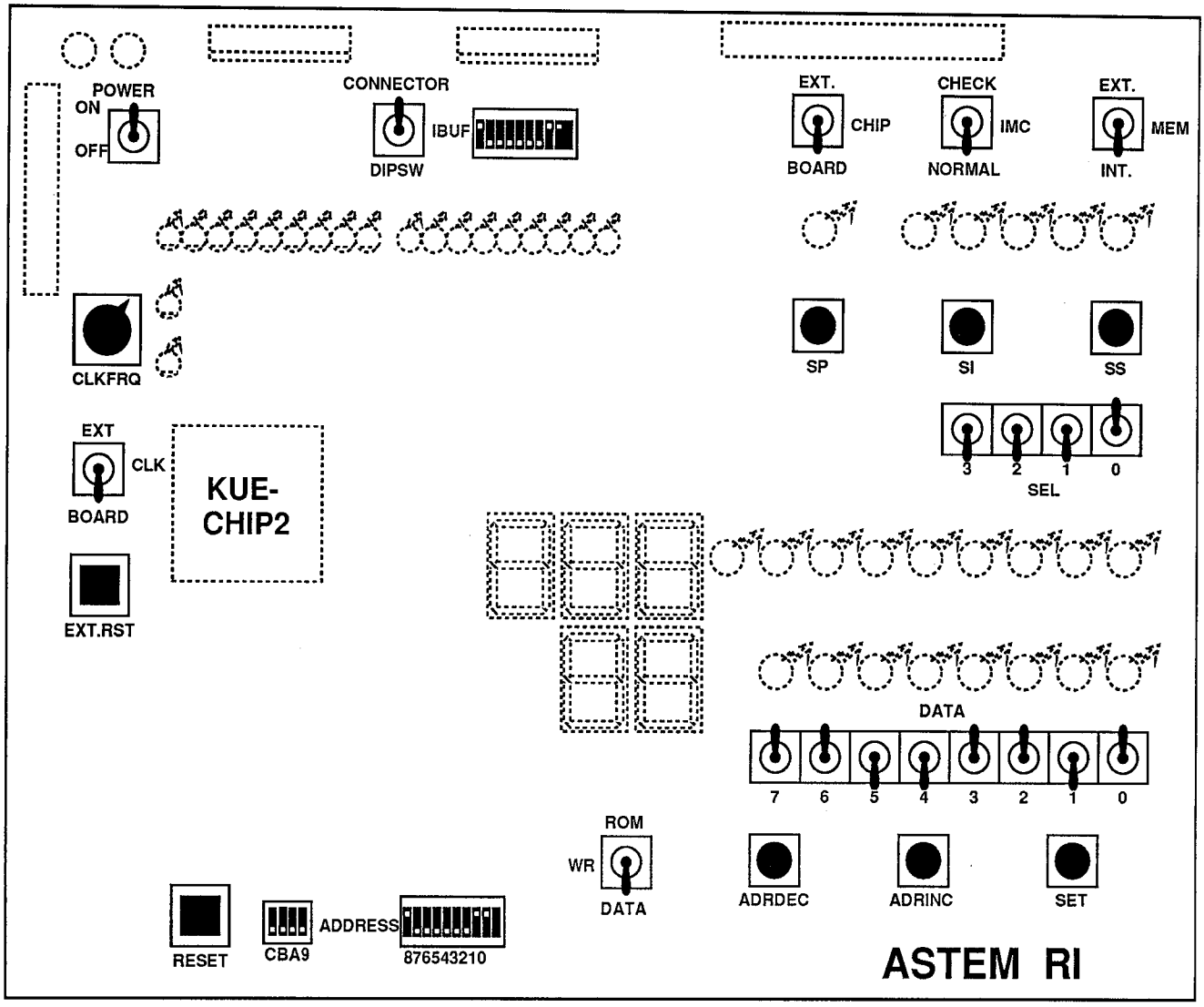


図 2.2: KUE-CHIP2 教育用ボードのスイッチ

2.3 ターミナルの機能

以下に KUE-CHIP2 教育用ボード上のターミナルの機能について説明する。また、各ターミナルのボード上における配置は図 2.3を参照されたい。

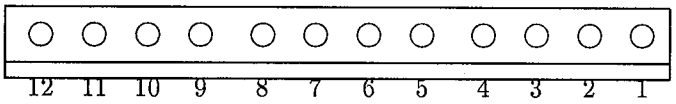
- VDD

  電源の +5.0V を接続する

- GND

  電源の GND を接続する

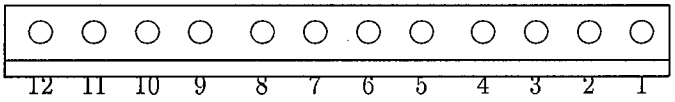
- JP1 (12×1 コネクタ)



12	OBUF のフラグを外部に出力している
11	OBUF のフラグを倒す信号を外部から入力する (‾__/‾ で倒れる)
10	OBUF のフラグの立ち下がりで 1.3μsec 幅の負のパルスを出力する
9	KUE-CHIP 教育用ボードの GND に接続されている
8	OBUF のデータの 最上位ビット (MSB) を出力する
7	OBUF 第 6
6	OBUF 第 5
5	OBUF 第 4
4	OBUF 第 3
3	OBUF 第 2
2	OBUF 第 1
1	OBUF 最下位ビット (LSB) を出力する

・注 JP1 の 11 ピンは、VDD にプルアップされている

- JP2 (12×1 コネクタ)



12	IBUF のフラグを外部に出力している
11	IBUF のフラグが KUE-CHIP2 により倒される時 1.3μsec 幅の負のパルスを出力する
10	IBUF への書き込み信号を外部から入力する (__/‾ で書き込み)
9	KUE-CHIP 教育用ボードの GND に接続されている
8	IBUF に書き込むデータの 最上位ビット (MSB) を入力
7	IBUF 第 6
6	IBUF 第 5
5	IBUF 第 4
4	IBUF 第 3
3	IBUF 第 2
2	IBUF 第 1
1	IBUF 最下位ビット (LSB) を入力する

・注 JP2 の 1~8,10 の計 9 本のピンは、VDD にプルアップされている

- JP3 (15×2 フラットケーブルコネクタ) ***

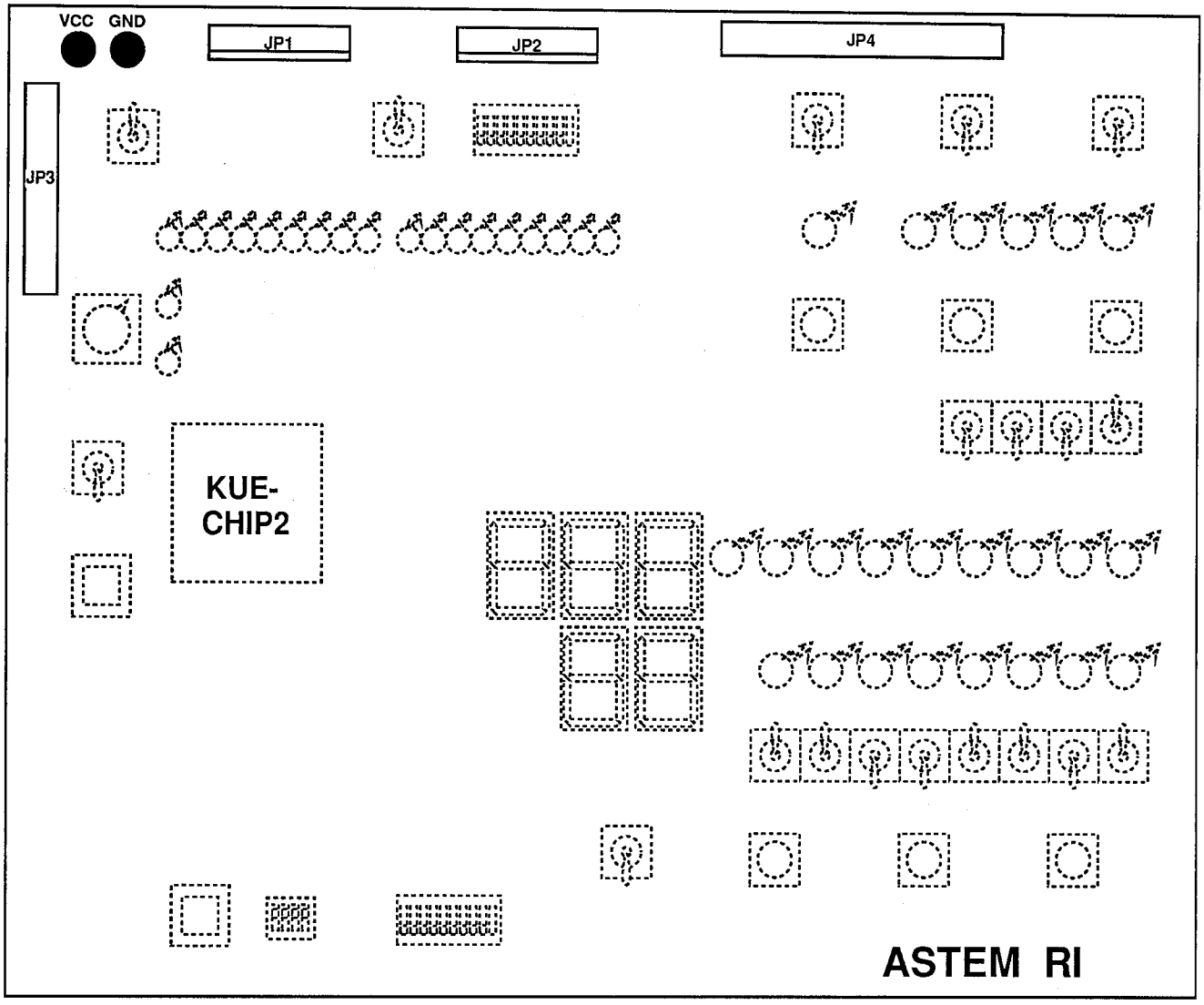
  教育用ボードの外部の KUE-CHIP2 への入力信号をボードの外部へ出力するコネクタである。それぞれの信号の意味については KUE-CHIP2 設計マニュアルの KUE-CHIP2 外部ピン仕様を参照されたい。

2	No Connection	1	VCC
4	reset	3	clock
6	si	5	sp
8	set	7	ss
10	adr_dec	9	adr_inc
12	ob_sel1	11	ob_sel0
14	ob_sel3	13	ob_sel2
16	mem_chk	15	mem_sel
18	obuf_flg_in	17	ibuf_flg_in
20	GND	19	No Connection
22	No Connection	21	No Connection
24	dbi1	23	dbi0
26	dbi3	25	dbi2
28	dbi5	27	dbi4
30	dbi7	29	dbi6

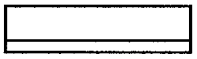
- JP4 (20×2 フラットケーブルコネクタ) ***

  教育用ボードの外部の KUE-CHIP2 の出力信号の取り込み及び双方向信号 (アドレスバス AB) の入出力を行なうコネクタである。それぞれの信号の意味については KUE-CHIP2 設計マニュアルの KUE-CHIP2 外部ピン仕様を参照されたい。

2	p0	1	op
4	p2	3	p1
6	p4	5	p3
8	ob1	7	ob0
10	ob3	9	ob2
12	ob5	11	ob4
14	ob7	13	ob6
16	mem_we	15	mem_ob
18	data_re	17	mem_re
20	ab1	19	ab0
22	ab3	21	ab2
24	ab5	23	ab4
26	ab7	25	ab6
28	NC	27	ab8
30	dbo1	29	dbo0
32	dbo3	31	dbo2
34	dbo5	33	dbo4
36	dbo7	35	dbo6
38	ibuf_flg_clr	37	ibuf_we
40	NC	39	obuf_we




terminal


flat cable

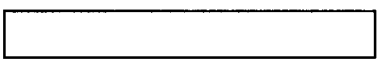

flat cable 2

図 2.3: KUE-CHIP2 教育用ボードのターミナル

第 3 章

プログラムの入力と実行

この章ではKUE-CHIP2 ボードを円滑に扱えるよう、あるプログラムを入力し実行に至るまでの操作方法を、非常に小さなプログラムを例にとり具体的に示す。

3.1 例題

下に示すプログラムはKUE-CHIP2 において、 ACC の値を IX の値回足す、即ち ACC×IX→ACC を行なう非常に簡単なプログラムである。

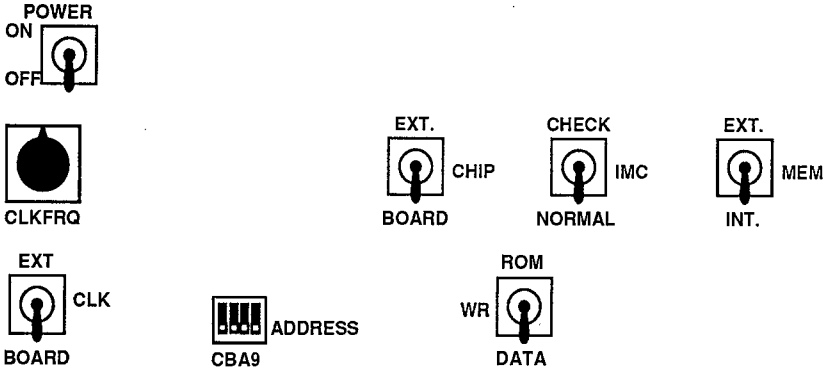
```
ADRS DATA  OPECODE
 00   75   ST  ACC,(03H)
 01   03
 02   C0   EOR ACC,ACC
 03   B5   ADD ACC,(03H)
 04   03
 05   AA   SUB IX,1
 06   01
 07   31   BNZ 03H
 08   03
 09   0F   HLT
```

3.2 例題の入力

前節で挙げた例題プログラムの入力方法を具体的に述べる。

1. それぞれのスイッチを下表のように設定する。表に無いスイッチは動かさなくて良い。

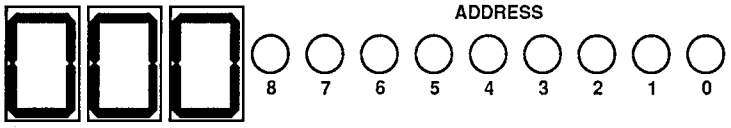
スイッチ	位置
POWER	OFF
CLKFRQ	0
CHIP	BOARD
IMC	NORMAL
MEM	INT.
CLK	BOARD
ADDRESS C~9	0000
WR	DATA



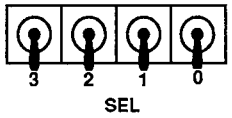
2. VCC,GND に 5V を加え、POWERsw を ON にする。

3. RESETsw を押す。

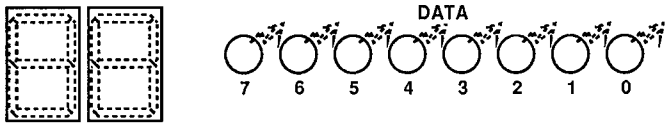
→ ADDRESS LED は 000H(000000000) を表示する (MAR が 00H にリセットされたから)。



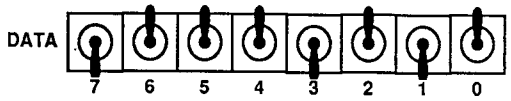
4. SELsw3~0 を 0000 にする。



→ DATA LED はメモリの 00H 番地の値 (不定) を表示する (ADDRESS LED は変化しない)。

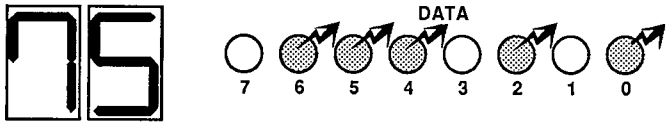


5. DATAsw を 75H(0111 0101) にする。



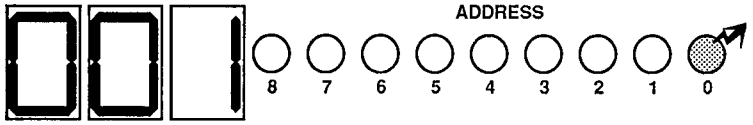
6. SET を押す。

→ DATA LED は 75H を表示する (00H 番地に 75H が書き込まれた)。

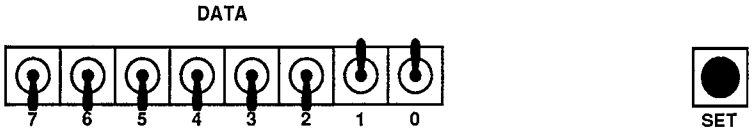


7. ADRINC を押す。

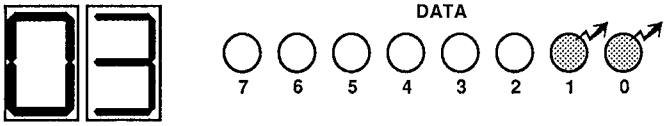
→ ADDRESS LED が 01H となる (MAR がインクリメントされた)。



8. DATAsw を 03H(0000 0011) にし、SET を押す。

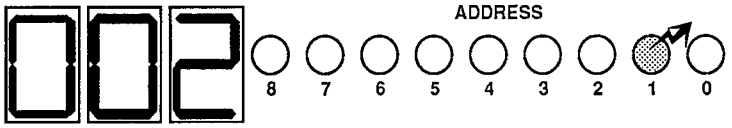


→ DATA LED は 03H を表示する (01H 番地に 03H が書き込まれた)。

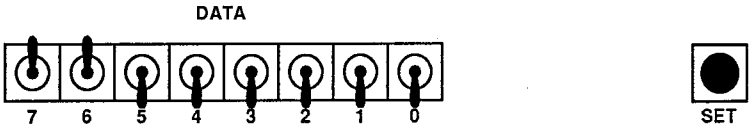


9. ADRINC を押す。

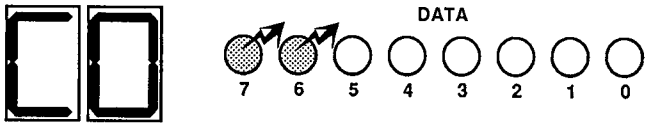
→ ADDRESS LED が 02H となる。



10. DATAsw を C0H(1100 0000) にし、SET を押す。

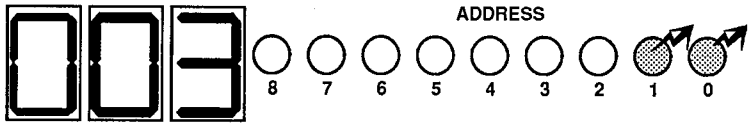


→ DATA LED は C0H を表示する (02H 番地に C0H が書き込まれた)。

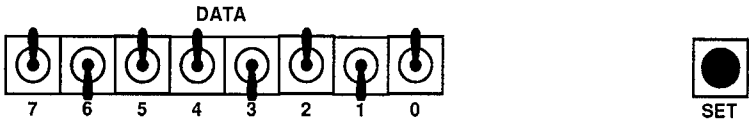


11. ADRINC を押す。

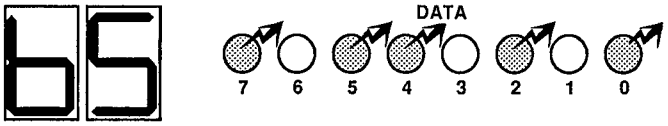
→ ADDRESS LED が 03H となる。



12. DATAsw を B5H(1011 0101) にし、SET を押す。

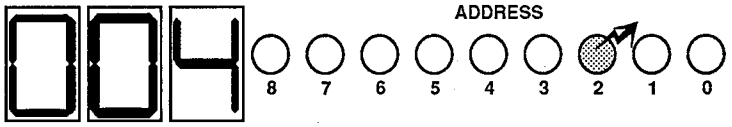


→ DATA LED は B5H を表示する (03H 番地に B5H が書き込まれた)。

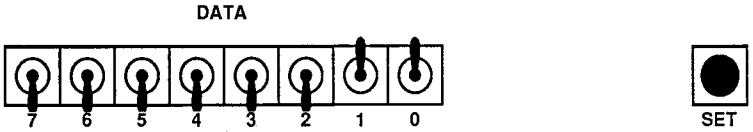


13. ADRINC を押す。

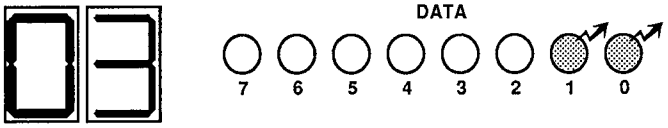
→ ADDRESS LED が 04H となる。



14. DATAsw を 03H(0000 0011) にし、SET を押す。

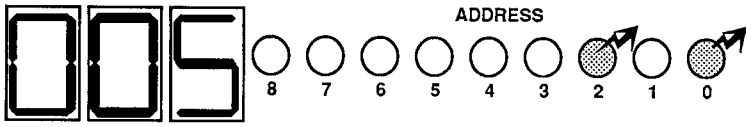


→ DATA LED は 03H を表示する (04H 番地に 03H が書き込まれた)。

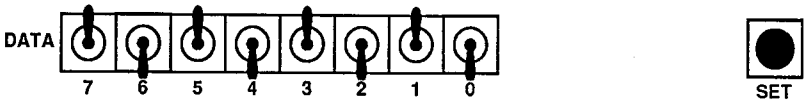


15. ADRINC を押す。

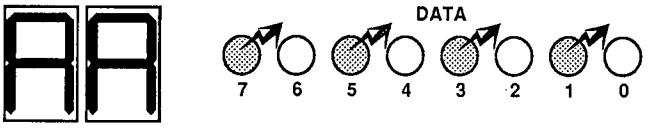
→ ADDRESS LED が 05H となる。



16. DATAsw を AAH(1010 1010) にし、SET を押す。

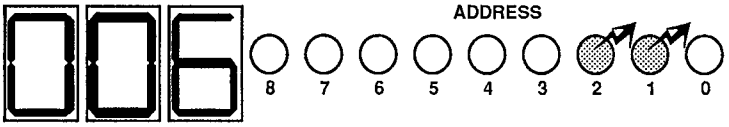


→ DATA LED は AAH を表示する (05H 番地に AAH が書き込まれた)。

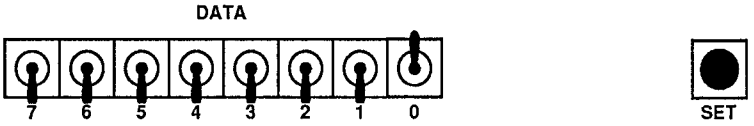


17. ADRINC を押す。

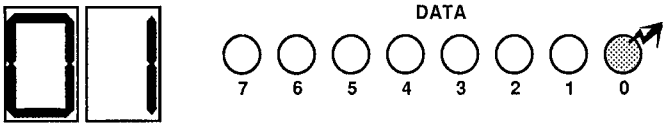
→ ADDRESS LED が 06H となる。



18. DATAsw を 01H(0000 0001) にし、SET を押す。

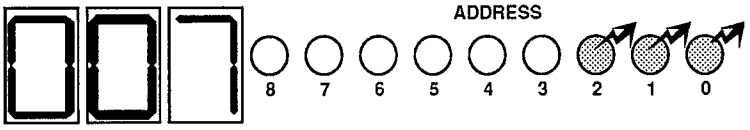


→ DATA LED は 01H を表示する (06H 番地に 01H が書き込まれた)。

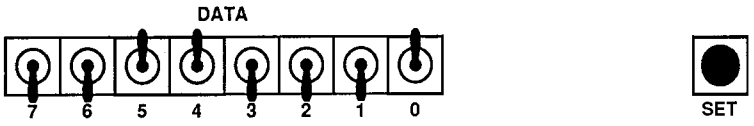


19. ADRINC を押す。

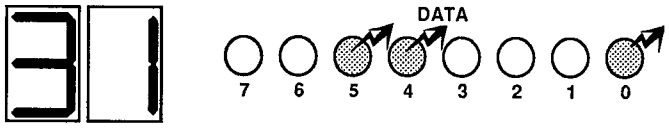
→ ADDRESS LED が 07H となる。



20. DATAsw を 31H(0011 0001) にし、SET を押す。

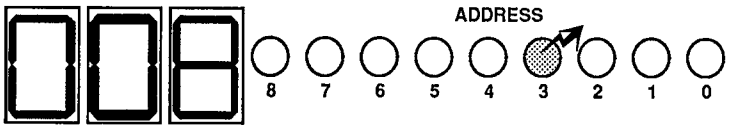


→ DATA LED は 31H を表示する (07H 番地に 31H が書き込まれた)。

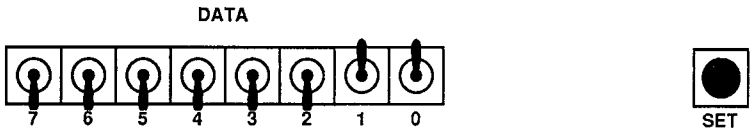


21. ADRINC を押す。

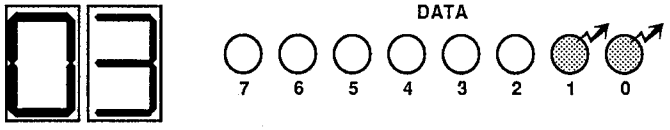
→ ADDRESS LED が 08H となる。



22. DATAsw を 03H(0000 0011) にし、SET を押す。

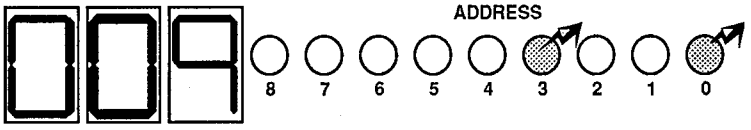


→ DATA LED は 03H を表示する (08H 番地に 03H が書き込まれた)。

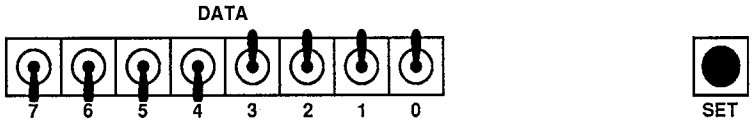


23. ADRINC を押す。

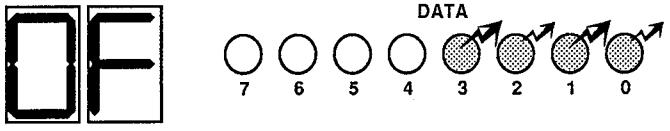
→ ADDRESS LED が 09H となる。



24. DATAsw を 0FH(0000 1111) にし、SET を押す。



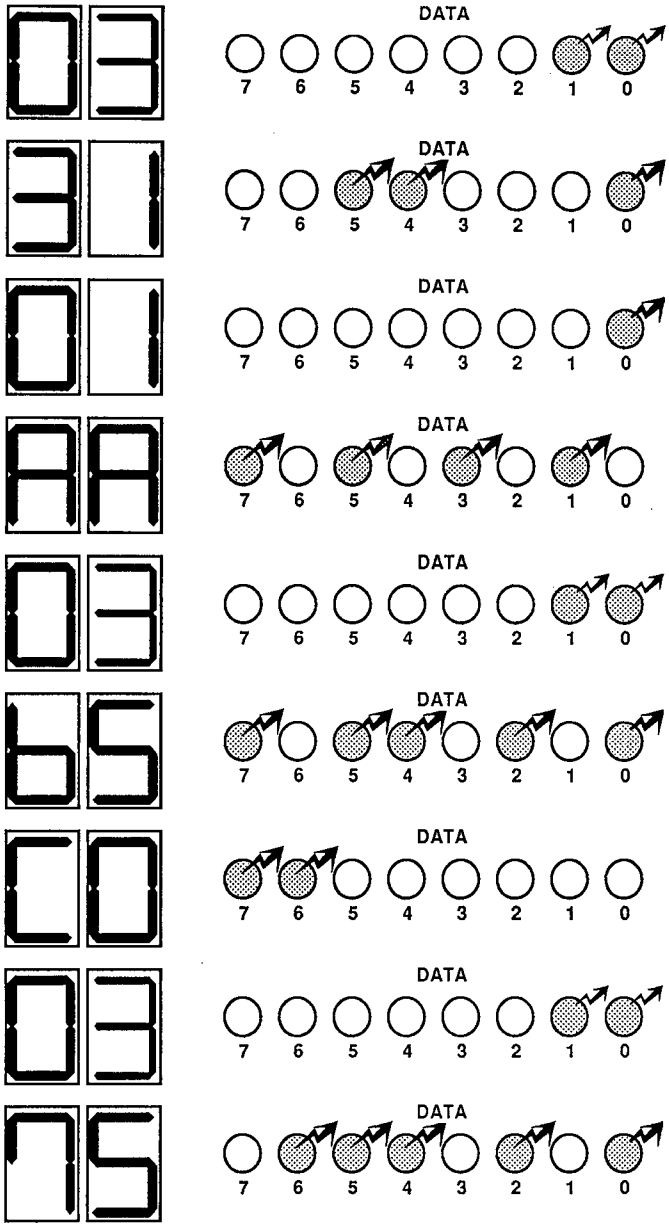
→ DATA LED は 0FH を表示する (09H 番地に 0FH が書き込まれた)。



以上でプログラムの入力は完了である。

25. 入力したプログラムの中身を確認するため、ADRDEC を 9 回押す。

→ MAR が 1 ずつデクリメントされ、08H、07H、06H、05H、04H、03H、02H、01H、00H 番地の順に、入力した命令またはオペランドが DATA LED に表示される。

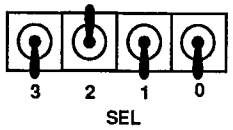


3.3 例題の実行

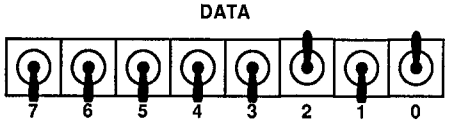
前節で入力した例題プログラムで $5 \times 4 = 20$ を計算してみる。

1. RESETsw を押す。

2. SELsw3~0 を 0100 にする。



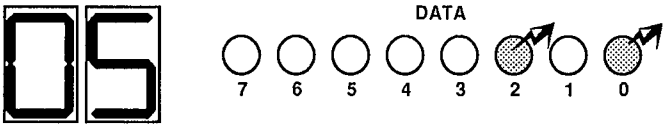
3. DATAsw7~0 を 05H(00000101) にする。



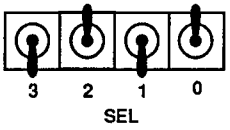
→ 表示は変化しない。

4. SETsw を押す。

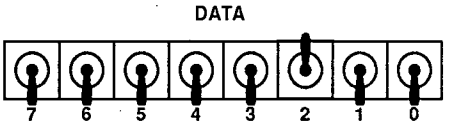
→ DATA LED は 05H を表示する (ACC に 05H が書き込まれた)。



5. SELsw3~0 を 0101 にする。



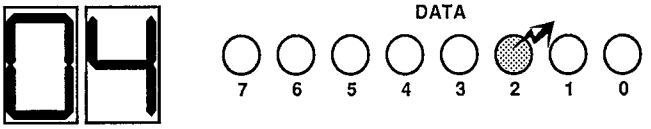
6. DATAsw7~0 を 04H(00000100) にする。



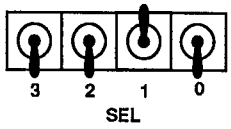
→ 表示は変化しない。

7. SETsw を押す。

→ DATA LED は 04H を表示する (IX に 04H が書き込まれた)。

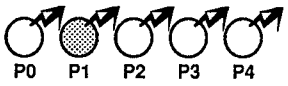


8. SELsw3~0 を 0010 にする。



9. SPsw を押す。

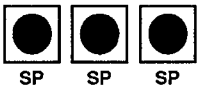
→ P1 が点灯する (00H 番地の ST ACC,(03H) 命令の P0 フェイズだけ実行された)。



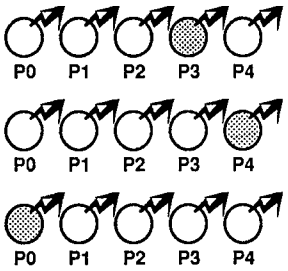
10. SPsw をもう一度押す。

→ P1 が消え P2 が点灯する (ST ACC,(03H) 命令の P1 フェイズだけ実行された)。。

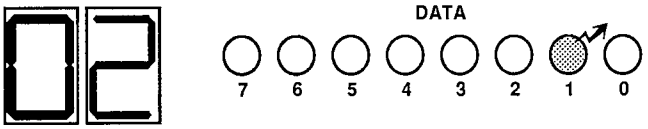
11. SPsw をもう三度押す。



→ p3,p4,p0 と順に点灯する (ST 命令は実行に 5 クロックフェーズを要する)。



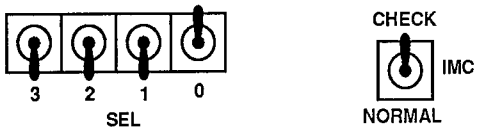
また DATA LED は 02H を表示する (ST 命令は 2 語命令で PC が 2 回インクリメントされた)。



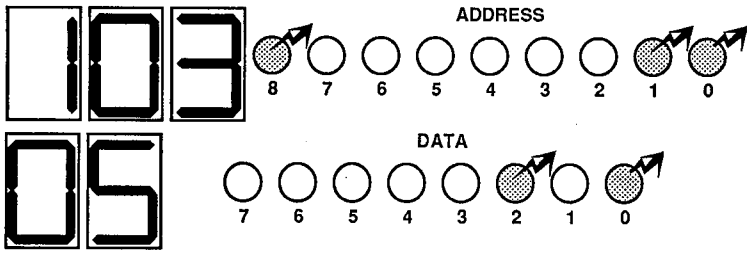
12. ADDRESSsw8~0 を 103H(100000011) にする。

→ 表示は変化しない。

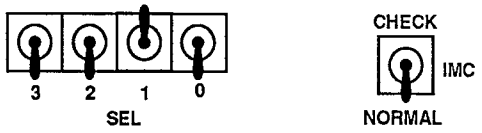
13. SELsw3~0 を 0001 にし、IMCsw を CHECK にする。



→ ADDRESS LED は 103H を表示し、DATA LED は 05H を表示する (メモリのデータ領域の 103H 番地に 05H が書き込まれた)。

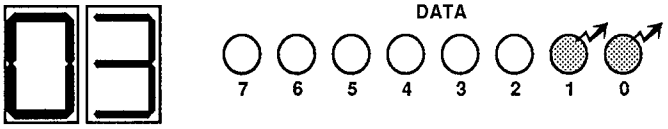


14. SELsw3~0 を 0010 に、IMCsw を NORMAL に戻す。

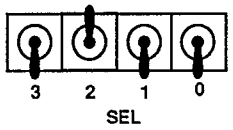


15. SIsw を一度押す。

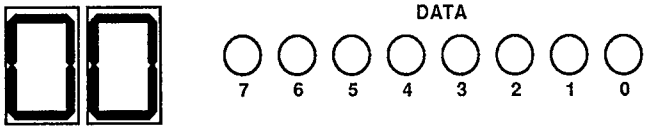
→ DATA LED は 03H を表示する (02H 番地の EOR ACC,ACC 命令だけが実行され、PC が 1 インクリメントされた)。



16. SELsw3~0 を 0100 にする。

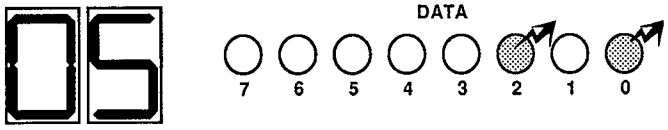


→ DATA LED は ACC のデータ 00H を表示する (EOR ACC,ACC 命令が実行されたことにより、ACC の値がクリアされた)。



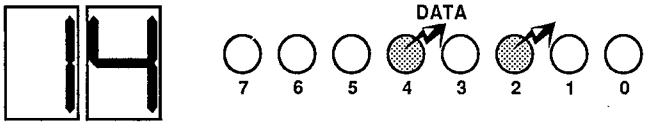
17. SIsw を一度押す。

→ DATA LED は 05H を表示する (03 番地の ADD ACC,(03H) 命令が実行された結果、ACC の値は 00H+05H=05H になった)。

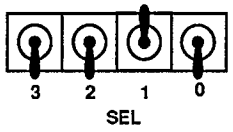


18. SSsw を一度押す。

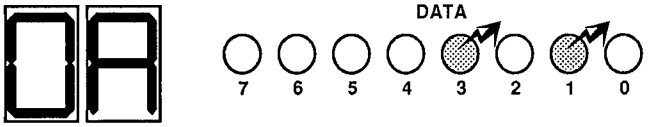
→ DATA LED に 14H(20) が表示される (ACC が 05H×04H=14H になった)。



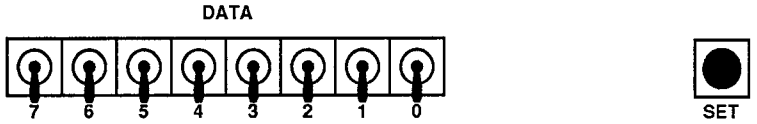
19. SELsw3~0 を 0010 にする。



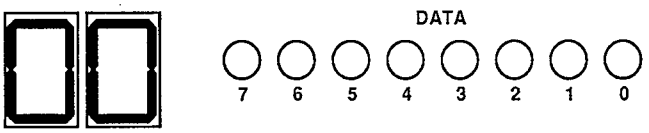
→ DATA LED は 0AH を表示する (09H 番地の HLT 命令で止まり、PC が 0AH になった)。



20. DATAsw7~0 を 00H(00000000) にし、SETsw を押す。



→ DATA LED は 00H を表示する (PC が 00H になった)。

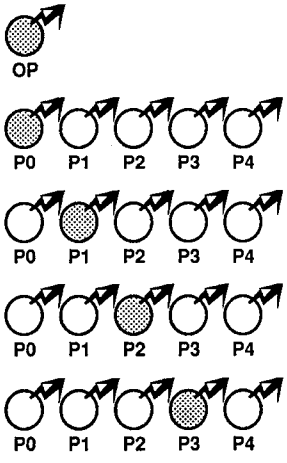


21. 3.3 節の 2から 7を行ない、ACC に 05H、IX に 04H を書き込む。

22. CLKFRQ を 12 にする。

23. SSsw を OP が点灯するまで一度押す。

→ OP が点灯し、P0~P4 が点滅する (ゆっくりプログラムを実行している)。



24. SSsw をもう一度 OP が消えるまで押す。

→ OP が消え、P0~P4 のいずれか一つが点灯する (プログラムの実行を停止した)。

ここでは、非常に簡単なプログラムで必要最小限の操作法を例示した。第 4章のサンプルプログラムのような、より複雑なプログラムを動かすのも全く同様の手順である。また、より高度な操作に関しては第 2章を参照されたい。

第 4 章

サンプルプログラム

この章では、KUE-CHIP2 のサンプルプログラムを示す。各プログラムには、簡単な説明と入出力の例を付けておいた。なお、KUE-CHIP2 では、メモリとして、256 バイトのプログラム領域の他に、256 バイトのデータ領域も利用できるが、ここでは、原則として、入出力のデータはプログラム領域においた。ただし、大量のデータを扱う一部のプログラムでは、データ領域も使用するが、この場合には、入出力例において、データ領域のアドレスを 100H から 1FFH と表記することによって区別をしている。

4.1 1からNまでの和

説明

1からN(80H番地)までの和を求め、SUM(81H番地)に格納する。なお、和は符号無し1バイトの範囲である。

入出力例

入力

```
80 : 0A (1から10までの和を求める)
```

出力

```
81 : 37 (結果:55)
```

コード

```
*** KUE-CHIP2 Assembler ver.1.0   by H.Ochi ***

* Sum from 1 to N
* Programmed by Akira Uejima, May. 5, 1992
* e-mail: uejima@mics.cs.ritsumei.ac.jp

* N
 80 :           N:      EQU             80H
* Sum(result)
 81 :           SUM:    EQU             81H

 00 :  6C 80            LD      IX,     [N]
 02 :  62 00            LD      ACC,    0
 04 :  B1       LOOP:   ADD     ACC,    IX
 05 :  AA 01            SUB     IX,     1
 07 :  33 04            BP              LOOP
 09 :  74 81            ST      ACC,    [SUM]
 0B :  0F               HLT

                        END
```

4.2 多倍長の加算

説明

インデックス修飾と、キャリーフラグを使用することによって、DATA1(80H 番地) と DATA2(90H 番地) からの、N(C0H 番地) バイトの 2 つの数を小さい番地側を MSB として加算し、ANS(A0H 番地) に格納する。

入出力例

入力

```
80 : 00 00 00 01 (データ 1)
90 : 1F FF FF FF (データ 2)
C0 : 04 (語長)
```

出力

```
A0 : 20 00 00 00 (結果)
```

コード

```
*** KUE-CHIP2 Assembler ver.1.0   by H.Ochi ***

* N byte Add (N byte + N bypte = N byte)
* Programmed by Akira Uejima, May. 4, 1992
* e-mail: uejima@mics.cs.ritsumei.ac.jp

* Data Length(byte)
 C0 :           N:      EQU             0C0H
* Data 1
 80 :           DATA1:  EQU             80H
* Data 2
 90 :           DATA2:  EQU             90H
* Result
 A0 :           ANS:    EQU             0A0H

 00 :   6C C0           LD      IX,     [N]
 02 :   20              RCF
 03 :   66 7F   LOOP:   LD      ACC,    [IX+DATA1-1]
 05 :   96 8F           ADC     ACC,    [IX+DATA2-1]
 07 :   76 9F           ST      ACC,    [IX+ANS-1]
 09 :   AA 01           SUB     IX,     1
 0B :   33 03           BP              LOOP
 0D :   0F              HLT

                        END
```

4.3 ユークリッドの互除法による最大公約数

説明

ユークリッドの互除法で、A(80H 番地), B(81H 番地) の最大公約数を求め、GCD(82 番地) に格納する。なお、A, B とも、127 以下の正数とする。

入出力例

入力

```
80 : 60 (A)
81 : 40 (B)
```

出力

```
82 : 20 (最大公約数)
```

コード

```
*** KUE-CHIP2 Assembler ver.1.0   by H.Ochi ***

* Calculate Greatest Common Divisor using Euclidean Algorithm
* Programmed by Akira Uejima, May. 3, 1992
* e-mail: uejima@mics.cs.ritsumei.ac.jp

* A (0 =< A < 128)
 80 :           A:      EQU             80H
* B (0 < B < 128)
 81 :           B:      EQU             81H
* GCD(result)
 82 :           GCD:    EQU             82H

 00 :   64 80           LD      ACC,    [A]
 02 :   6C 81           LD      IX,     [B]
 04 :   A1      LOOP:   SUB     ACC,    IX
 05 :   32 04           BZP             LOOP
 07 :   B1              ADD     ACC,    IX
 08 :   C8              EOR     IX,     ACC
 09 :   C1              EOR     ACC,    IX
 0A :   C8              EOR     IX,     ACC
 0B :   31 04           BNZ             LOOP
 0D :   74 82           ST      ACC,    [GCD]
 0F :   0F              HLT

                        END
```

4.4 バブルソートによる整列

説明

バブルソートによって、データ領域のDATA(00H番地)から始まる、N(80H番地)バイトのデータを昇順に整列する。ここで、データは、2個から256個の範囲とする。ただし、データが256個の場合は、Nを0とする。なお、ソートの完了の判定に、キャリーフラグを利用している。

入出力例

入力

```
80 : 08 (ソートするデータの長さ)
100 : 10 FF 40 80 C0 D8 7F CD (ソートするデータ)
```

出力

```
100 : 80 C0 CD D8 FF 10 40 7F (ソート結果)
```

コード

```
*** KUE-CHIP2 Assembler ver.1.0   by H.Ochi ***

* Bubble Sorting
* Programmed by Akira Uejima, May. 3, 1992
* e-mail: uejima@mics.cs.ritsumei.ac.jp

* Data(signed) on data page
 00 :           DATA:   EQU             00H
* Data Length(byte) on program page
 80 :           N:      EQU             80H
* Work Area(loop counter) on program page
 90 :           WORK1:  EQU             90H
* Work Area(swap area) on program page
 91 :           WORK2:  EQU             91H

 00 :   6C 80           LD      IX,     [N]
 02 :   AA 01           SUB     IX,     1
 04 :   7C 90           ST      IX,     [WORK1]
 06 :   C9      LP1:    EOR     IX,     IX
 07 :   20              RCF
 08 :   67 00   LP2:    LD      ACC,    (IX+DATA)
 0A :   F7 01           CMP     ACC,    (IX+DATA+1)
 0C :   3F 19           BLE             SKIP
 0E :   74 91           ST      ACC,    [WORK2]
 10 :   67 01           LD      ACC,    (IX+DATA+1)
 12 :   77 00           ST      ACC,    (IX+DATA)
 14 :   64 91           LD      ACC,    [WORK2]
 16 :   77 01           ST      ACC,    (IX+DATA+1)
 18 :   2F              SCF
 19 :   BA 01   SKIP:   ADD     IX,     1
 1B :   FC 90           CMP     IX,     [WORK1]
 1D :   31 08           BNZ             LP2
 1F :   35 29           BNC             FIN
 21 :   6C 90           LD      IX,     [WORK1]
 23 :   AA 01           SUB     IX,     1
 25 :   7C 90           ST      IX,     [WORK1]
 27 :   31 06           BNZ             LP1
 29 :   0F      FIN:    HLT

                        END
```

4.5 CRCの計算

説明

シフト／ローテート命令や、論理演算命令を使用して、データ領域のDATA(00H番地)から始まるN(80H番地)バイトのデータのCRC(Cyclic Redundancy Check)コードを計算する。ただし、データの最大長は256バイトである。データが256バイトの場合は、バイト数Nを0とする。なお、生成多項式として、CCITT X.25規格の$x^{16}+x^{12}+x^5+1$を用いた。

入出力例

入力

```
80 : 08 (データのバイト数)
100 : 62 FF 75 C0 75 C1 C9 7D (データ)
```

出力

```
C0 : B0 8A (CRC)
```

コード

```
*** KUE-CHIP2 Assembler ver.1.0   by H.Ochi ***

* Calculate CRC(Cyclic Redundancy Check) Code
* Programmed by Akira Uejima, May. 3, 1992
* e-mail: uejima@mics.cs.ritsumei.ac.jp

* Data on data page
 00 :           DATA:   EQU             00H
* Data Length(byte) on program page
 80 :           N:      EQU             80H
* Resultant CRC on program page
 C0 :           C1:     EQU             0C0H
 C1 :           C2:     EQU             0C1H
* Work Area on program page
 F0 :           WORK:   EQU             0F0H

 00 :   62 FF           LD      ACC,    0FFH
 02 :   74 C0           ST      ACC,    [C1]
 04 :   74 C1           ST      ACC,    [C2]
 06 :   C9              EOR     IX,     IX
 07 :   7C F0           ST      IX,     [WORK]
 09 :   64 C0   LP1:    LD      ACC,    [C1]
 0B :   C7 00           EOR     ACC,    (IX+DATA)
 0D :   74 C0           ST      ACC,    [C1]
 0F :   6A 08           LD      IX,     8
 11 :   64 C1   LP2:    LD      ACC,    [C2]
 13 :   43              SLL     ACC
 14 :   74 C1           ST      ACC,    [C2]
 16 :   64 C0           LD      ACC,    [C1]
 18 :   45              RLA     ACC
 19 :   74 C0           ST      ACC,    [C1]
 1B :   35 27           BNC             SKIP
 1D :   C4 42           EOR     ACC,    [P1]
 1F :   74 C0           ST      ACC,    [C1]
 21 :   64 C1           LD      ACC,    [C2]
 23 :   C4 43           EOR     ACC,    [P2]
 25 :   74 C1           ST      ACC,    [C2]
 27 :   AA 01   SKIP:   SUB     IX,     1
 29 :   33 11           BP              LP2
 2B :   6C F0           LD      IX,     [WORK]
 2D :   BA 01           ADD     IX,     1
 2F :   FC 80           CMP     IX,     [N]
 31 :   7C F0           ST      IX,     [WORK]
 33 :   31 09           BNZ             LP1
 35 :   64 C1           LD      ACC,    [C2]
 37 :   C2 FF           EOR     ACC,    0FFH
 39 :   74 C1           ST      ACC,    [C2]
 3B :   64 C0           LD      ACC,    [C1]
 3D :   C2 FF           EOR     ACC,    0FFH
 3F :   74 C0           ST      ACC,    [C1]
 41 :   0F              HLT

* CRC Generator Polynomial on program page
* CCITT(x^16 + x^12 + x^5 + 1) -> 1 0001 0000 0010 0001
*                                  1    0    2    1
 42 :   10      P1:     PROG            10H
 43 :   21      P2:     PROG            21H

                        END
```

4.6 符号無し1バイトの乗算

説明

DATA1(80H番地)とDATA2(81H番地)の積を求め、ANS(82H番地)からの2バイトに格納する。なお、乗数、被乗数、積は、全て符号無しの数とする。

入出力例

入力

```
80 : 55 (乗数)
81 : AA (被乗数)
```

出力

```
82 : 38 72 (結果)
```

コード

```
*** KUE-CHIP2 Assembler ver.1.0   by H.Ochi ***

* Unsigned Multiply (1byte*1byte=2byte)
* Programmed by A.Uejima, May. 6, 1992
* e-mail: uejima@mics.cs.ritsumei.ac.jp

* Multiplier
 80 :           DATA1:  EQU             80H
* Multiplicand
 81 :           DATA2:  EQU             81H
* Product
 82 :           ANS:    EQU             82H
* Work Area
 F0 :           WORK:   EQU             0F0H

 00 :   C0              EOR     ACC,    ACC
 01 :   74 82           ST      ACC,    [ANS]
 03 :   74 83           ST      ACC,    [ANS+1]
 05 :   74 F0           ST      ACC,    [WORK]
 07 :   64 81           LD      ACC,    [DATA2]
 09 :   74 F1           ST      ACC,    [WORK+1]
 0B :   6C 80           LD      IX,     [DATA1]
 0D :   4A      LOOP:   SRL     IX
 0E :   35 1D           BNC             SKIP
 10 :   20              RCF
 11 :   64 83           LD      ACC,    [ANS+1]
 13 :   94 F1           ADC     ACC,    [WORK+1]
 15 :   74 83           ST      ACC,    [ANS+1]
 17 :   64 82           LD      ACC,    [ANS]
 19 :   94 F0           ADC     ACC,    [WORK]
 1B :   74 82           ST      ACC,    [ANS]
 1D :   FA 00   SKIP:   CMP     IX,     0
 1F :   39 2D           BZ              FIN
 21 :   64 F1           LD      ACC,    [WORK+1]
 23 :   41              SLA     ACC
 24 :   74 F1           ST      ACC,    [WORK+1]
 26 :   64 F0           LD      ACC,    [WORK]
 28 :   45              RLA     ACC
 29 :   74 F0           ST      ACC,    [WORK]
 2B :   30 0D           BA              LOOP
 2D :   0F      FIN:    HLT

                        END
```

4.7 符号無し4バイトの乗算

説明

DATA1(80H番地)からの4バイトと、DATA2(84H番地)からの4バイトを小さい番地側をMSBとして積を求め、ANS(88H番地)からの8バイトに格納する。なお、乗数、被乗数、積は、全て符号無しの数とする。

入出力例

入力

```
80 : 88 88 88 88 (乗数)
84 : 22 22 22 22 (被乗数)
```

出力

```
88 : 12 34 56 78 76 54 32 10 (結果)
```

コード

```
*** KUE-CHIP2 Assembler ver.1.0   by H.Ochi ***

* Unsigned Multiply (4byte*4byte=8byte)
* Programmed by Koichi YASUOKA, Apr. 17, 1992
* Ask more to Hiroyuki OCHI
*  Dept. of Info. Sci., Kyoto Univ., Kyoto 606-01, Japan
*  e-mail:ochi@kuis.kyoto-u.ac.jp

* Storage for Multiplier (Given)
 80 :           DATA1:  EQU             80H
* Storage for Multiplicand (Given)
 84 :           DATA2:  EQU             84H
* Storage for Product (Result)
 88 :           RESULT: EQU             88H
* Storage for Partial Product (Work Area)
 E0 :           WORK:   EQU             0E0H
* Storage for Loop Counter (Work Area)
 F0 :           COUNT:  EQU             0F0H

 00 :   62 20           LD      ACC,    32
 02 :   74 F0           ST      ACC,    [COUNT]
 04 :   6A 04           LD      IX,     4
 06 :   C0      LP1:    EOR     ACC,    ACC
 07 :   76 DF           ST      ACC,    [IX+WORK-1]
 09 :   76 87           ST      ACC,    [IX+RESULT-1]
 0B :   76 8B           ST      ACC,    [IX+RESULT+4-1]
 0D :   66 83           LD      ACC,    [IX+DATA2-1]
 0F :   76 E3           ST      ACC,    [IX+WORK+4-1]
 11 :   AA 01           SUB     IX,     1
 13 :   31 06           BNZ             LP1
 15 :   64 83   LP2:    LD      ACC,    [DATA1+3]
 17 :   42              SRL     ACC
 18 :   6A FC           LD      IX,     -4
 1A :   66 84   LP3:    LD      ACC,    [IX+DATA1+4]
 1C :   44              RRA     ACC
 1D :   76 84           ST      ACC,    [IX+DATA1+4]
 1F :   BA 01           ADD     IX,     1
 21 :   31 1A           BNZ             LP3
 23 :   35 32           BNC             LP5
 25 :   6A 08           LD      IX,     8
 27 :   20              RCF
 28 :   66 87   LP4:    LD      ACC,    [IX+RESULT-1]
 2A :   96 DF           ADC     ACC,    [IX+WORK-1]
 2C :   76 87           ST      ACC,    [IX+RESULT-1]
 2E :   AA 01           SUB     IX,     1
 30 :   31 28           BNZ             LP4
 32 :   6A 08   LP5:    LD      IX,     8
 34 :   20              RCF
 35 :   66 DF   LP6:    LD      ACC,    [IX+WORK-1]
 37 :   45              RLA     ACC
 38 :   76 DF           ST      ACC,    [IX+WORK-1]
 3A :   AA 01           SUB     IX,     1
 3C :   31 35           BNZ             LP6
 3E :   64 F0           LD      ACC,    [COUNT]
 40 :   A2 01           SUB     ACC,    1
 42 :   74 F0           ST      ACC,    [COUNT]
 44 :   31 15           BNZ             LP2
 46 :   0F              HLT
                        END
```

4.8 符号無し1バイトの除算

説明

DVD(80H 番地) を、 DVS(81H 番地) で割り、商を QOT(82H 番地) に、余りを RMD(83H 番地) に格納する。なお、被除数、除数、商、余りは、全て符号無しの1バイトの数とする。

入出力例

入力

```
80 : F0 (被除数)
81 : 3F (除数)
```

出力

```
82 : 03 (商)
83 : 33 (余り)
```

コード

```
*** KUE-CHIP2 Assembler ver.1.0   by H.Ochi ***

* Unsigned Division (1byte/1byte=1byte...1byte)
* Programmed by A.Uejima, May. 8, 1992
* e-mail: uejima@mics.cs.ritsumei.ac.jp

* Dividend
 80 :           DVD:    EQU             80H
* Divisor (!= 0)
 81 :           DVS:    EQU             81H
* Quotient
 82 :           QOT:    EQU             82H
* Remainder
 83 :           RMD:    EQU             83H
* Work Area
 F0 :           WORK:   EQU             0F0H

 00 :   C0              EOR     ACC,    ACC
 01 :   74 82           ST      ACC,    [QOT]
 03 :   74 83           ST      ACC,    [RMD]
 05 :   64 80           LD      ACC,    [DVD]
 07 :   74 F0           ST      ACC,    [WORK]
 09 :   6A 08           LD      IX,     8
 0B :   64 F0   LOOP:   LD      ACC,    [WORK]
 0D :   41              SLA     ACC
 0E :   74 F0           ST      ACC,    [WORK]
 10 :   64 83           LD      ACC,    [RMD]
 12 :   45              RLA     ACC
 13 :   20              RCF
 14 :   84 81           SBC     ACC,    [DVS]
 16 :   3D 1B           BC              SP1
 18 :   2F              SCF
 19 :   30 1E           BA              SP2
 1B :   B4 81   SP1:    ADD     ACC,    [DVS]
 1D :   20              RCF
 1E :   74 83   SP2:    ST      ACC,    [RMD]
 20 :   64 82           LD      ACC,    [QOT]
 22 :   45              RLA     ACC
 23 :   74 82           ST      ACC,    [QOT]
 25 :   AA 01           SUB     IX,     1
 27 :   33 0B           BP              LOOP
 29 :   0F              HLT

                        END
```

4.9 マーチングによるメモリテスト

説明

KUE-CHIP2の内部メモリ、または、ボード上の外部メモリが正常かどうかをテストする。

出力

エラーが検出されない間、OBUFのLEDが点滅を続ける。クロック周波数が1MHzのとき、毎秒2回程度点滅する。

コード

```
*** KUE-CHIP2 Assembler ver.1.0   by H.Ochi ***

* Marching Memory Test
* Programmed by Hiroyuki OCHI, Apr. 20, 1992
*  Dept. of Info. Sci., Kyoto Univ., Kyoto 606-01, Japan
*  e-mail:ochi@kuis.kyoto-u.ac.jp

* Run with PC=prog.
* Tests program page [test] through [prog-1] and data page (0) through (0ffh).
* Blinks OBUF LEDs while no error is detected.
* Halts when an error is detected.

* Original Address of Program
 00 :           PROG:   EQU             0

 00 :           CA:     EQU             PROG

 00 :   C0              EOR     ACC,    ACC
 01 :   6A 47           LD      IX,     TEST
 03 :   76 00   LP0:    ST      ACC,    [IX]
 05 :   BA 01           ADD     IX,     1
 07 :   FA 00           CMP     IX,     PROG
 09 :   31 03           BNZ             LP0
 0B :   C9              EOR     IX,     IX
 0C :   77 00   LP1:    ST      ACC,    (IX)
 0E :   BA 01           ADD     IX,     1
 10 :   31 0C           BNZ             LP1
 12 :   6A 47   LP2:    LD      IX,     TEST
 14 :   F6 00   LP3:    CMP     ACC,    [IX]
 16 :   31 46           BNZ             ERR
 18 :   C2 80           EOR     ACC,    80H
 1A :   47              RLL     ACC
 1B :   76 00           ST      ACC,    [IX]
 1D :   39 23           BZ              NX3
 1F :   F2 FF           CMP     ACC,    0FFH
 21 :   31 14           BNZ             LP3
 23 :   C2 FF   NX3:    EOR     ACC,    0FFH
 25 :   BA 01           ADD     IX,     1
 27 :   FA 00           CMP     IX,     PROG
 29 :   31 14           BNZ             LP3
 2B :   C9              EOR     IX,     IX
 2C :   F7 00   LP4:    CMP     ACC,    (IX)
 2E :   31 46           BNZ             ERR
 30 :   C2 80           EOR     ACC,    80H
 32 :   47              RLL     ACC
 33 :   77 00           ST      ACC,    (IX)
 35 :   39 3B           BZ              NX4
 37 :   F2 FF           CMP     ACC,    0FFH
 39 :   31 2C           BNZ             LP4
 3B :   C2 FF   NX4:    EOR     ACC,    0FFH
 3D :   BA 01           ADD     IX,     1
 3F :   31 2C           BNZ             LP4
 41 :   C2 FF           EOR     ACC,    0FFH
 43 :   10              OUT
 44 :   30 12           BA              LP2
 46 :   0F      ERR:    HLT

 47 :           TEST:   EQU             CA
                        END
```

第 5 章

外部インタフェース回路のサンプル

この章では、KUE-CHIP2 教育用ボードの外部インタフェースに新たなハードウェアを接続する方法、および、そのハードウェアを操作する KUE-CHIP2 のサンプルプログラムを紹介する。

5.1 外付けキーボードを利用したメモリのエディタ

説明

次ページの回路図で示されるキーボードをJP2に接続し、これを利用してデータ領域のメモリをエディットする。20個のキースイッチのうち、 0 から F のキーはデータ入力に使用し、 10 11 12 13 のキーは、それぞれアドレスに−2、−1、+1、+2を加える機能に使用する。

入出力例

入力

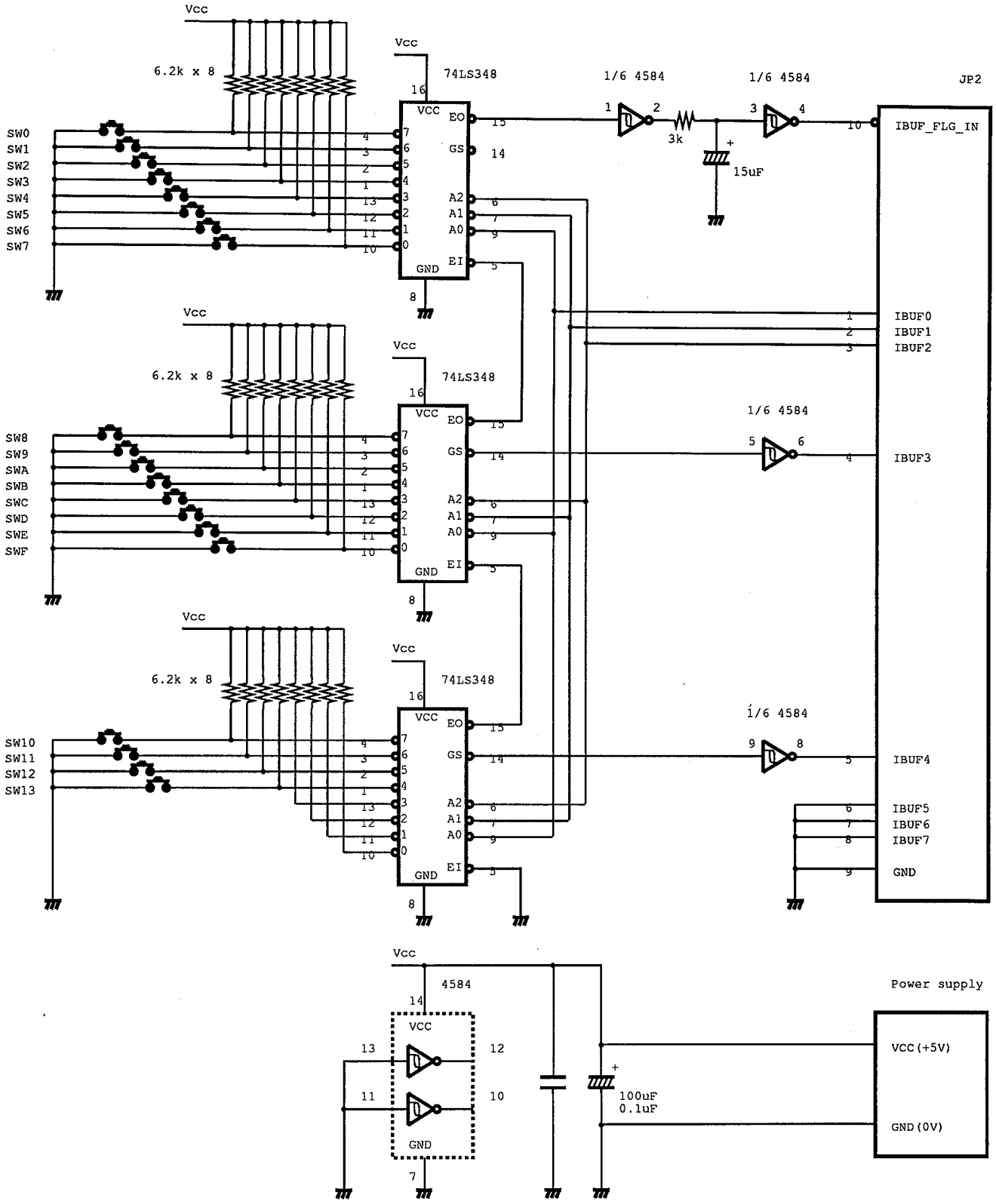
次の手順でキーボードのキースイッチを押す。

1 2 (100H番地に12Hを書き込む)
12 (1番地進める)
3 4 (101H番地に34Hを書き込む)
12 (1番地進める)
5 6 (102H番地に56Hを書き込む)
12 (1番地進める)
7 8 (103H番地に78Hを書き込む)
10 (2番地戻す)
9 A (101H番地に9AHを書き込む)

出力

```
100 : 12 9A 56 78 (上のキー操作で書き換えられたデータ領域)
```

JP2 に接続するキーボードの回路図



- 74LS348(プライオリティ・エンコーダ) の出力 A0 ～ A2 は負論理であるが、このように入力 0 ～ 7 の結線を逆にしてやることにより、論理反転のためのインバータを節約できる。
- ここでは IBUF7 などは使用せずに GND に結線しているが、例えば IBUF7 にシフトキー (他のキーとは独立で、押すと IBUF7=1 になるように結線されたキースイッチ) を追加すると、さらに応用が広がると思われる。

コード

```
*** KUE-CHIP2 Assembler ver.1.0   by H.Ochi ***

* Memory Editor with Keyboard
* Programmed by Hiroyuki OCHI, May. 30, 1992
*  Dept. of Info. Sci., Kyoto Univ., Kyoto 606-01, Japan
*  e-mail:ochi@kuis.kyoto-u.ac.jp

* Connect an encoded and chattering-free keyboard to JP2. The circuit
* diagram of the keyboard can be found in the KUE-CHIP2 Board Reference Manual.
* Before starting the program, set SEL switches to 0101 in order to observe IX.
* OBUF LED's indicate the contents of (IX), where (IX) is a memory byte of
* data page pointed to by IX.
* Effect of the keys are as follows:
*   13H       : IX is incremented by 2.
*   12H       : IX is incremented by 1.
*   11H       : IX is decremented by 1.
*   10H       : IX is decremented by 2.
*   00H - 0FH : Bit 3 - 0 of (IX) are shifted to bit 7 - 4 of (IX), then
*               input value is substituted into bit 3 - 0 of (IX).

 00 :   67 00   LP0:    LD      ACC,    (IX)
 02 :   10      LP1:    OUT
 03 :   34 03   LP2:    BNI             LP2
 05 :   1F              IN
 06 :   F2 10           CMP     ACC,    10H
 08 :   3A 10           BN              LP3
 0A :   20              RCF
 0B :   92 EE           ADC     ACC,    0EEH
 0D :   98              ADC     IX,     ACC
 0E :   30 00           BA              LP0
 10 :   74 1E   LP3:    ST      ACC,    [WORK]
 12 :   67 00           LD      ACC,    (IX)
 14 :   43              SLL     ACC
 15 :   43              SLL     ACC
 16 :   43              SLL     ACC
 17 :   43              SLL     ACC
 18 :   D4 1E           OR      ACC,    [WORK]
 1A :   77 00           ST      ACC,    (IX)
 1C :   30 02           BA              LP1

 1E :           WORK:   EQU             CA

                        END
```

5.2 プリンタを用いたメモリの 16 進ダンプ

説明

次ページの回路図に従ってセントロニクス規格インターフェース準拠のプリンタを接続し、これを利用してプログラム領域及びデータ領域の全メモリの内容を 16 進数でダンプする。

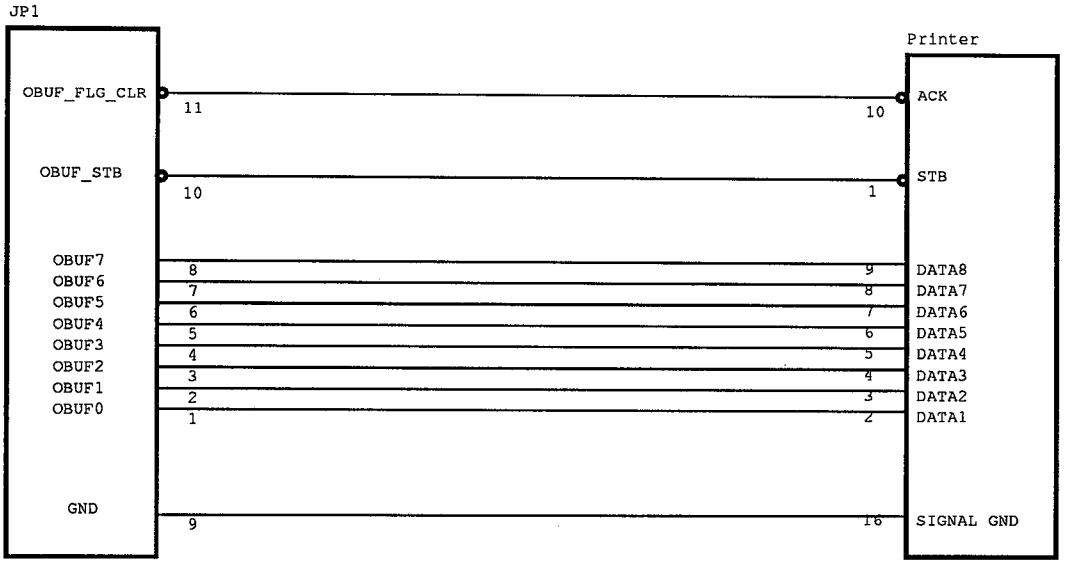
入出力例

出力

次のような印字が得られる。前半 16 行はプログラム領域の内容を、後半 16 行はデータ領域の内容を表す。プログラム領域の 71H 番地まではこのプログラム自身が格納されていることを示している。プログラム領域の 72H 番地以降、及びデータ領域の内容はこの印字例の限りではない。

```
00:C0 74 50 6A 00 7C 52 C0 74 51 64 52 46 46 46 46
10:74 52 E2 0F B2 30 F2 3A 3A 1C B2 07 10 3C 1D 64
20:51 C2 80 74 51 3A 0A E2 01 31 38 62 3A 10 3C 2E
30:64 51 D2 01 74 51 30 44 62 20 10 3C 3B BA 01 61
40:E2 0F 39 56 64 50 F2 00 31 50 66 00 74 52 30 0A
50:67 00 74 52 30 0A 62 0D 10 3C 59 62 0A 10 3C 5E
60:FA 00 31 05 64 50 C2 80 74 50 31 03 62 0C 10 3C
70:6F 0F 55 55 55 55 55 55 55 55 55 55 55 55 55 55
80:00 01 01 55 55 55 55 55 55 55 55 55 55 55 55 55
90:55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
A0:55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
B0:55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
C0:55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
D0:55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
E0:55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
F0:55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
00:55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
10:55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
20:55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
30:55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
40:55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
50:55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
60:55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
70:55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
80:55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
90:55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
A0:55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
B0:55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
C0:55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
D0:55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
E0:55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
F0:55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
```

JP1 へのプリンタの接続方法



- ここではセントロニクス規格準拠のプリンタを想定している。また、機種によってはこれ以外にも結線しなければならない信号線がある場合があるので、プリンタの取り扱い説明書で確認されたい。
- 実装にあたっては、JP1 とプリンタとの結線にツイステッドペア線を使用し、リターン線を両端で接地するべきである。これを行なわない場合はノイズによる誤動作の可能性があるので、なるべく 30cm 以内のケーブルで結線すること。

コード

```
*** KUE-CHIP2 Assembler ver.1.1   by H.Ochi ***

* Memory Dump with Printer
* Programmed by Hiroyuki OCHI, June 26, 1992
*  Dept. of Info. Sci., Kyoto Univ., Kyoto 606-01, Japan
*  e-mail:ochi@kuis.kyoto-u.ac.jp

* Connect a printer with centronics standard interface. The circuit
* diagram for connection can be found in the KUE-CHIP2 Board Reference
* Manual. See also your printer's reference manual.

 0D :           CR:     EQU             0DH
 0A :           LF:     EQU             0AH
 0C :           FF:     EQU             0CH
 20 :           SP:     EQU             20H
 3A :           CL:     EQU             3AH

 80 :           FLAG0:  EQU             80H
 81 :           FLAG1:  EQU             81H
 82 :           DATA:   EQU             82H

 00 :   C0              EOR     ACC,    ACC
 01 :   74 80           ST      ACC,    [FLAG0]
 03 :   6A 00   LP0:    LD      IX,     0
 05 :   7C 82   LP1:    ST      IX,     [DATA]
 07 :   C0              EOR     ACC,    ACC
 08 :   74 81           ST      ACC,    [FLAG1]
 0A :   64 82   LP2:    LD      ACC,    [DATA]
 0C :   46              RRL     ACC
 0D :   46              RRL     ACC
 0E :   46              RRL     ACC
 0F :   46              RRL     ACC
 10 :   74 82           ST      ACC,    [DATA]
 12 :   E2 0F           AND     ACC,    0FH
 14 :   B2 30           ADD     ACC,    30H
 16 :   F2 3A           CMP     ACC,    3AH
 18 :   3A 1C           BN              NX3
 1A :   B2 07           ADD     ACC,    41H-3AH
 1C :   10      NX3:    OUT
 1D :   3C 1D   LP4:    BNO             LP4
 1F :   64 81           LD      ACC,    [FLAG1]
 21 :   C2 80           EOR     ACC,    80H
 23 :   74 81           ST      ACC,    [FLAG1]
 25 :   3A 0A           BN              LP2
 27 :   E2 01           AND     ACC,    1
 29 :   31 38           BNZ             NX6
 2B :   62 3A           LD      ACC,    CL
 2D :   10              OUT
 2E :   3C 2E   LP5:    BNO             LP5
 30 :   64 81           LD      ACC,    [FLAG1]
 32 :   D2 01           OR      ACC,    1
 34 :   74 81           ST      ACC,    [FLAG1]
 36 :   30 44           BA              NX8
 38 :   62 20   NX6:    LD      ACC,    SP
 3A :   10              OUT
 3B :   3C 3B   LP7:    BNO             LP7
 3D :   BA 01           ADD     IX,     1
 3F :   61              LD      ACC,    IX
 40 :   E2 0F           AND     ACC,    0FH
 42 :   39 56           BZ              NX10
 44 :   64 80   NX8:    LD      ACC,    [FLAG0]
 46 :   F2 00           CMP     ACC,    0
 48 :   31 50           BNZ             NX9
```

```
4A :   66 00          LD      ACC,    [IX]
4C :   74 82          ST      ACC,    [DATA]
4E :   30 0A          BA              LP2
50 :   67 00   NX9:   LD      ACC,    (IX)
52 :   74 82          ST      ACC,    [DATA]
54 :   30 0A          BA              LP2
56 :   62 0D   NX10:  LD      ACC,    CR
58 :   10             OUT
59 :   3C 59   LP11:  BNO             LP11
5B :   62 0A          LD      ACC,    LF
5D :   10             OUT
5E :   3C 5E   LP12:  BNO             LP12
60 :   FA 00          CMP     IX,     0
62 :   31 05          BNZ             LP1
64 :   64 80          LD      ACC,    [FLAGO]
66 :   C2 80          EOR     ACC,    80H
68 :   74 80          ST      ACC,    [FLAGO]
6A :   31 03          BNZ             LP0
6C :   62 0C          LD      ACC,    FF
6E :   10             OUT
6F :   3C 6F   LP13:  BNO             LP13
71 :   0F             HLT

                      END
```

5.3 シンクロスコープを用いた文字の表示

説明

次ページの回路図で示されるD-Aコンバータを JP1 に接続し、その出力をシンクロスコープに接続する。これにより、シンクロスコープに $0(0V) \sim 127(約3V)$ の電圧と、トリガー信号を入力することができる。

そして、ソフトウェアがデータ領域に用意されたフォントデータに基づいて $v(V)$ と $0(V)$ を適宜出力することで、$v(V)$ の位置に破線を表示する。この v を徐々に変化させ、トリガ入力により同期をとり、テレビモニタのラスタースキャン方式のごとく文字を表示する。

入出力例

入力

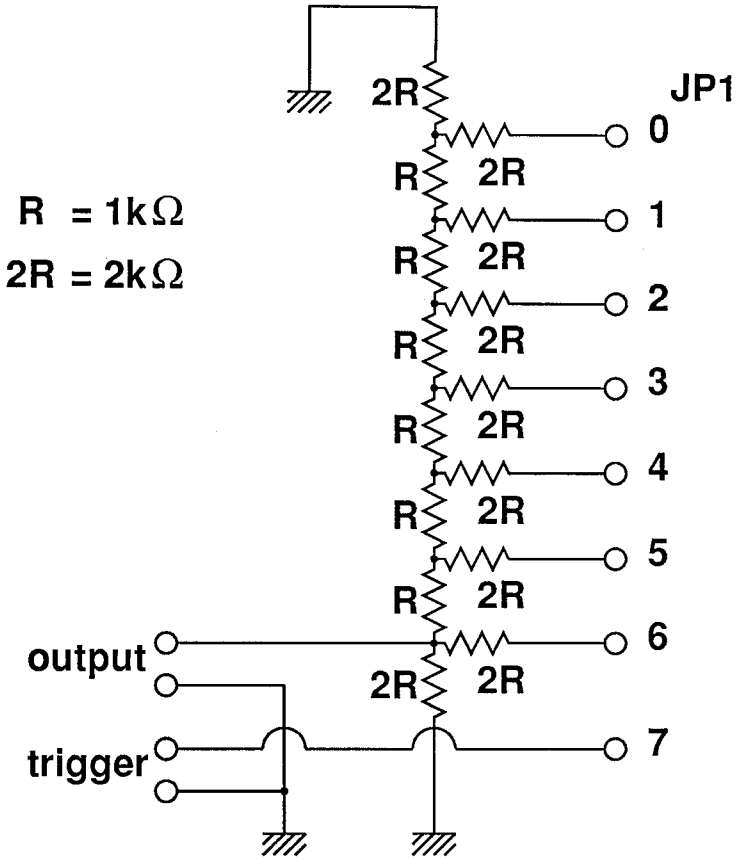
以下に示す16×15ドットのフォントデータをデータ領域に与える。

```
100 : 0F 80 10 60 20 10 40 08 40 08 80 04 80 04 80 04
110 : 80 04 8F 88 90 48 50 31 50 12 30 EC 0F 00
```

出力

シンクロスコープに文字Qが表示される。この場合、KUE-CHIP2のクロックを1MHzとし、シンクロスコープの掃引周期を0.2mSec付近とした時、同期する。

JP1 に接続する D-A コンバータの回路図



- OBUF0~OBUF6 の値 0~127 をアナログ値 0(V) ~ (約)3(V) に変換し、シンクロスコープの入力端子に繋ぐ。
- OBUF7 の出力はシンクロスコープのトリガ端子に繋ぐ。
- 抵抗はできるだけ高精度のものを用いる。

コード

```
*** KUE-CHIP2 Assembler ver.1.0   by H.Ochi ***

* Syncro Scope Display
* Programmed by OKADA Kazuhisa, Jun. 30, 1992
*  Dept. of Electr. Eng., Kyoto Univ., Japan
*  e-mail:okada@tamaru.kuee.kyoto-u.ac.jp

              LEVEL0: EQU     00H * normally zero
              MAXLVL: EQU     40H
              ADDLVL: EQU     04H * MAXLVL/ADDLVL-1 = Y dots
              XBYTE:  EQU     02H * XBYTE x 8 = X dots
              WTIME:  EQU     80H * waiting sweep time

80 :          COUNT0: EQU     80H
81 :          COUNT1: EQU     81H
82 :          COUNT2: EQU     82H
91 :          DATA:   EQU     91H

00 :    C9    LOOP3:  EOR     IX,     IX
01 :    62 40         LD      ACC,    MAXLVL
03 :    74 82 LOOP2:  ST      ACC,    [COUNT2]
05 :    62 80         LD      ACC,    80H
07 :    10            OUT
08 :    62 02         LD      ACC,    XBYTE
0A :    74 81 LOOP1:  ST      ACC,    [COUNT1]
0C :    62 80         LD      ACC,    80H
0E :    74 80 LOOP0:  ST      ACC,    [COUNT0]
10 :    E7 00         AND     ACC,    (IX)
12 :    39 18         BZ      LBL1
14 :    64 82         LD      ACC,    [COUNT2]
16 :    30 1B         BA      LBL2
18 :    62 00 LBL1:   LD      ACC,    LEVEL0
1A :    00            NOP
1B :    10    LBL2:   OUT
1C :    64 80         LD      ACC,    [COUNT0]
1E :    42            SRL     ACC
1F :    39 2A         BZ      LBL3
21 :    00            NOP
22 :    00            NOP
23 :    00            NOP
24 :    00            NOP
25 :    00            NOP
26 :    00            NOP
27 :    00            NOP
28 :    30 0E         BA      LOOP0
2A :    BA 01 LBL3:   ADD     IX,     1
2C :    64 81         LD      ACC,    [COUNT1]
2E :    A2 01         SUB     ACC,    1
30 :    31 0A         BNZ     LOOP1
32 :    62 00         LD      ACC,    LEVEL0
34 :    10            OUT
35 :    62 80         LD      ACC,    WTIME
37 :    A2 01 WAIT:   SUB     ACC,    1
39 :    31 37         BNZ     WAIT
3B :    64 82         LD      ACC,    [COUNT2]
3D :    A2 04         SUB     ACC,    ADDLVL
3F :    31 03         BNZ     LOOP2
41 :    30 00         BA      LOOP3
                      END
```

16×15 のフォントの場合、

0000111110000000 (00 番地 0F) (01 番地 80)
0001000001100000 (02 番地 10) (03 番地 60)
0010000000010000 (04 番地 20) (05 番地 10)
⋮ ⋮
0000111100000000 (1C 番地 0F) (1D 番地 00)

のフォントデータをデータ領域のそれぞれの番地に書き込む。

5.4 ボード間通信

説明

次ページに示すように、2台のKUE-CHIP2教育用ボード(A,B)のJPを接続し、ボードAのデータ領域の内容の一部または全部を、ボードBのデータ領域の指定番地へ送る。ただし、データ領域全部(256バイト)を送る場合は、送信データ長を0とする。

入出力例

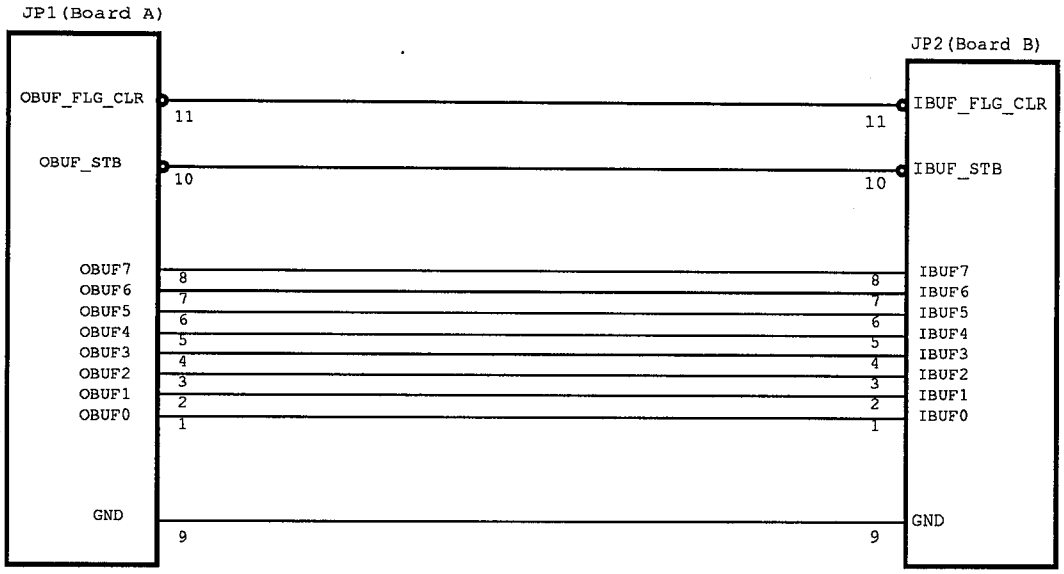
入力

```
A> 80 : 00 (送信データ開始アドレス)
A> 81 : 08 (送信データ長)
A> 82 : 80 (行先アドレス)
A>100 : 00 11 33 77 FF EE CC 88  (送信データ)
```

出力

```
B>180 : 00 11 33 77 FF EE CC 88  (受信データ)
```

2台のKUE-CHIP2教育用ボードの接続方法



コード

送信側

```
*** KUE-CHIP2 Assembler ver.1.0   by H.Ochi ***

* Transmitter for Communications Between two KUE-CHIP2 Boards
* Programmed by Akira Uejima, Jul. 2, 1992
* e-mail: uejima@mics.cs.ritsumei.ac.jp

* Transmit Data Address
 80 :           DT:     EQU             80H
* Transmit Length
 81 :           LN:     EQU             81H
* Destination Address
 82 :           TO:     EQU             82H
* Work Area
 F0 :           WORK:   EQU             0F0H

 00 :   64 80           LD      ACC,    [DT]
 02 :   B4 81           ADD     ACC,    [LN]
 04 :   74 F0           ST      ACC,    [WORK]
 06 :   64 82           LD      ACC,    [TO]
 08 :   10              OUT
 09 :   3C 09   NO1:    BNO             NO1
 0B :   64 81           LD      ACC,    [LN]
 0D :   10              OUT
 0E :   3C 0E   NO2:    BNO             NO2
 10 :   6C 80           LD      IX,     [DT]
 12 :   67 00   LOOP:   LD      ACC,    (IX+0)
 14 :   10              OUT
 15 :   3C 15   NO3:    BNO             NO3
 17 :   BA 01           ADD     IX,     1
 19 :   FC F0           CMP     IX,     [WORK]
 1B :   31 12           BNZ             LOOP
 1D :   0F              HLT

                        END
```

受信側

```
*** KUE-CHIP2 Assembler ver.1.0   by H.Ochi ***

* Receiver for Communications Between two KUE-CHIP2 Boards
* Programmed by Akira Uejima, Jul. 2, 1992
* e-mail: uejima@mics.cs.ritsumei.ac.jp

* Work Area
 F0 :            WORK:   EQU             0F0H

 00 :   34 00    NI1:    BNI             NI1
 02 :   1F               IN
 03 :   68               LD      IX,     ACC
 04 :   34 04    NI2:    BNI             NI2
 06 :   1F               IN
 07 :   B8               ADD     IX,     ACC
 08 :   7C F0            ST      IX,     [WORK]
 0A :   A8               SUB     IX,     ACC
 0B :   34 0B    LOOP:   BNI             LOOP
 0D :   1F               IN
 0E :   77 00            ST      ACC,    (IX+0)
 10 :   BA 01            ADD     IX,     1
 12 :   FC F0            CMP     IX,     [WORK]
 14 :   31 0B            BNZ             LOOP
 16 :   0F               HLT

                         END
```

第 6 章

演習課題

この章では、KUE-CHIP2 ボードを利用した演習課題の例を列挙する。データの与え方、解の出力の方法など詳細な仕様は適当に設定されたい。

- 2 台の KUE-CHIP2 ボードを用意し、片方 (これを送信側と呼ぶ) の JP1 と他方 (これを受信側と呼ぶ) の JP2 を結線してデータを転送するプログラム (送信用プログラム及び受信用プログラム) を作れ (IN、OUT 命令の他、BNI、BNO 命令も利用すること)。また、送信側ボードのクロック周波数が受信側ボードのクロック周波数より高い場合、及びその逆の場合について、それぞれ実際に動作させ確認せよ。
- 算術演算命令を使わずに、ADC 命令をシミュレートする命令列を考えよ。
- 16 進－10 進変換／10 進－16 進変換をするプログラムを作れ (例えば前者は 063H が与えられた時にその 2 進化 10 進表現 099H を求めるものであり、後者はその逆である)。8 ビットを越える大きな数を扱えることが望ましい。
- 与えられた 2 つの行列の積を求めるプログラムを作れ。
- クイックソートを行なうプログラムを作り、所要フェーズ数をバブルソートと比較せよ。
- ACC の初期値が n の時、ちょうど (n^3) フェーズで停止するプログラムを作れ (但し、$10 < n < 128$ などの仮定を設けてよい)。
- データ領域のメモリをテープに見立て、内部状態、テープヘッドの位置をそれぞれ ACC、IX で与えた時、チューリングマシンの 1 ステップの動作をシミュレートするプログラムを作れ (必要なら、与えられたチューリングマシンの状態遷移関数などから KUE-CHIP2 のプログラムを生成するコンパイラを開発せよ)。なお、1 ステップの動作をシミュレートするたびに停止するよう HLT 命令を挿入し、その直後にプログラムの先頭に分岐する BA 命令を付け加えておけば、SS スイッチを押すごとに 1 ステップずつシミュレートを繰り返すことができるので便利である。
- 浮動小数点表現で表された 2 数の加算を行なうプログラムを作れ。また、乗除算も考えよ。
- 第 5章のシンクロスコープに文字を表示する項を参照して、X-4bit、Y-4bit のインタフェース回路を作成し、16×16 ドットのグラフィック表示を試みよ。また、適当な時定数の一次遅れ回路を付加し、一筆書きで図形を描け。
- 第 5章の外付けキーボードを利用して電卓を実現せよ (入出力が 10 進数でできるようにするとか、四則演算以外の機能を付加するなど、工夫せよ)。
- 第 5章の D-A コンバータの出力をアンプに入力することで、さまざまな音を鳴らすことが出来る。三角波、矩形波、サイン波の音を出すプログラムを作成し、それぞれの音を比較せよ。また、外付けキーボードを利用して、エレクトーンを作れ。
- IBUF,OBUF にスタックメモリを繋ぎ、サブルーチンコールを実現せよ。外部にカウンタを用意し、IBUF(12 ピン) と OBUF(12 ピン) により、カウンタをインクリメント、デクリメントさせ、メモリのアドレスとすればよい。

第 7 章

KUE-CHIP2 命令仕様

7.1 アセンブラ文法

KUE-CHIP2 のアドレスモード

ACC	アキュムレータ
IX	インデックスレジスタ
d	即値アドレス
[d]	絶対アドレス (プログラム領域)
(d)	絶対アドレス (データ領域)
[IX+d]	インデックス修飾アドレス (プログラム領域)
(IX+d)	インデックス修飾アドレス (データ領域)

KUE-CHIP2 論理アドレスモード

{ea}	全てのアドレスモード
{reg}	レジスタアドレスモード
{imm}	即値アドレスモード
{ma}	メモリ参照アドレスモード

アドレスモード対応表

	ACC	IX	d	[d]	(d)	[IX+d]	(IX+d)
{ea}	○	○	○	○	○	○	○
{reg}	○	○	×	×	×	×	×
{imm}	×	×	○	×	×	×	×
{ma}	×	×	×	○	○	○	○

KUE-CHIP2 命令表

略記号	アドレスモード	命令機能
HLT		Halt
NOP		No Operation
IN		INput
OUT		OUTput
SCF		Set Carry Flug
RCF		Reset Carry Flug
LD	{reg},{ea}	LoaD
ST	{reg},{ma}	STore
ADD	{reg},{ea}	ADD
ADC	{reg},{ea}	ADd with Carry
SUB	{reg},{ea}	SUBtract
SBC	{reg},{ea}	SuBtract with Carry
CMP	{reg},{ea}	CoMPare
AND	{reg},{ea}	AND
OR	{reg},{ea}	OR
EOR	{reg},{ea}	Exclusive OR
S*sm*	{reg}	Shift
R*sm*	{reg}	Rotate
RA		Right Arithmetically
LA		Left Arithmetically
RL		Right Logically
LL		Left Logically
B*cc*	{imm}	Branch
A		Always
VF		on oVerFlow
NZ		on Not Zero
Z		on Zero
ZP		on Zero or Positive
N		on Negative
P		on Positive
ZN		on Zero or Negative
NI		on No Input
NO		on No Output
NC		on No Carry
C		on Carry
E		on Greater than or Equal
LT		on Less Than
GT		on Greater Than
LE		on Less than or Equal

7.2 命令セット

略記号	命令コード(1語目)								B'(2語目)	命令機能の概略	
NOP	0	0	0	0	0	-	-	-	×	No OPeration	
HLT	0	0	0	0	1	-	-	-	×	HaLT	停止
	0	1	0	1	-	-	-	-	×		未使用 (HLT)
OUT	0	0	0	1	0	-	-	-	×	OUTput	$(ACC) \rightarrow OBUF$
IN	0	0	0	1	1	-	-	-	×	INput	$(IBUF) \rightarrow ACC$
RCF	0	0	1	0	0	-	-	-	×	Reset CF	$0 \rightarrow CF$
SCF	0	0	1	0	1	-	-	-	×	Set CF	$1 \rightarrow CF$
Bcc	0	0	1	1	cc				◎	Branch cc	条件が成立すれば $B' \rightarrow PC$
Ssm	0	1	0	0	A	0	sm		×	Shift sm	$(A) \rightarrow$ shift , rotate $\rightarrow A$
Rsm	0	1	0	0	A	1	sm		×	Rotate sm	はみ出したビット $\rightarrow CF$
LD	0	1	1	0	A		B		○	LoaD	$(B) \rightarrow A$
ST	0	1	1	1	A		B		◎	STore	$(A) \rightarrow B$
SBC	1	0	0	0	A		B		○	SuB with Carry	$(A)-(B)-CF \rightarrow A$
ADC	1	0	0	1	A		B		○	ADd with Carry	$(A)+(B)+CF \rightarrow A$
SUB	1	0	1	0	A		B		○	SUBtract	$(A)-(B) \rightarrow A$
ADD	1	0	1	1	A		B		○	ADD	$(A)+(B) \rightarrow A$
EOR	1	1	0	0	A		B		○	Exclusive OR	$(A) \oplus (B) \rightarrow A$
OR	1	1	0	1	A		B		○	OR	$(A) \vee (B) \rightarrow A$
AND	1	1	1	0	A		B		○	AND	$(A) \wedge (B) \rightarrow A$
CMP	1	1	1	1	A		B		○	CoMPare	$(A)-(B)$

cc : Condition Code

A	0	0	0	0	Always	常に成立
VF	1	0	0	0	on oVerflow	桁あふれ $VF=1$
NZ	0	0	0	1	on Not Zero	$\neq 0$ $ZF=0$
Z	1	0	0	1	on Zero	$=0$ $ZF=1$
ZP	0	0	1	0	on Zero or Positive	≥ 0 $NF=0$
N	1	0	1	0	on Negative	<0 $NF=1$
P	0	0	1	1	on Positive	>0 $(NF \vee ZF)=0$
ZN	1	0	1	1	on Zero or Negative	≤ 0 $(NF \vee ZF)=1$
NI	0	1	0	0	on No Input	$\overline{IBUF\_FLG\_IN}=0$
NO	1	1	0	0	on No Output	$OBUF\_FLG\_IN=1$
NC	0	1	0	1	on Not Carry	$CF=0$
C	1	1	0	1	on Carry	$CF=1$
GE	0	1	1	0	on Greater than or Equal	≥ 0 $(VF \oplus NF)=0$
LT	1	1	1	0	on Less Than	<0 $(VF \oplus NF)=1$
GT	0	1	1	1	on Greater Than	>0 $((VF \oplus NF) \vee ZF)=0$
LE	1	1	1	1	on Less than or Equal	≤ 0 $((VF \oplus NF) \vee ZF)=1$

sm : Shift Mode

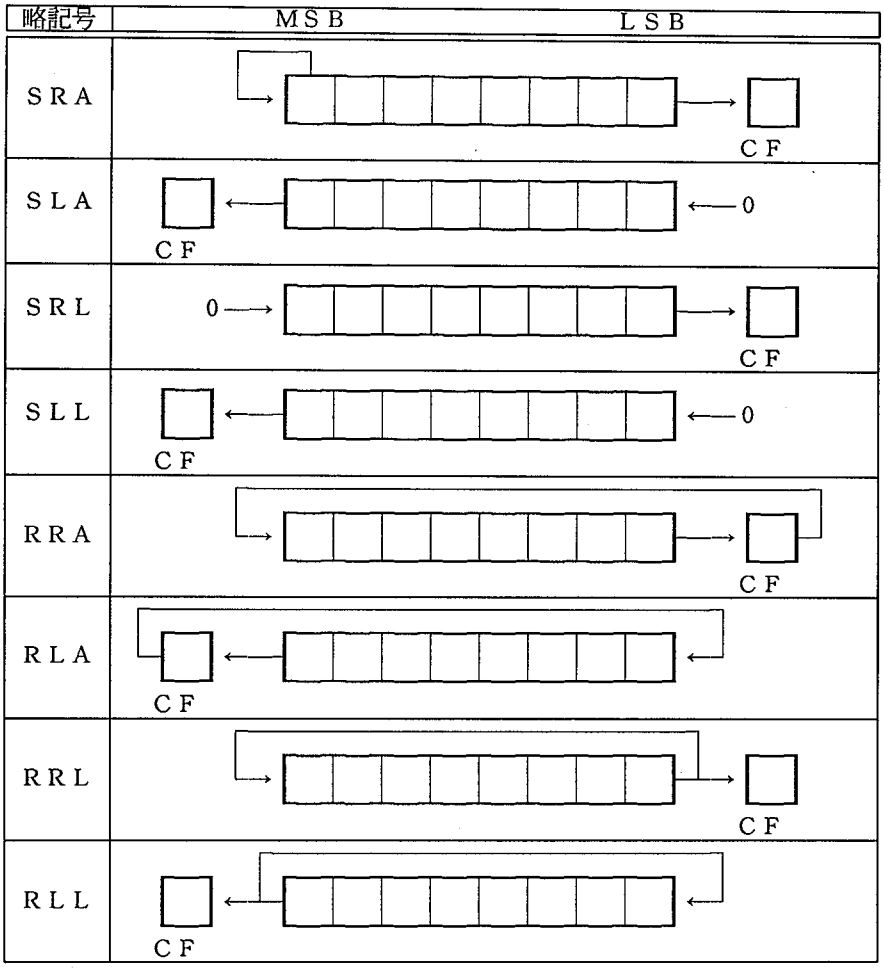
RA	0	0	Right Arithmetically
LA	0	1	Left Arithmetically
RL	1	0	Right Logically
LL	1	1	Left Logically

A = 0 : ACC
A = 1 : IX

B'(2語目)
×：不用
○：不用 or 必要
◎：必要

B = 000 : ACC
B = 001 : IX
B = 01– : d (即値アドレス)
B = 100 : [d] (絶対アドレス)
B = 101 : (d) (絶対アドレス)
B = 110 : [IX+d] (インデックス修飾アドレス)
B = 111 : (IX+d) (インデックス修飾アドレス)

7.3 Shift / Rotate 命令の機能



ＳＬＡとＳＬＬでは，実行後のフラグの状態が異なる (7.4節を参照されたい)

7.4 フラグ機能

CF : Carry Flag　VF : oVerflow Flag　NF : Negative Flag　ZF : Zero Flag

略記号	命令機能の概略	実行への影響 †				実行後の状態 ‡			
		CF	VF	NF	ZF	CF	VF	NF	ZF
NOP	No OPeration	—	—	—	—	—	—	—	—
HLT	HaLT	—	—	—	—	—	—	—	—
OUT	OUTput	—	—	—	—	—	—	—	—
IN	INput	—	—	—	—	—	—	—	—
RCF	Reset Carry Flag	—	—	—	—	0	—	—	—
SCF	Set Carry Flag	—	—	—	—	1	—	—	—
SRA	Shift Right Arithmetically	—	—	—	—	$b0$	0	N	Z
SLA	Shift Left Arithmetically	—	—	—	—	$b7$	V	N	Z
SRL	Shift Right Logically	—	—	—	—	$b0$	0	N	Z
SLL	Shift Left Logically	—	—	—	—	$b7$	0	N	Z
RRA	Rotate Right Arithmetically	$b7$	—	—	—	$b0$	0	N	Z
RLA	Rotate Left Arithmetically	$b0$	—	—	—	$b7$	V	N	Z
RRL	Rotate Right Logically	—	—	—	—	$b0$	0	N	Z
RLL	Rotate Left Logically	—	—	—	—	$b7$	0	N	Z
LD	LoaD	—	—	—	—	—	—	—	—
ST	STore	—	—	—	—	—	—	—	—
SBC	SuBtract with Carry	c	—	—	—	C	V	N	Z
ADC	ADd with Carry	c	—	—	—	C	V	N	Z
SUB	SUBtract	—	—	—	—	—	V	N	Z
ADD	ADD	—	—	—	—	—	V	N	Z
EOR	Exclusive OR	—	—	—	—	—	0	N	Z
OR	OR	—	—	—	—	—	0	N	Z
AND	AND	—	—	—	—	—	0	N	Z
CMP	CoMPare	—	—	—	—	—	V	N	Z
BA	Branch Always	—	—	—	—	—	—	—	—
BVF	Branch on oVerFlow	—	VF	—	—	—	—	—	—
BNZ	Branch on Not Zero	—	—	—	$\overline{ZF}$	—	—	—	—
BZ	Branch on Zero	—	—	—	ZF	—	—	—	—
BZP	Branch on Zero or Positive	—	—	$\overline{NF}$	—	—	—	—	—
BN	Branch on Negative	—	—	NF	—	—	—	—	—
BP	Branch on Positive	—	—	$\overline{NF \vee ZF}$		—	—	—	—
BZN	Branch on Zero or Negative	—	—	$NF \vee ZF$		—	—	—	—
BNI	Branch on No Input	—	—	—	—	—	—	—	—
BNO	Branch on No Output	—	—	—	—	—	—	—	—
BNC	Branch on Not Carry	$\overline{CF}$	—	—	—	—	—	—	—
BC	Branch on Carry	CF	—	—	—	—	—	—	—
BGE	Branch on Greater than or Equal	—	$\overline{VF \oplus NF}$		—	—	—	—	—
BLT	Branch on Less Than	—	$VF \oplus NF$		—	—	—	—	—
BGT	Branch on Greater Than	—	$\overline{(VF \oplus NF) \vee ZF}$			—	—	—	—
BLE	Branch on Less than or Equal	—	$(VF \oplus NF) \vee ZF$			—	—	—	—

†実行への影響
- c : Referred to as carry/borrow input.
- $b7$: Substituted into bit 7 of the operand A.
- $b0$: Substituted into bit 0 of the operand A.
- Exp. : Condition which causes branch.
- — : No effect.

‡実行後の状態
- C : Set to 1 iff carry/borrow occur.
- V : Set to 1 iff overflow occur.
- N : Set to 1 iff bit 7 of the result is 1.
- Z : Set to 1 iff all bits of the result are 0.
- $b7$: Set to 1 iff bit 7 of the operand A was 1.
- $b0$: Set to 1 iff bit 0 of the operand A was 1.
- 0 : Set to 0.
- 1 : Set to 1.
- — : Not modified.

7.5 命令コード早見表

ACC に対する代入および算術命令

	ACC	IX	d	[d]	(d)	[IX+d]	(IX+d)
LD ACC,	60	61	62	64	65	66	67
ST ACC,	–	–	–	74	75	76	77
SBC ACC,	80	81	82	84	85	86	87
ADC ACC,	90	91	92	94	95	96	97
SUB ACC,	A0	A1	A2	A4	A5	A6	A7
ADD ACC,	B0	B1	B2	B4	B5	B6	B7
EOR ACC,	C0	C1	C2	C4	C5	C6	C7
OR ACC,	D0	D1	D2	D4	D5	D6	D7
AND ACC,	E0	E1	E2	E4	E5	E6	E7
CMP ACC,	F0	F1	F2	F4	F5	F6	F7

IX に対する代入および算術命令

	ACC	IX	d	[d]	(d)	[IX+d]	(IX+d)
LD IX,	68	69	6A	6C	6D	6E	6F
ST IX,	–	–	–	7C	7D	7E	7F
SBC IX,	88	89	8A	8C	8D	8E	8F
ADC IX,	98	99	9A	9C	9D	9E	9F
SUB IX,	A8	A9	AA	AC	AD	AE	AF
ADD IX,	B8	B9	BA	BC	BD	BE	BF
EOR IX,	C8	C9	CA	CC	CD	CE	CF
OR IX,	D8	D9	DA	DC	DD	DE	DF
AND IX,	E8	E9	EA	EC	ED	EE	EF
CMP IX,	F8	F9	FA	FC	FD	FE	FF

シフト命令

	ACC	IX
SRA	40	48
SLA	41	49
SRL	42	4A
SLL	43	4B
RRA	44	4C
RLA	45	4D
RRL	46	4E
RLL	47	4F

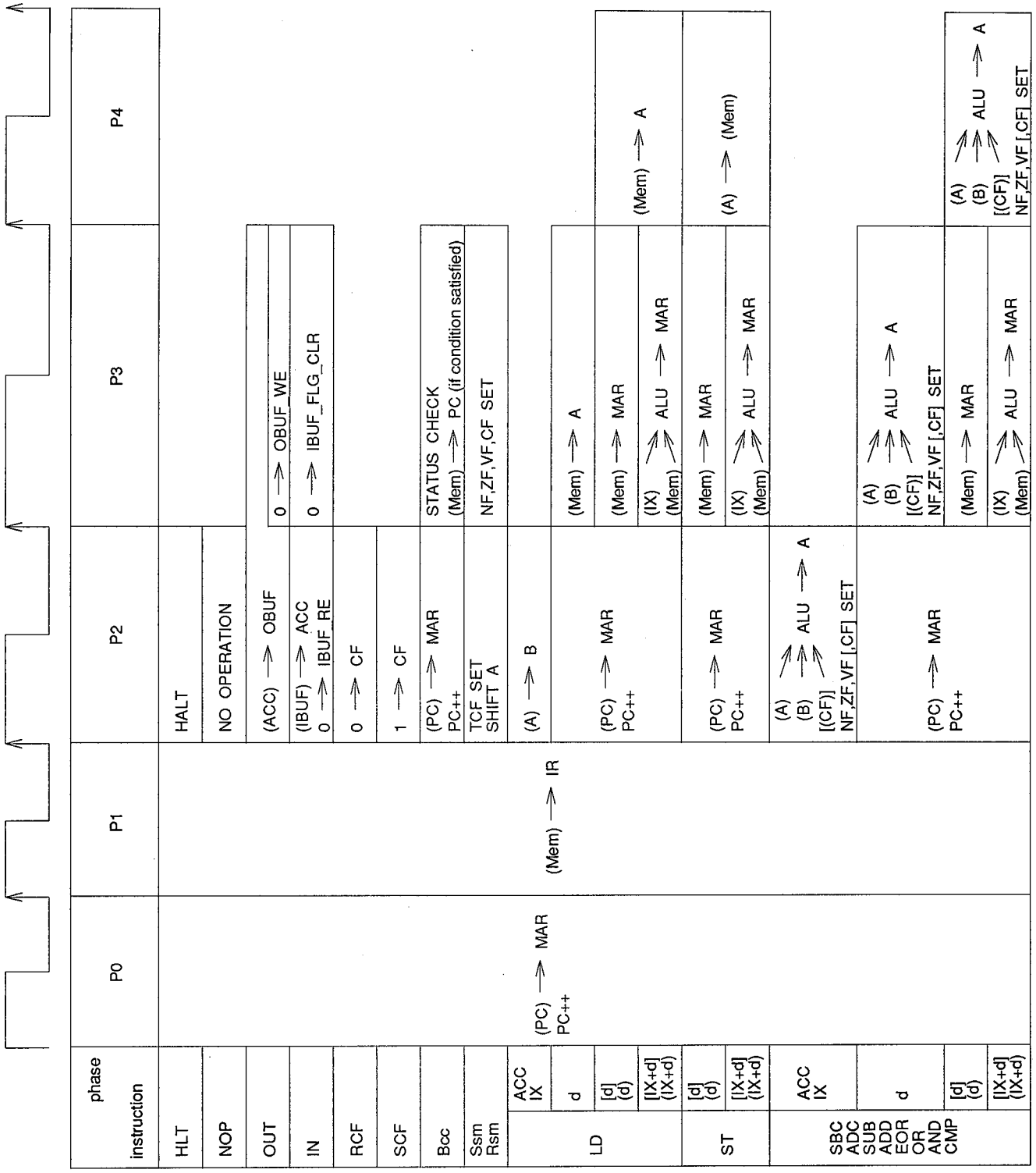
分岐命令

BA	30		
BVF	38		
BNZ	31	BZ	39
BZP	32	BN	3A
BP	33	BZN	3B
BNI	34	BNO	3C
BNC	35	BC	3D
BGE	36	BLT	3E
BGT	37	BLE	3F

制御命令

NOP	00
HLT	0F
OUT	10
IN	1F
RCF	20
SCF	2F

7.6 命令実行フェーズ


phase
instruction
P0
P1
P2
P3
P4
HLT
HALT
NOP
NO OPERATION
OUT
(ACC) → OBUF
0 → OBUF_WE
IN
(IBUF) → ACC
0 → IBUF_RE
0 → IBUF_FLG_CLR
RCF
0 → CF
SCF
1 → CF
Bcc
(PC) → MAR
PC++
STATUS CHECK
(Mem) → PC (if condition satisfied)
Ssm
Rsm
TCF SET
SHIFT A
NF,ZF,VF,CF SET
(PC) → MAR
PC++
(Mem) → IR
(A) → B
LD
ACC
IX
d
[d]
(d)
[IX+d]
(IX+d)
(Mem) → A
(Mem) → MAR
(IX)
(Mem)
ALU → MAR
(Mem) → A
ST
(A) → (Mem)
SBC
ADC
SUB
ADD
EOR
OR
AND
CMP
(A)
(B)
[(CF)]
ALU → A
NF,ZF,VF [,CF] SET

7.7 KUE-CHIP2 ブロックダイヤグラム