Orton NIBL computer with PAGE2 support

Manual

Index

Disclaimer

Karen Orton 2020

Chapter 1 Overview

The computer described in this document was inspired by an SC/MP-based computer that I built in the late 1970s. This computer ran National Semiconductor's 'National's Industrial BASIC Language' ('NIBL'), a 4k byte tiny BASIC intended to ease application of the SC/MP to industrial control applications. There are several distinctive features of this tiny BASIC:

1. NIBL programs are not tokenised when entered into memory.

2. NIBL programs are not scanned for syntax compliance during editing.

3. NIBL programs recognise the inherent paging of the SC/MP microprocessor, leading to as many as seven independent (but chainable) 4 k byte programs in memory.

4. NIBL is able to identify a ROM-based program ('Page 2') and run this automatically at start up.

These observations inspired the development of a set of utility routines to accompany NIBL. These work in such a way as to complement NIBL rather than compete with it. For example, some of these routines take a NIBL page number as their argument. This document describes the package ('PAGE2') and the commands it supports.

Chapter 2 Hardware expectations

The PAGE2 system makes use of the SC/MP processor flags as follows:

Flag	NIBL use	PAGE2 use
F0	Console serial output (inverted)	Same as NIBL
F1	Teletype reader relay	General I/O indicator LED
F2	(not used)	Auxiliary serial output (inverted)
SA	(not used)	Auxiliary serial hardware flow control input (CTS)
SB	Console serial input	Same as NIBL
SIN	(not used)	FSK cassette replay input
SOUT	(not used)	FSK cassette record output

PAGE2 resides in ROM on Page 2. PAGE2 assumes a maximum of 24k bytes of RAM (in Pages 1, 3, 4, 5, 6 and 7). Memory addresses #8000 and above are available for system expansion. Console serial I/O is performed at 1200 Baud, 8 data bits, no parity and one stop bit.

Chapter 3 PAGE2 commands

General

PAGE2 has eight commands as follows:

Command	Arguments	Function
NIB	none	Return to NIBL
DIR	none	Directory of pages
DMP	<address>	Dump memory
CHM	<address>	Change memory
SAV	<page>	Save page to cassette
LOD	<page>	Load page from cassette
PRN	<page>	Print page to auxiliary port
ASM	<page>	Assemble page

Note that commands are identified by the first three characters. Therefore, the 'NIB' command could be entered as 'NIBL', 'SAV' as 'SAVE' etc. In PAGE2, all numeric input follows NIBL convention: either decimal, or hexadecimal preceded by a hash '#'.

NIB

This command causes a return to NIBL with the current page being 2. PAGE2 can be re-entered by simply entering 'RUN' while on Page 2. On either change, the new environment is identified ('PAGE2' or 'NIBL'). While in the PAGE2 environment, the prompt is an asterisk '*'. The NIBL prompt is of course a greater than sign '>'.

DIR

The DIR command shows the first program line of Pages 1 to 7, if present. This might assist in identifying loaded programs. A typical response to the DIR command might be:

```
1: REM SYMTAB
2: REM PAGE2.SYS
3: FOR N=1 TO 10
4: REM SQUARE ROOT
5:
6:
7:
```

DMP <address>

The DMP command dumps the contents of memory in tabular form. The dump begins from the address supplied, or the previous boundary consistent with the number of bytes dumped per line. The contents are displayed as both hexadecimal and ASCII characters, provided the latter correspond to printable ASCII characters. A typical response to the DMP command is:

```
DMP #4002
4000 00 04 05 06  ....
4004 30 31 32 33  0123
4008 00 00 00 00  ....
400C 00 00 00 00  ....
4010 00 00 00 00  ....
4014 00 00 00 00  ....
4018 00 00 00 00  ....
401C 00 00 00 00  ....
4020 00 00 00 00  ....
4024 00 00 00 00  ....
4028 00 00 00 00  ....
402C 00 00 00 00  ....
-
```

The final line shows a minus symbol '-'. This is the in-command PAGE2 prompt and accepts the following inputs:

	Return alone: Dump next block
.	Dot return: Exit DMP command
@<address>	Dump from new address

CHM <address>

The CHM command allows editing of memory contents. A typical response to the CHM command is:

```
CHM #4040
4040 04 -
```

Again, the minus sign is an in-command PAGE2 prompt and accepts the following inputs:

	Return alone: Show next location
.	Dot return: Exit CHM command
<new contents>	Enter new memory contents
@<address>	Show contents at new address

SAV <page>

The SAV command records the contents of the specified page to cassette using Frequency Shift Keying (FSK), at about eighty characters per second. A header is first recorded to make start identification easier for the load process. The size of the program is determined automatically by finding the NIBL end-of-program termination. The I/O indicator LED is illuminated during program save.

LOD <page>

The LOD command loads a program from cassette. The I/O indicator LED is illuminated when the header of a saved program is detected. The size of a program is recorded during save, and this is used on load to determine whether the program will fit into the chosen page. The LOD command terminates before loading commences, if the proposed cassette file is too large for the selected page. This can occur when a very large program is saved from Page 3, 4, 5, 6 or 7, and then subsequently re-loaded to Page 1.

PRN <page>

The PRN command lists the selected page at the auxiliary serial port. The listing format is identical to the NIBL LIST command. The expectation is that a printer is connected to the auxiliary port, thereby generating hardcopy. The auxiliary serial port is hard coded for 1200 Baud, 8 data bits, no parity and one stop bit. Hardware flow control is implemented.

ASM <page>

The ASM command assembles SC/MP machine mnemonics in the NIBL program file on the selected page. This is a very simple and minimal assembler whose statements and features are described in the next section.

Error messages

Message	Meaning
?WHAT?	Command not recognised
?SYNTX	Syntax error
?EMPTY	Specified page is empty
?PAGNO	Specified page is invalid
?ABORT	User has aborted command
?BADCS	Loaded cassette file has bad checksum
?BADSZ	Loaded cassette file too big for page
?OVFLO	Number argument exceeds 16 bits
?OPCOD	Unrecognised opcode
?NOLBL	Specified label or constant not found
?JMPSZ	Jump distance too big

When running the assembler, the errant line is identified by the error message, e.g.:

```
*ASM 4
PASS 1:
?SYNTX IN 230
*
```

Chapter 4 Assembler

The PAGE2 assembler is intended to complement the NIBL language by adding integrated binary code generation. Assembler mnemonics, along with a few assembler directives, permit the generation of this binary in the memory following a NIBL program. The assembler does not support symbolic constants. Instead, line numbers serve as both constants and labels. The PAGE2 assembler has only four directives:

Directive	Function	Example
REM	Turns line into a comment	`REM LOAD POINTER 1`
EQU	Defines a constant	`30 EQU 26`
SPC	Reserve space	`20 SPC 80`
BYT	Generates values in the binary	`40 BYT 1,2,3,"STRING"`

The EQU directive allows commonly referenced constants to be associated with a line number. In the example above, for example, a constant defined as 26 could be referenced using '$30'. The SPC directive reserves a block of memory of up to 255 bytes in size. This is useful for statements of the kind INPUT $N which need some memory to store a user response.

Assembler operands

The PAGE2 assembler has no expression capability. This means that operands are strictly limited to simple decimal or hexadecimal values (the latter preceded by a hash '#'). The operand field is parsed by first looking for the following prefixes:

Prefix	Effect
-	Sign change
$	Symbolic constant look-up
H	High byte of symbolic constant
L	Low byte of symbolic constant
@	Adjusts opcode for auto-indexed operation

Symbolic constant look-up determines the address associated with a line number and is useful in two contexts: calculating displacements for jump instructions, and calculating displacements for PC-relative indexed instructions. Examples:

```
50 REM JUMP TO 'LINE' 100
60 JMP $100

30 REM LOAD FROM SPACE IN LINE 100
40 LD $100
..
..
..
100 SPC 1
```

Parsing concludes with a search for one of the following postfixes:

Postfix	Effect
(0)	Adjusts opcode for required pointer
(1)	
(2)	
(3)	

This minimal parser generates SC/MP binary correctly but can be confused by incongruous combinations e.g.:

```
50 LDI  -3(1)
```

... which actually generates:

```
50 LD   @-3(1)
```

Note that offset calculations for jump instructions are correctly calculated, provided the targets are of the form $<line number>.

Comments

The REM directive is recognised for the purpose of turning an entire line into a comment. Comments can also be added after directives, opcodes and operands, provided all expected fields are present. Failing this, the assembler might attempt to parse the comment field.

IMPORTANT

The assembler needs memory in which to store a symbol table. Page 1 is used for this purpose, and any program previously resident in Page 1 will be destroyed by use of the assembler. On completion of assembly, Page 1 will be found to hold a single NIBL program line:

```
1 REM SYMTAB
```

Development life cycle

The combined NIBL/assembler development lifecycle is as follows:

A program combining NIBL statements and assembler mnemonics is entered on the desired page. NIBL statements appear first. Assembler mnemonics follow after the NIBL statements, the start of these being identified by a line beginning 'ASM', e.g.

```
10 REM PROGRAM
20 FOR N=1 TO 10
30 LINK TOP
40 NEXT N
50 END
60 ASM
70 CSA
80 XRI #02
90 CAS
100 XPPC P3
```

The assembler locates the 'ASM' statement in order to determine the beginning of the assembler code. Executable instructions are automatically written to the free memory immediately following the program text.

There are two important issues to keep in mind: Firstly, ANY change to the textual part of the program (i.e. NIBL or assembler) may corrupt executable instructions. If any edits are performed, then the assembler must be re-run to regenerate the executable binary. Secondly, loading a program from tape will require that the assembler be run if the loaded program contains assembler code.

Access to executable code should use NIBL statements of the format LINK(TOP+N). Multiple subroutines can be called this way, provided the executable code begins with a table of JMP instructions to the routines provided. Parameters should be passed and returned using NIBL variables, which are at convenient offsets from SC/MP pointer 2.

Chapter 5 Hardware

A hardware platform for testing the PAGE2 system was constructed which employs a PIC emulation of a 4MHz SC/MP chip. This emulation is 'cycle perfect' and so accurately approximates a real SC/MP chip in terms of instruction execution time.

The computer has an integrated cassette transport . A relay determines the play/record state, and is energised by a press button. The relay is de-energised on cassette ejection so as to prevent inadvertent over-recording.

The computer has a single 8k byte EPROM which maps to the SC/MP address space as follows:

EPROM range	SC/MP range
0000 - 0FFF	0000 - 0FFF
1000 - 1FFF	2000 - 2FFF

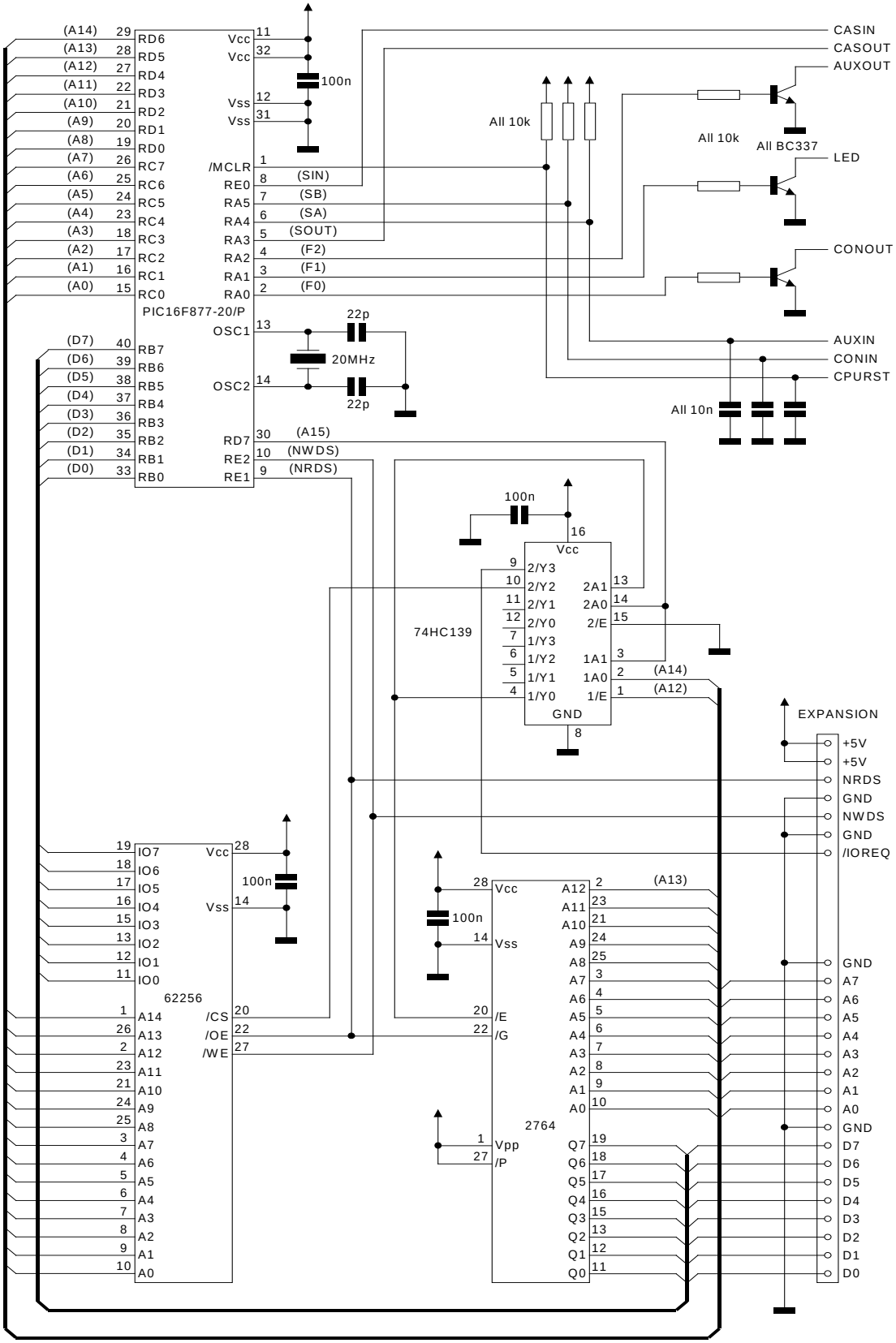
Consequently, the lower half of the EPROM holds the NIBL interpreter (Page 0) while the upper half holds the PAGE2 system (Page 2).

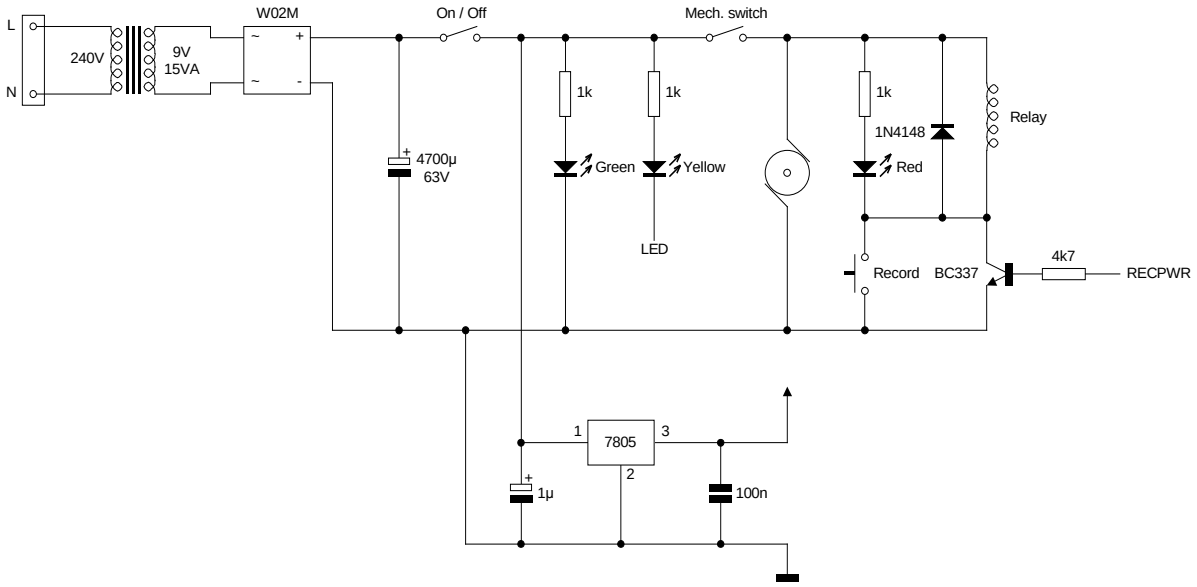
The computer has two serial ports: one for the console and another 'auxiliary' port for a printer or external device. Standard 5V logic levels are used for these ports since, in hobby application, serial leads rarely extend beyond the same desk. Accordingly, both serial ports are wired to non-standard connectors to prevent mating with devices using RS232 voltage levels. These connectors feature a pin carrying the 5V power rail, plus a master reset signal.

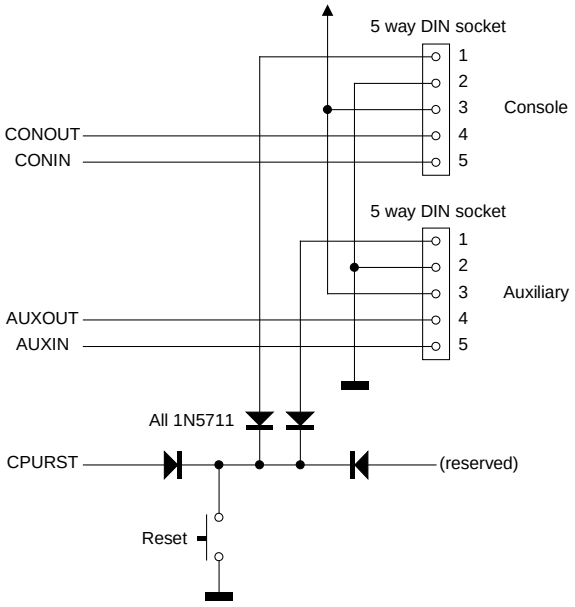
The SC/MP address space is divided as follows:

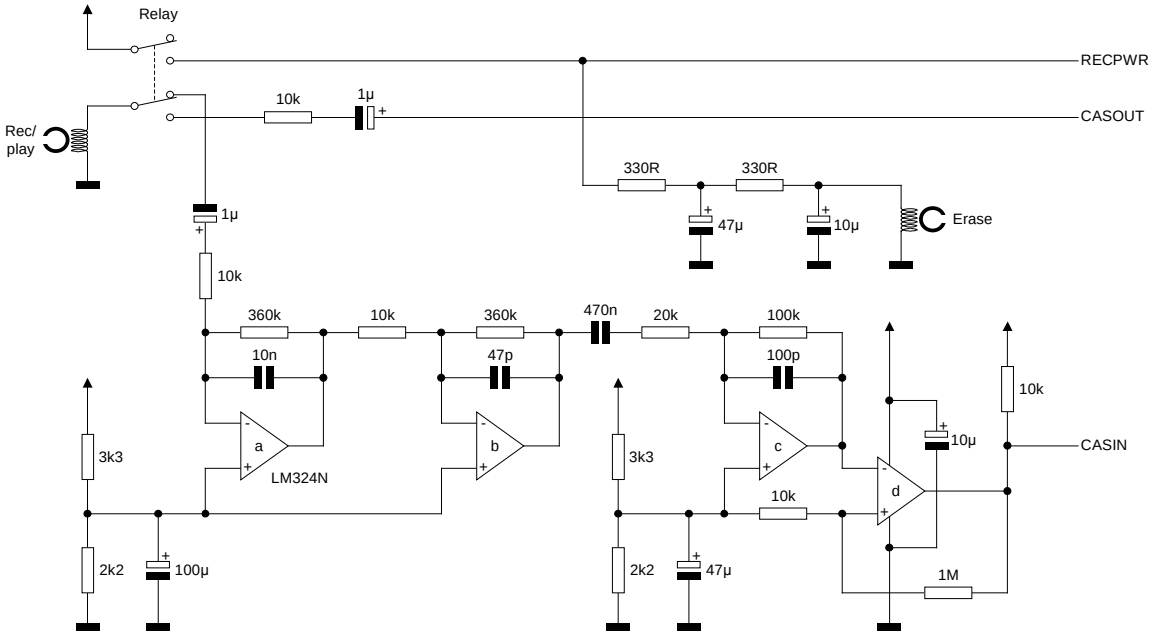
SC/MP	Device
0000 - 0FFF	EPROM
1000 - 1FFF	RAM
2000 - 2FFF	EPROM
3000 - 7FFF	RAM
8000 - 80FF	Expansion

No interrupts are supported by the computer.

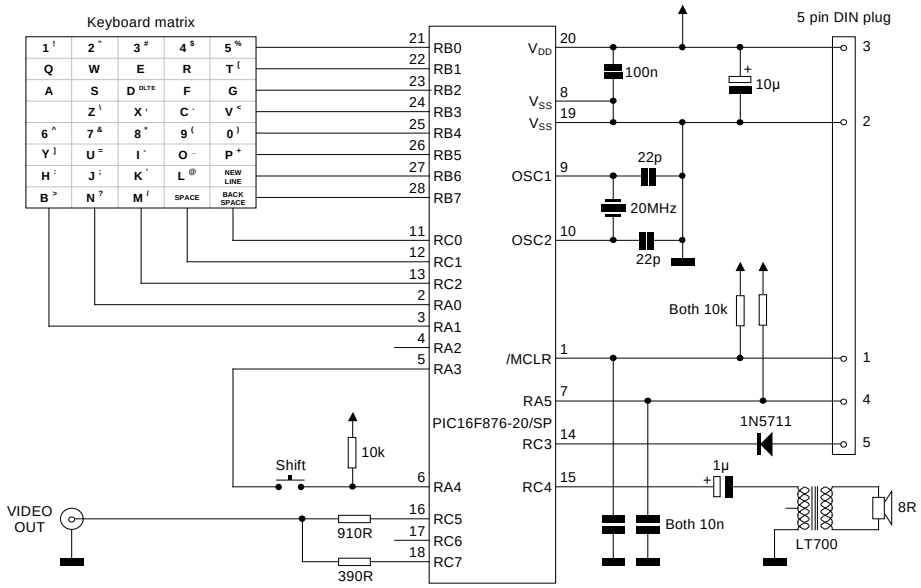

PIC16F877-20/P
RD6
RD5
RD4
RD3
RD2
RD1
RD0
RC7
RC6
RC5
RC4
RC3
RC2
RC1
RC0
RB7
RB6
RB5
RB4
RB3
RB2
RB1
RB0
Vcc
Vcc
Vss
Vss
/MCLR
RE0
RA5
RA4
RA3
RA2
RA1
RA0
OSC1
OSC2
RD7
RE2
RE1
(A14)
(A13)
(A12)
(A11)
(A10)
(A9)
(A8)
(A7)
(A6)
(A5)
(A4)
(A3)
(A2)
(A1)
(A0)
(D7)
(D6)
(D5)
(D4)
(D3)
(D2)
(D1)
(D0)
100n
22p
20MHz
22p
(SIN)
(SB)
(SA)
(SOUT)
(F2)
(F1)
(F0)
(A15)
(NWDS)
(NRDS)
All 10k
All 10k
All BC337
All 10n
CASIN
CASOUT
AUXOUT
LED
CONOUT
AUXIN
CONIN
CPURST
74HC139
2/Y3
2/Y2
2/Y1
2/Y0
1/Y3
1/Y2
1/Y1
1/Y0
2A1
2A0
2/E
1A1
1A0
1/E
GND
62256
IO7
IO6
IO5
IO4
IO3
IO2
IO1
IO0
/CS
/OE
/WE
2764
Vpp
/P
/E
/G
Q7
Q6
Q5
Q4
Q3
Q2
Q1
Q0
EXPANSION
+5V
+5V
NRDS
GND
NWDS
GND
/IOREQ
GND
GND


L
N
240V
9V
15VA
W02M
~
+
~
-
On / Off
Mech. switch
1k
1k
1k
1N4148
Relay
4700µ
63V
Green
Yellow
Red
LED
Record
BC337
4k7
RECPWR
1
7805
3
2
1µ
100n


5 way DIN socket
1
2
3
4
5
Console
CONOUT
CONIN
5 way DIN socket
1
2
3
4
5
Auxiliary
AUXOUT
AUXIN
All 1N5711
CPURST
(reserved)
Reset



A terminal was constructed for use with the SC/MP computer. This is based on a PIC microcontroller, and displays a 16x25 text display on a 625 line monitor. The terminal scans a 40 key keyboard. Most people will of course prefer to use a terminal emulation program running on a modern computer. The terminal communicates using 5V serial at 1200 Baud.



Chapter 6 Photo

Chapter 7 Software

A number of source files are needed to construct the SC/MP computer:

File	Purpose
NIBL1200.asm	The NIBL source code *
PAGE2.asm	The PAGE2 system source code
SCMPemu.asm	The PIC SC/MP emulator source code
Terminal.asm	The 1200 Baud terminal source code

* This source file is essentially the original NIBL source code, modified to use a 1200 Baud console. I would be seriously remiss if I did not acknowledge the valuable work undertaken by Roger Marin to scan the original NIBL source using OCR methods, and his painstaking correction of the scan to arrive at the exact original source code. I and others in the SC/MP enthusiast community are indebted to Roger for all his hard work.

I should also mention the assembler used to prepare the PAGE2 system. This is the remarkable 'AS' assembler created by Alfred Arnold and his team:

http://john.ccac.rwth-aachen.de:8000/as/

Chapter 8 Example program

The PAGE2 assembler was tested on a simple routine, which takes a string input from the user, and then sends it to the auxiliary (printer) port. The program was entered into page 3.

```
10 REM PRINT STRING
20 INPUT $TOP
30 LINK (TOP+80)
40 END
1000 ASM
1010 EQU  #2EEA    ; PUTAUX-1 IN PAGE 2
1020 SPC  80
1030 LDI  L$1010
1040 XPAL 3
1050 ST   0(2)
1060 LDI  H$1010
1070 XPAH 3
1080 ST   1(2)
1090 LDI  L$1020
1100 XPAL 1
1110 LDI  H$1020
1120 XPAH 1
1130 LD   0(1)
1140 XPPC 3
1150 LD   @1(1)
1160 XRI  13
1170 JNZ  $1130
1180 LD   0(2)
1190 XPAL 3
1200 LD   1(2)
1210 XPAH 3
1220 XPPC 3
```