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

H4.SMR. 403/14

FIFTH COLLEGE ON MICROPROCESSORS: TECHNOLOGY AND APPLICATIONS IN PHYSICS

2 - 27 October 1989

The Colombo '84 Kernel

C. VERKERK
CERN, Geneva, Switzerland
A. COLAVITA
Microprocessors Laboratory, ICTP, Trieste

CASE STUDY.

- WE WILL INVESTIGATE THE [illegible] BOARD.
- EXAMPLE OF A
  - SIMPLE
  - CHEAP
  - DEDICATED

  APPLICATION OF A 6809 MICROPROCESSOR.
- IN NEXT WEEK'S PROJECTS YOU MAY DEVELOP APPLICATIONS FOR THIS BOARD.
- DEVELOPMENT CAN BE DONE USING THE ROSY STATIONS.
- A FEW OF THE PROGRAMS DEVELOPED WILL BE PUT IN EPROM.
- BOARD IS VERY SIMPLE, BUT WE WILL EXPAND A LITTLE ON DESIGN PRINCIPLES.
- ANOTHER OCCASION TO STRESS THE IMPORTANCE OF:
  - TOP-DOWN DESIGN
  - STRUCTURED DESIGN AND PROGRAMMING
  - STEPWISE REFINEMENT
  - [illegible]

PROGRAM DOCUMENTATION

MOST DOCUMENTATION SHOULD BE DONE DURING
- SPECIFICATION — WHAT
- DESIGN — HOW

PHASES OF DEVELOPMENT.

WRITE READABLE PROGRAMS!

A PROVEN SOLUTION IS TO WRITE

SELF-DOCUMENTING PROGRAMS

WHEN A PROGRAM HAS BEEN BROKEN DOWN INTO MODULES, THEN PROCEED BY WRITING THE CODE AT THE SAME TIME (OR EVEN AFTER) WRITING THE DOCUMENTATION AS PART OF THE SOURCE CODE

EACH MODULE AND EACH ROUTINE IN THE MODULE SHOULD BE PRECEDED BY A

PROLOGUE.

THE PROLOGUE EXPLAINS – IN ENGLISH – A NUMBER OF THINGS:

PROLOGUE:

- WHAT THE MODULE DOES
- HOW IT DOES IT (ALGORITHM)
- DATA STRUCTURE(S) USED
- LOCAL AND GLOBAL VARIABLES
- LIMITS ON DATA
- INPUT PARAMETERS
- OUTPUT PARAMETERS
- USE OF MACHINE REGISTERS
- SPECIAL CONDITIONS
- ERROR RECOVERY
- SIDE EFFECTS
- VERSION, DATE, ETC.

THE PROLOGUE SHOULD CONTAIN EVERYTHING SOMEBODY WHO WILL HAVE TO FIX A BUG, OR MODIFY THE PROGRAM, SHOULD KNOW.

A GLOBAL DATA STRUCTURE SHOULD BE EXPLAINED AT LEAST ONCE, AT THE START OF THE MODULE.

YOU MAY REPEAT IT IN THE ROUTINES ACCESSING THE DATA STRUCTURE.

IT SHOULD GIVE INFORMATION ON POINTERS, WHAT THEY DO, HOW THEY CHANGE, ETC.

PICTORIAL INFORMATION MAY BE INCLUDED.

PROBLEM FOR SMALL SYSTEMS:
SOURCE CODE MAY NEED 30-100 BYTES

- MAKE THEM TERSE AND MEANINGFUL.

- `LDA #1` LOAD 1 INTO A REGISTER
  IS USELESS AS A COMMENT
- `LDA #100` INITIALIZE LOOP COUNTER
  MAKES SENSE.

WRITE COMMENTS WHICH ADD INFORMATION OR EXPLAIN WHAT THE INTENTION IS.

COMMENTS MAY BE:

- IN "PLAIN" ENGLISH
- IN A HIGH-LEVEL PROGRAMMING LANGUAGE.
  READABILITY IS THE CRITERION, NOT SYNTACTICAL CORRECTNESS.
  THIS IS IMPORTANT FOR THE CONTROL OF FLOW CONSTRUCTS.
  WRITING THE COMMENTS FIRST, AND THEN THE CODE, FORCES THE PROGRAMMER TO WRITE STRUCTURED CODE, IF THE COMMENTS WERE IN A STRUCTURED LANGUAGE.

ONE-TO-ONE RELATION BETWEEN COMMENTS AND CODE IS EXTREMELY IMPORTANT.

COLOMBO 84

THE COLOMBO 84 BOARD WAS DESIGNED BY

IAN BARNETT

WHO ALSO LOOKED AFTER MANUFACTURING MOUNTING, TESTING ETC.

HE WROTE MOST OF THE SOFTWARE ALSO.

OTHER CONTRIBUTIONS FROM A NUMBER OF PEOPLE (RAICH, GOVIND)

POSSIBLE PROJECTS:

- SIMPLE COUNTER
- UP/DOWN COUNTER
- INTERVAL TIMER
  - PULSE WIDTH
  - REACTION TIME TESTER
- FREQUENCY METER
- FREQUENCY GENERATOR
- DIGITAL VOLTMETER
- DIGITAL CLOCK
  - STOP WATCH
  - ALARM CLOCK

AND POSSIBLY MORE.

SUMMARY OF HARDWARE.

TWO PARTS:

1. 6809 PROCESSOR + 2K RAM
   + 2K EPROM
   + SOCKET FOR EPROM

   ADDRESS DECODING:

RAM	$1=800	0800, 1800, 2800
EPROM2	$F800	C800, D800, E800
(EPROM1	$F000	C000, D000, E000)

   PIA WITH OWN ADDRESS DECODING: $7800

   PROCESSOR PART CAN BE STOPPED WITH JUMPER.

2. I/O PART: PLUG IN PARALLEL WITH PIA.
   - A-SECTION OF PIA USED TO DRIVE 7-SEGMENT LED DISPLAYS.
     4 LOW-ORDER BITS SELECT THE DIGIT LATCH
     4 HIGH-ORDER CONTAIN THE DIGIT (DECIMAL)
     TO ENTER DATA, PULL THE BIT DOWN, THEN PUT IT UP AGAIN.
   - PROCESSOR CLOCK IS DIVIDED BY $2^{15}, 2^{14}, 2^{13}$
     JUMPERS SELECT 50, 100 AND 200 Hz TO CA1
   - B-SECTION OF PIA:
     2 LOW BITS: PUSH-BUTTONS
     2 NEXT: SWITCHES
     4 HIGH ORDER BITS: ROTARY SWITCH
   - VOLTAGE TO FREQUENCY CONVERTER, CONNECTED TO CB1
   - IRQA AND IRQB FROM PIA STRAPPED TO FIRQ AND

SOFTWARE FOR COLOMBO 84

- SOFTWARE FOR COLOMBO 84 SHOULD BE DEVELOPED BY THE PARTICIPANTS.
- YOU CHOOSE ONE OR MORE OF THE APPLICATIONS AND IMPLEMENT THEM.
- THERE ARE VARIOUS OPTIONS:

1. YOU WRITE EVERYTHING.
   VERY GOOD DIDACTICALLY:
   - YOU DISCOVER THE THINGS YOU MUST THINK OF,
   - YOU FIND THE DIFFICULTIES,
   - YOU FIND THAT WRITING A PROGRAM TO RUN ON ROSY IS QUITE DIFFERENT FROM DEVELOPING A PROGRAM TO RUN ELSEWHERE.

   BUT: WE WOULD HAVE TO MAKE 85 DIFFERENT EPROMS, ALL THE LAST DAY.
2. YOU WRITE YOUR SOFTWARE, BUT YOU GO HOME WITH THE "STANDARD EPROM".
   NO POSSIBILITY TO IMPLEMENT YOUR OWN IDEAS AND TAKE THEM HOME.
3. YOU WRITE APPLICATIONS, USING A STANDARD KERNEL.
   ONE EXAMPLE OF EACH APPLICATION IS SELECTED TO MAKE UP A STANDARD EPROM TO BE REPRODUCED 85 TIMES.

KERNEL.

SPECIFICATIONS:

THE KERNEL MUST PERFORM THE FOLLOWING FUNCTIONS:

1. INITIALISE THE SYSTEM, FOLLOWING "RESET" OR "POWER ON".
2. FIND THE USER PROGRAM TO BE EXECUTED ACCORDING TO THE SETTING OF THE ROTARY SWITCHES. START EXECUTION OF THE SELECTED PROGRAM, AFTER DISPLAY OF NUMBER.
3. PROVIDE THE DISPLAY ROUTINES, INCLUDING BINARY-TO-BCD CONVERSION.
4. GIVE AN ERROR INDICATION WHENEVER
   - A NON DEFINED INTERRUPT OCCURS
   - THE REQUESTED PROGRAM CANNOT BE FOUND.

IN ADDITION:

5. IT MUST BE POSSIBLE TO LOAD A USER PROGRAM AT ANY PLACE IN RAM OR ROM, WITHOUT THE NEED FOR THE USER TO MODIFY THE KERNEL, OR TO RE-ASSEMBLE THE KERNEL.
6. THE KERNEL MUST BE ABLE TO RUN EITHER IN ROSY, OR IN AN EPROM ON THE "COLOMBO 84" BOARD.
   TO TRANSPORT THE KERNEL FROM ROSY TO "COLOMBO 84" MINIMAL CHANGES SHOULD BE NECESSARY (CHANGE 1 LINE; ASSEMBLE)

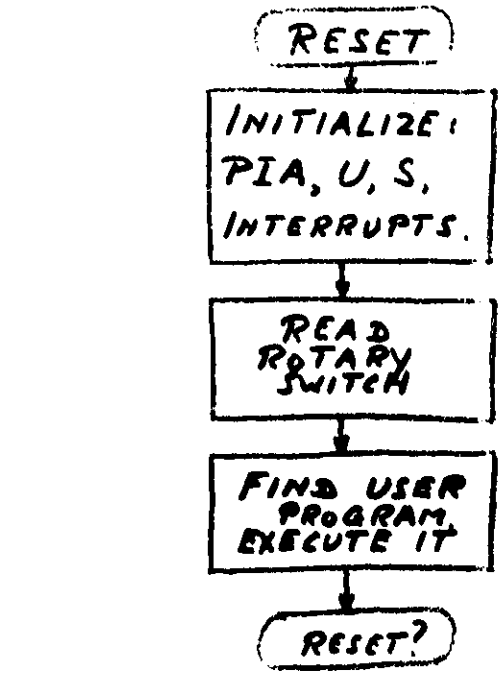
7. THE KERNEL MUST PROVIDE DEFAULTS FOR EVERYTHING THE USER MAY FORGET TO SPECIFY (SOME DEFAULTS MAY RESULT IN ERROR CONDITIONS.)

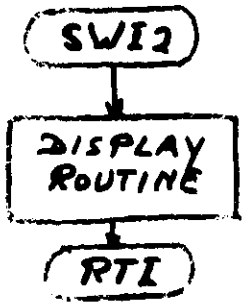
INTERFACE BETWEEN USER PROGRAM AND KERNEL:

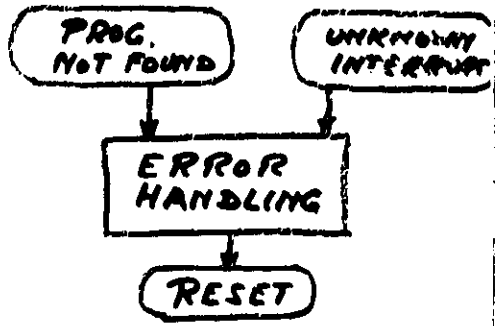
8. THE USER PROGRAM MUST HAVE A VERY SIMPLE "HEADER", ENABLING THE KERNEL TO DETECT ITS PRESENCE AND TRANSFER CONTROL TO IT.
9. THE USER PROGRAM MUST HAVE EASY ACCESS TO THE DISPLAY ROUTINE, WITHOUT THE NEED TO KNOW ITS ADDRESS, OR TO ASSEMBLE THE USER PROGRAM TOGETHER WITH THE KERNEL.

THESE ARE THE SPECIFICATIONS.
- WHAT DO THEY IMPLY?
- HOW SHOULD THE KERNEL BE DESIGNED?

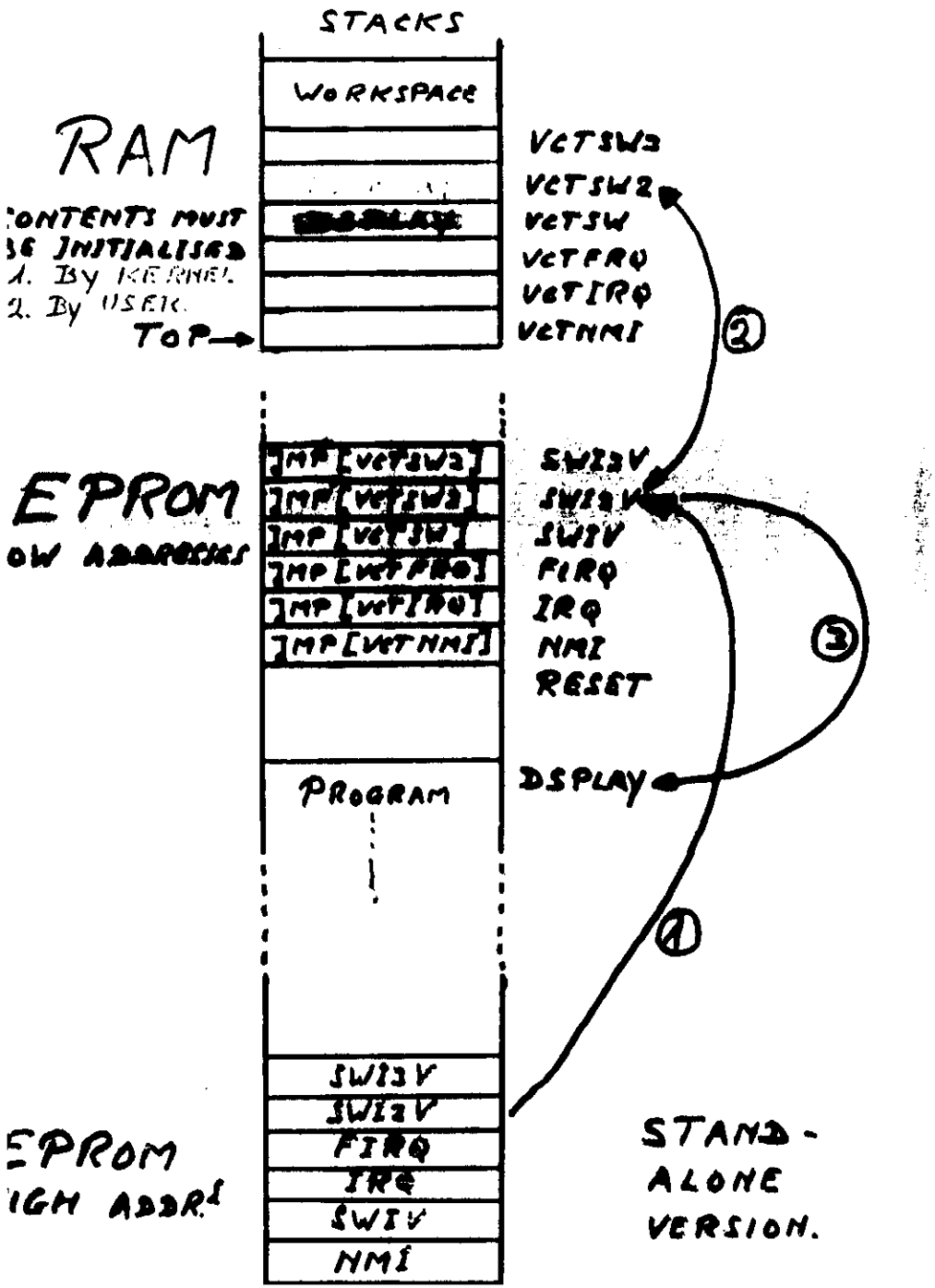
FLOW DIAGRAM:







INTERRUPT VECTORING.



INTERRUPT VECTORS

THE 6809 FINDS ADDRESSES WHERE TO START EXECUTION AFTER AN INTERRUPT OR RESET AT THE TOP OF MEMORY, e.g. IN EPROM. INTERRUPT VECTORS CANNOT BE EASILY CHANGED.

TO OVERCOME THIS DIFFICULTY, WE GO IN STEPS:

IN FFFA/FFFB : AN ADDRESS IN EPROM: FIRQ

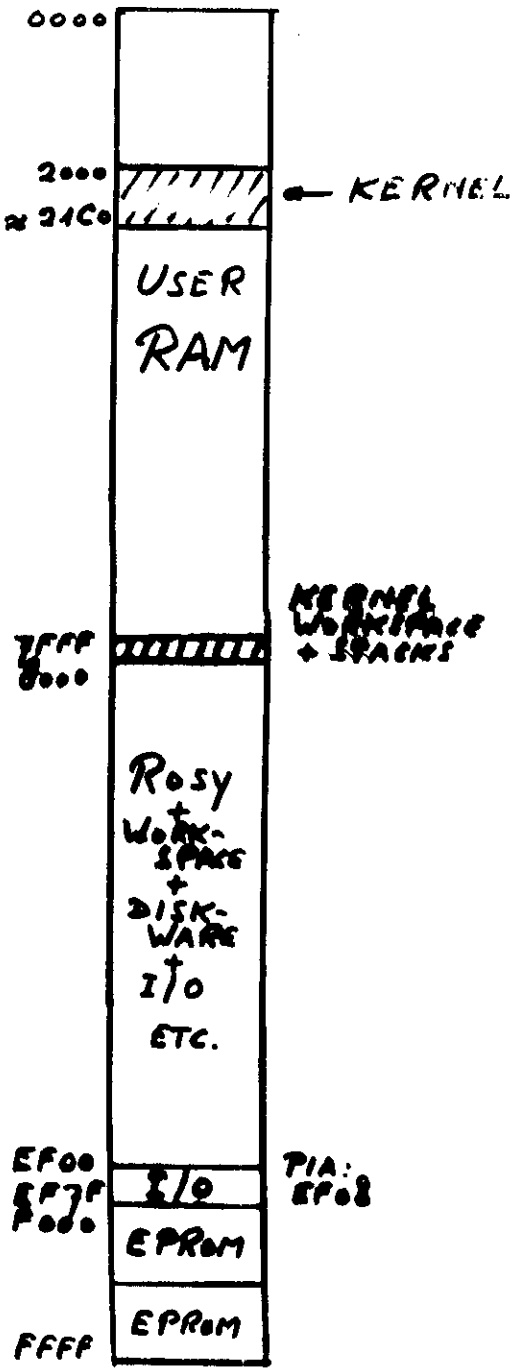
AT FIRQ : JMP [VCTFRQ]

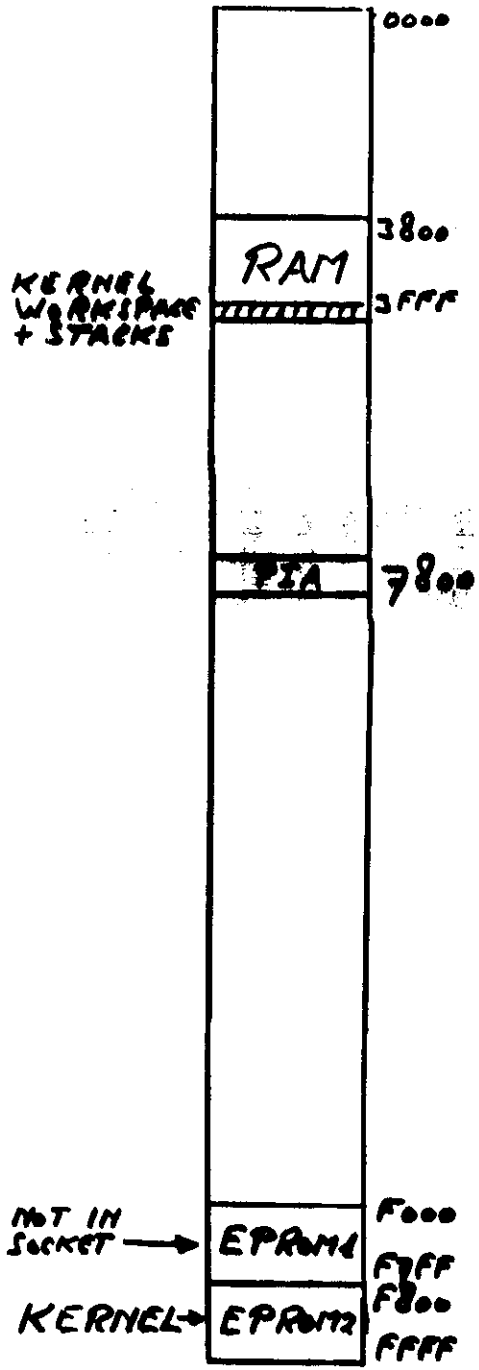
VCTFRQ IS AN ADDRESS IN RAM. IN THIS ADDRESS YOU MAY WRITE THE FINAL DESTINATION ADDRESS.

```
                    NAM     BLABLA
            RAM     EQU     $400
          RAMTOP    EQU     RAM+$7FF
          VCTFRQ    EQU     RAMTOP-5
BFR       EPROM     EQU     $F000
IN RAM!
                    ORG     EPROM
F000 JMP BFR4  FIRQ JMP     [VCTFRQ]
                    ORG     $FFFA
FFFA F000      VFIRQ FDB    FIRQ
```

WRITE THE FINAL DESTINATION ADDRESS IN VCTFRQ (=$BFA)

IF CONTENT OF [illegible] IS, FOR INSTANCE $235A WE WILL END UP THERE.





WHEN COMPARING THE TWO MEMORY MAPS, WE NOTE:

- IN ROSY KERNEL CANNOT RUN AT TOP OF MEMORY, JUST BELOW $FFFF.
- THE PIA ADDRESSES ARE DIFFERENT.
- THE KERNEL CANNOT CHANGE INTERRUPT VECTORS IN ROSY IN THE SAME WAY AS ON THE ICTP BOARD.

WE MUST HAVE TWO DIFFERENT VERSIONS OF THE KERNEL.

NEVERTHELESS IT IS WORTH WRITING THE KERNEL IN POSITION INDEPENDENT CODE. WE CAN THEN LOAD IT WHEREVER IT IS CONVENIENT.

WHEN THE KERNEL MUST RUN UNDER ROSY, ESSENTIALLY 3 THINGS MUST BE CHANGED:

- RAM, PIA ADDRESSES, AND ORG.
- INTERRUPTS.
  THEY REMAIN UNDER CONTROL OF ROSY, UNTIL CHANGED BY MONITOR CALL.
  EXCEPTION: SWI2 MUST LAND IN DSPLAY (DISPLAY ROUTINE) IN THE KERNEL
- FINDING OF A USER PROGRAM.
  IN ROSY ONLY RAM IS SEARCHED.

CONDITIONAL ASSEMBLY IS USED TO MAKE THE CHANGES.

```
        ROSY    EQU   1
        IBB     EQU   2

        SYSTEM  EQU   ROSY
OR:     SYSTEM  EQU   IBB

        IFN   SYSTEM-IBB
        CODE FOR
        STAND-ALONE
        ENDIF
        IFN   SYSTEM-ROSY
          CODE FOR RUNNING UNDER
          ROSY
```

POSITION-/INDEPENDENT CODE IS USEFUL FOR ROMs.

YOU CARRY YOUR ROM WITH YOU, PLUG IT IN ANY SOCKET AND IT WILL WORK.

```
        LDA   #89
        ASLA
        ADCA  VAR,PCR
        STA   VAR,PCR

VAR     RMB   1
```

THIS CODE IS POSITION INDEPENDENT.

CAN IT BE USED IN A ROM?

SOLUTION:

```
        NAM   BLABLA
RAM     EQU   $800
EPROM   EQU   $E000
        :
        ORG   RAM
VAR     RMB   1
BUF     RMB   672
        :
        ORG   EPROM+$7FF
BEGIN   LEAX  3,U
        STX   6,S
        :
        LDA   #8g
        ASLA
        ADCA  VAR
        STA   VAR
        :
        END
```

IF YOU WANT TO RUN YOUR PROGRAM IN AN EPROM AT ANY PLACE, YOU MUST USE ABSOLUTE RAM ADDRESSES FOR YOUR VARIABLES (BUT NOT FOR THE CONSTANTS).

STARTING EXECUTION OF A USER PROGRAM.

TO BE ABLE TO EXECUTE A USER PROGRAM ANYWHERE IN MEMORY, IT SHOULD BE PRECEDED BY A HEADER.
THE KERNEL WILL READ THE DIP SWITCHES AFTER RESET AND THEN FORM A STRING OF ASCII CHARACTERS:

PRG @

WHERE @ STANDS FOR 0-9, A-F.
@ HAS JUST BEEN READ FROM DIP SWITCHES.

THIS IS PREFERRED TO A JUMP TABLE.
A JUMP TABLE WOULD REQUIRE
- EITHER MANUAL INTERVENTION
- OR ASSEMBLING ALL PROGRAMS TOGETHER

CONSEQUENCE:
A USER PROGRAM MUST START WITH

FCC /PRG@/ (OR PRGA, ETC)
ANY EXECUTABLE STATEMENT

No RMB, FDB OR OTHER FCC ARE ALLOWED BETWEEN THESE TWO LINES.

SMALL PROBLEM: WE MAY FIND PRG@ IN THE EDIT BUFFER!

CALLING THE DISPLAY ROUTINE.

FOR USER CONVENIENCE, THE DISPLAY ROUTINE IS PROVIDED IN THE KERNEL. THE USER DOES NOT NEED TO KNOW THE ABSOLUTE ADDRESS OF THE ROUTINE, AND N' RE-ASSEMBLY OF USER PROGRAM AND KERNEL TOGETHER IS NEEDED.
THE DISPLAY ROUTINE IS ENTERED WITH

SWI2

TO DISPLAY A POSITIVE NUMBER USE THE FOLLOWING SEQUENCE:

```
LDY   NUMBER
LDB   #0 IF BINARY → BCD CONVERSION IS NEEDED
      #1 IF NO CONVERSION IS REQUIRED
SWI2
```

THE KERNEL WILL INITIALIZE ITSELF TO ROUTE SWI2 TO THE DISPLAY ROUTINE. WHEN RUNNING IN PESY, THE KERNEL WILL MAKE A MONITOR CALL TO CHANGE THE SWI2 VECTOR.
THE STAND-ALONE VERSION WRITES THE ADDRESS OF THE DISPLAY ROUTINE IN THE APPROPRIATE PLACE IN RAM.
NOTE: ALL NUMBERS ARE CONSIDERED TO BE POSITIVE!

BINARY → BCD CONVERSION AND VICE-VERS

MOST PEOPLE CONTINUE TO LIVE IN A DECIMAL WORLD. THEY WANT TO INPUT DATA IN DECIMAL FORM AND GET READABLE OUTPUT FROM THEIR COMPUTERS.
THE COMPUTER IS VERY WELL SUITED TO MAKE THE NECESSARY CONVERSIONS.

1. INPUT OF NUMBERS.

THE TERMINAL WILL DELIVER ASCII CHARACTER STRINGS, EVEN IF WE TYPE IN A NUMBER.

- SINGLE DIGIT NUMBER:
  SUBTRACT $30 FROM ASCII CODE; THIS GIVES BINARY REPRESENTATION.
  (MAKE SURE IT WAS A NUMBER!)

- A MULTIPLE DIGIT NUMBER MUST BE CONVERTED.
  - FIRST CONVERT EACH DIGIT → BINARY
    THIS "BINARY" IS IN FACT
    BCD: BINARY CODED DECIMAL
    1 DIGIT OCCUPIES 4-BITS:

0000	"0"	1000	"8"
0001	"1"	1001	"9"
⋮			
0111	"7"	OTHERS ARE INVALID	

  - THEN USE THE FORMULA:

$$V = (\dots((D_k \cdot 10 + D_{k-1}) \cdot 10 + D_{k-2}) \cdot 10 + \dots D_1) 10 + D_0$$

THIS REQUIRES MULTIPLICATIONS.
YOU MAY USE A TRICK:

$10 * a = 8 * a + 2 * a$

SO, CONTENTS OF A CAN BE QUICKLY AND BASILY MULTIPLIED BY 10:

```
ASLA          MULTIPLY BY 2
STA   0,S     STORE TEMPORARILY
ASLA
ASLA          NOW YOU MULTIPLIED BY 8
ADDA  0,S     ADD TOGETHER 8*A+2*A
```

2. **OUTPUT OF NUMBERS.**

WE NOW MUST CONVERT A BINARY NUMBER INTO A STRING OF BCD OR ASCII CHARACTERS.
THERE ARE AT LEAST THREE METHODS:

i) DIVIDE BY 10 ; REMAINDER = LAST DECIMAL DIGIT
DIVIDE QUOTIENT BY 10 ; REM = NEXT DIGIT
ETC.
THIS NEEDS DIVISION.
IN MICROPROCESSOR PRACTICE DIFFICULT!

ii) REPEATED SUBTRACTION OF POWERS OF 10.
EASY TO UNDERSTAND,
BUT EXECUTION TIME IS VARIABLE
AND CODE IS RATHER LONG

IN PASCAL

```
BEGIN
  RESULT := DATA
  FOR p = max TO 1 DO
    BEGIN
      COUNT := 0
      WHILE RESULT ≥ 10**p DO
        BEGIN
          RESULT := RESULT - 10**p
          COUNT := COUNT + 1
        END
      NEXT-DIGIT := COUNT
    END
  NEXT-DIGIT := RESULT
END
```

SO, FOR A 16-BIT NUMBER:
REPEATEDLY SUBTRACT 10 000
THEN 1 000
THEN 100
THEN 10
REST IS THE LAST DIGIT.

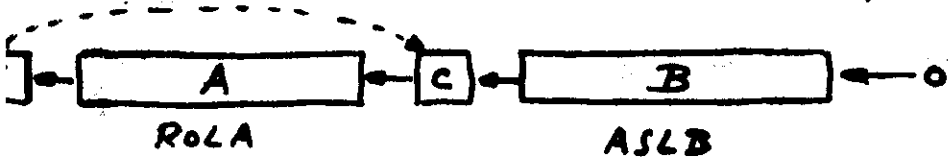
(iii) BUILD UP THE BCD NUMBER, USING THE FAMOUS FORMULA:

$$V = (\ldots((b_{n-1}\cdot 2 + b_{n-2})\cdot 2 + b_{n-3})\cdot 2 + \ldots b_1)\cdot 2 + b_0$$

AND USING DECIMAL ARITHMETIC.

STP 1 – FIND NEXT BIT b_{next} OF BINARY NUMBER, STARTING FROM MOST SIGNIFICANT, BY SHIFTING IT INTO THE CARRY BIT.

ASLB	ASL 3,S
ROLA	ROL 2,S



STP 2 – $\ldots$ (PARTIAL RESULT)$*2 + b_{next}$

IS DONE AS FOLLOWS:

LDA	1,S	PARTIAL RESULT IS IN 1,S
ADCA	1,S	THIS DOES *2 AND ADDS b_{next} (= THE CARRY)
DAA		MYSTERIOUS SEE BELOW
STA	1,S	STORE NEW PARTIAL RESULT

REPEAT FOR MOST SIGNIFICANT BYTE.

THE DAA INSTRUCTION ASSURES THAT STEP 2 IS DONE IN DECIMAL ARITHMETIC.

– REPEAT STEP 1 AND STEP 2 AS MANY TIMES AS THERE ARE BITS IN THE ORIGINAL DATA.

WHAT DOES DAA DO?

A BYTE CAN REPRESENT:

- A SIGNED NUMBER: $-128 \le N \le 127$
- AN UNSIGNED NUMBER: $0 \le M \le 255$
- AN ASCII CHARACTER
- A BIT PATTERN (MASK)
- A USER-DEFINED OBJECT
- AN OPCODE
- A PRE- OR POSTBYTE
- A BYTE OF AN ADDRESS
- TWO DECIMAL DIGITS

$\longrightarrow 95_{10}$

BIT-PATTERNS $1010 \rightarrow 1111$ ($A \rightarrow F_{16}$) ARE NOT VALID, OF COURSE.

YOU CAN DO ARITHMETIC WITH DECIMAL NUMBERS REPRESENTED THIS WAY:

43_{10}	0100 0011
36_{10} +	0011 0110 +
79_{10}	0111 1001
43	0100 0011
39 +	0011 1001 +
82	0111 1100 $\xrightarrow{?}$ 7?

$10_{10} + \alpha$

WE WANT α IN LOW-ORDER NIBBLE AND "1" ADDED TO HIGH NIBBLE

TRANSFORMING $10_{10}+\alpha$ INTO

$16_{10}+\alpha$ WILL

- PRODUCE THE DESIRED VALUE OF α
- PRODUCE THE CARRY INTO HIGH NIBBLE

SOLUTION: ADD 6 TO THE RESULT (LOW-ORDER NIBBLE):

```
43        0100 0011
38        0011 1000
--+       ---------+
81        0111 1011    RESULT OF ADD
          0000 0110    DAA  ADD CORRECTION
          ---------
          1000 0001
"HALF CARRY" --> (1)
```

- THE DAA INSTRUCTION ADDS TO EITHER, OR BOTH NIBBLES INDEPENDENTLY, A CORRECTION TERM.
- THE CORRECTION IS EITHER "0", OR "6".
- A "HALF-CARRY" GENERATED IN CORRECTING LOW-ORDER NIBBLE IS ACCOUNTED WHEN ADJUCTING HIGH-ORDER NIBBLE
- A "CARRY-IN" INTO LOW NIBBLE IS ACCOUNTED. A "CARRY-OUT" FROM HIGH-NIBBLE SETS C. MULTIPLE PRECISION DECIMAL ARITHMETIC IS POSSIBLE.

ONE MORE EXAMPLE:

```
38        0011 1000
39        0011 1001
--+       ---------+
77        0111 0001  --DAA--> 0111 0111 = 77_10
          (1)
```

THUS: "6" IS ADDED TO LOW NIBBLE WHEN

- EITHER RESULT > 9
- OR H = 1

(NOTE THE ERROR IN "BLUE BOOK"!)

NOW THAT WE KNOW THE EFFECT OF DAA, WE CAN WRITE A BINARY-TO-BCD CONVERSION ROUTINE:

```
        PSHS  D      SAVE DATA ON STACK
        CLR   ,-S    INITIALIZE TWO BYTES ON STACK
        CLR   ,-S    WHICH WILL RECEIVE RESULT
        LDB   #16    LOOP COUNTER
LOOP    ASL   3,S    SHIFT MOST SIGNIFICANT BIT OF
        ROL   2,S    DATA INTO CARRY BIT
        LDA   1,S    LOAD PARTIAL RESULT AND
        ADCA  1,S    DOUBLE IT; ADD NEXT BIT
        DAA          CORRECT FOR DECIMAL ARITHM.
        STA   1,S    STORE PARTIAL RESULT
        LDA   0,S      REPEAT FOR HIGH-BYTE.
        ADCA  0,S      POSSIBLE CARRY IS TAKEN
        DAA            INTO ACCOUNT (WITH ADCA)
        STA   0,S
        DECB           REPEAT 16 TIMES
        BNE   LOOP
```

$4C9 ⟶ 1225$_{10}$

			0	4	C	9
			0000	0100	1100	1001
		1	0000	1001	1001	0010
		0	0001	0011	0010	0100
		0	0010	0110	0100	1000
		0	0100	1100	1001	0000
	0	0	1001	1001	0010	0000
1	1	1	0011	0010	0100	0000
10	2	0	0110	0100	1000	0000
100	4	0	1100	1001	0000	0000
1001	9	1	1001	0010		
DAA → 1 0011 / 1 1001	19	1	0010	0100		
DAA → 0011 0010 / 0011 1000	38	0	0100	1000		
DAA → 0111 0000 / 0111 0110	76	0	1001	0000		
DAA → 1110 1101 / 0101 0011	153	1	0010	0000		
10 1010 0110	306	0	0100	0000		
11 0000 0110						
110 0001 1000	612	0	1000	0000		
110 0001 0010						
1100 0010 0101	1225	1	0000	0000		
0010 0010 0101						

ADVANTAGES OF THIS CONVERSION:

- FIXED TIME, INDEPENDENT OF DATA
- SHORT ROUTINE
- NO STORAGE FOR CONSTANTS NEEDED

ERROR INDICATIONS.

TWO SORTS OF ERRORS ARE DETECTED:

- AN INTERRUPT FOR WHICH NO INTERRUPT ROUTINE IS PROVIDED. ERROR NUMBERS 0-4 ARE ASSIGNED TO THE DIFFERENT INTERRUPTS
- A PROGRAM CANNOT BE FOUND. ERROR NUMBER = 9.

THE ERROR NUMBERS ARE DISPLAYED IN "TIMES SQUARE" FASHION. THEY RUN FOREVER, UNLESS STOPPED BY RESET.