DIS8051
Cross-Disassembler
User's Guide

Data Sync Engineering
P.O. Box 146
East Stroudsburg, PA 18301

TEL (717) 421-1977
FAX (717) 421-9095

DIS8051

USER NOTICE

Data Sync Engineering would like to thank all of those who have contributed towards the development of this software.

We have decided to place the full version of DIS8051 into the public domain. You may copy and use this software on the conditions that the program is not distributed in modified form, that no fee or consideration is charged, and that this notice is not bypassed or removed.

Any contributions for use of this software would be greatly appreciated.

Table of Contents

GENERAL

1.0 Distribution Disk Contents 1
1.1 General Description 1
1.2 System Requirements 1

USING THE DIS8051 DISASSEMBLER

2.0 Disassembler Output Format 2
2.1 Input File Types 3
2.2 Getting Started 4
2.3 The Output File 5
2.4 Tag File Description 6
2.5 Advanced DIS8051 Options 7
2.6 A Guided Tour 8
2.7 Disassembly Output Description 12
2.8 Altering the Disassembler Format 13
2.9 Cross Reference Lists 14
2.10 Operand Text Files 15
2.11 Text File Modification Rules 16
2.12 Error Messages 17

APPENDICES

3.0 INTEL HEX Format 18
3.1 8051/52 - Arithmetic Instructions 19
3.2 8051/52 - Logical Instructions 20
3.3 8051/52 - Data Transfer Instructions 21
3.4 8051/52 - Boolean Variable Instructions 22
3.5 8051/52 - Program Branching 23
3.6 Special Function Register Map 25
3.7 Bit Addressable Register Map 26
3.8 Special Function Register Assignments 27
3.9 Interrupt Vectors and Priority Levels 31
3.10 User RAM Memory Map 31

1.0 DISTRIBUTION DISK CONTENTS

DIS8051.COM - Disassembler program.
OPERAND.BIT - Mnemonic text for Bit Addressable SFR's.
OPERAND.BYT - Mnemonic text for Data Memory SFR's.
EXAMPLE.BIN - "Example" program assembled as a BINARY file.
EXAMPLE.HEX - "Example" program assembled as a HEX file.
EXAMPLE.LST - "Example" program listing.

1.1 GENERAL DESCRIPTION

The DIS8051 is a cross-disassembly tool. "Cross" meaning that while running on one type of a computer, it disassembles code for another. In many ways, the DIS8051 is similar to a debugger with the exception that it produces a source program code that is suitable for an assembler.

Features, such as, ASCII character display, Label generation, and cross-reference lists greatly aid in the complicated task of reconstructing or debugging a program.

A separate control file, called a TAG file, allows the user to mark (or tag) specific program areas to be disassembled as Text statements, HEX byte statements or simply ignored from disassembly.

External Operand mnemonic text files, enable the DIS8051 to be "tailored" or "adapted" to a wide variety of 8051 type Micro-controllers.

1.2 SYSTEM REQUIREMENTS

DIS8051 operates under MS-DOS 2.0 or greater and uses about 15K of memory. An additional 390K of memory is allocated for symbols and table storage. A typical system requirement would be about 512K bytes of memory.

DIS8051 produces a disassembled program with cross-references in three passes. Pass #1 reads the input file and tabulates symbols and references. Pass #2 rereads the input file and begins writing each line of program code. Pass #3 writes the sorted cross-reference lists for Labels, Data Memory, Bit Addressable Memory and Immediate Bytes.

An 8K Binary file, typically 23K in HEX format, may produce a disassembled source file of approximately 173K. While it is possible to run DIS8051 on a floppy disk, a hard disk would be recommended for speed considerations.

2.0 DISASSEMBLER OUTPUT FORMAT

All program code is written out using the conventional assembler format. This format divides each line in the program into four fields: label, op code, operand and comments.

The LABEL field is used to assign a symbolic name or label to the location of an instruction, so that it can be referenced by other instructions in the program. For example, the instruction LJMP L0100 will cause the program counter to be unconditionally loaded with the memory address 0100H which was assigned to the label L0100. The instruction at label L0100 will be the next instruction to be executed after the JUMP (LJMP) instruction is executed. Most instructions will not be labeled, however, if an instruction is referenced, the label will begin in the leftmost column of the line and will begin with the optional character (default = "L"). The body of the label will contain the target address value. The label is ended with the optional character (default = ":") and followed by a tab. If no label was assigned, the label field is skipped using a tab character.

The OP CODE field is mandatory for every line in the program that contains an instruction. The op code begins in field 2 and is separated from the label field by a tab character.

The OPERAND field is used to specify data or an address for instructions that require an operand. The operand begins in field 3 and is separated from the op code by a tab character. The DIS8051 provides three types of operand forms:

000H - 0FFH	Hexadecimal format.
L0000 - LFFFF	Label format.
ACC, DPL, SBUF	Mnemonic format.

The COMMENT field is used to add an explanatory note to a statement. The contents of the comment field are ignored by the assembler. The text of the comment field will be preceded by the optional character (default = semicolon ";"). Comments may be used alone without being appended to a line that contains an instruction. DIS8051 uses the comment field in one of three forms:

1. To identify the location and contents of the current instruction.

2. To deblock program segments for easier interpretation.

3. To append the cross-reference lists.

2.1 INPUT FILE TYPES

The DIS8051 can disassemble Hex or Binary coded files.

HEX — Hex files are translated versions of binary files. These translations are in a format of all ASCII characters. The entire file is broken down into groups of one-byte digits (which are ASCII characters). Each such group is given a separate load address (the place where it is to be loaded in memory) and a count (Number of characters in the group). In addition, there's a checksum in each group for detecting errors. HEX files can be transmitted from computer to computer or Downloaded to EPROM programmers.

If we take a look at an actual HEX file, we'll see something like this:

```
:03000000020100FA
:1001000090011812010880FEE493A3B4000122308C
:1001100099FDC299F59921084558414D504C45002B
:00000001FF
```

BINARY — Binary files are images of the program as it will appear in memory. The binary image cannot be easily transmitted or Downloaded because it consists of eight-bit bytes. Communication adapters usually use the eighth bit for Parity checking which leaves only seven bits for communication. Since every possible combination of eight bits can be part of a binary file, there would be no way to signal the end of a transmission.

Just as the eighth bit is used as a Parity check in communications, it is also used to enable graphic or special character symbols in video display adapters. Viewing a binary file on the screen would show a series of characters and symbols that won't make any sense. Programs that organize and display the contents of binary files are usually called DUMP or HEXDUMP programs.
The following is a view of a binary file using HEXDUMP:

```
0000  02 01 00 FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
0010  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
      :  :  :  :  :  :  :  :  :  :  :  :  :  :  :  :
0100  90 01 18 12 01 08 80 FE E4 93 A3 B4 00 01 22 30   ..............."0
0110  99 FD C2 99 F5 99 21 08 45 58 41 4D 50 4C 45 00   ......!.EXAMPLE.
```

2.2 GETTING STARTED

The HEX and BINARY data shown on the previous page, were from the example files supplied on this disk. The files are called EXAMPLE.HEX and EXAMPLE.BIN. The original listing, EXAMPLE.LST which is a text file, has been included for reference.

We will use these files within the descriptions of command and statement syntax.

****

To disassemble the EXAMPLE.HEX program in a very basic format, simply type:

```
DIS8051 EXAMPLE.HEX <RETURN>
```

The following messages will appear:

```
.....................................................................
.                                                                   .
.   DIS8051  Cross Disassembler    Version 2.x    PC/MS-DOS         .
.   Copyright (C) 1989 Data Sync Engineering   All rights reserved. .
.                                                                   .
.   Memory initialization ...                                       .
.   No Tag File                                                     .
.   Starting PASS Number 1  -  Processing: 0110                     .
.   Starting PASS Number 2  -  Processing: 0110                     .
.                                                                   .
.   Successful Disassembly -- Source File Created.                  .
.....................................................................
```

EXAMPLE.HEX has been disassembled and a new file called EXAMPLE.SRC was created. This new file contains the assembly language source code.

2.3 THE OUTPUT FILE

The source file created (EXAMPLE.SRC) will look like:

```
;=============================================================;
; Disassembled Using DIS8051   -  (C)1989 Data Sync Engineering ;
;=============================================================;
;
;
        ORG     00000H
;
        LJMP    L0100
;
        ORG     00100H
;
;
L0100:  MOV     DPTR,#L0118
        LCALL   L0108
;
L0106:  SJMP    L0106
;
L0108:  CLR     A
        MOVC    A,@A+DPTR
        INC     DPTR
        CJNE    A,#000H,L010F
        RET
;
L010F:  JNB     TI,L010F
        CLR     TI
        MOV     SBUF,A
        AJMP    L0108
;
L0118:  ORL     A,058H
        AJMP    L024D
;
        JNC     L016A
        ORL     A,000H
;
;       Unresolved Address Reference list
;
;
L016A:          EQU        0016AH
L024D:          EQU        0024DH
;
        END
```

As you can see, DIS8051 automatically inserted Labels for memory location references and Directive statements for Assembler control. A code segmentation method is also performed by inserting blank comment lines after certain instructions, such as RET and JMP, and before a line that contains a Label.

Since PASS 1 assigned label operands to the memory referenced instructions, PASS 2 inserts these labels at their appropriate addresses. If a memory location reference was not found as an instruction address then DIS8051 automatically EQUATES the value to that label and shows it in the Unresolved Reference list.

Unresolved References are generally caused by the disassembly of ASCII text characters, Program Tables or by accesses made to undefined I/O devices or other external memory. A separate file, called a Tag file, can be used to further control the disassembly process by marking these areas of the program for a specified mode of disassembly.

2.4 TAG FILE DESCRIPTION

The Tag file tells the DIS8051 what mode of disassembly to switch to. This file can be created with any ASCII text editor and is identified with the same filename but has the file extension of ".TAG". Entries into the tag file consist of a four character hexadecimal start address followed by the "=" character then the command character. There are four basic disassembly modes; Instruction disassembly, Skip or ignore byte disassembly, Hex Byte table disassembly and ASCII Text disassembly (Define Byte statements containing ASCII text characters). An additional command called "Generate Label equate", forces DIS8051 to build an equate list defining the I/O and External Memory addresses. This "Equate List" is written at the beginning of the disassembled file and is not recorded within the Unresolved Address References.

Tag File Command Summary:

Function Character	Result
G or g	GENERATE Label Equate for specified address.
S or s	SKIP disassembly (ignores marked bytes).
I or i	INSTRUCTION disassembly.
B or b	Define BYTE disassembly (DB 000H).
T or t	TEXT disassembly (DB 'text').

* See ALTERING DISASSEMBLER FORMAT for more Tag File features.

2.5 ADVANCED DIS8051 OPTIONS

The DIS8051 contains powerful disassembly options that provide a more complete detail on the program that is being disassembled.

Advanced Option Summary:

Option letter	Result
B or b	Tells DIS8051 to input a Binary coded file.
C or c	Append comment field Hexdump to each instruction line.
L or l	Converts output assembly code to lower case characters.
R or r	Append Cross-reference lists to end of source file.
T or t	Include Tag File disassembly parameters.
X or x	Write Cross-Reference lists only, no assembly code.

Option letters must be preceded by a slash "/" and may be grouped together (e.g. DIS8051 EXAMPLE.HEX /LCTR).

COMMENT FIELD HEXDUMP

Comment fields are appended to each line containing a disassembled instruction. They show the Program Counters value, the HEX bytes utilized by the current instruction and the displayable ASCII character equivalent of the Hex bytes. Undisplayable characters, such as control characters, are shown as periods ".".

CROSS-REFERENCE LISTS

The Cross-reference lists produced by the DIS8051, map out all Program Code and Memory usages. The listing shows the Label or Memory address followed by all locations that refer to that address. The referencing locations are identified by the Program Counter value shown in the comment field hexdump.

2.6 A GUIDED TOUR

STEP 1 - Disassemble the EXAMPLE.HEX program with hexdump and cross-reference lists. Type:

```
DIS8051 EXAMPLE.HEX /RC <RETURN>
```

The following messages will appear:

```
.....................................................................
.                                                                   .
.   DIS8051  Cross Disassembler    Version 2.x    PC/MS-DOS         .
.   Copyright (C) 1989 Data Sync Engineering   All rights reserved.  .
.                                                                   .
.   Memory initialization ...                                       .
.   No Tag File                                                     .
.   Starting PASS Number 1  -  Processing: 0110                     .
.   Starting PASS Number 2  -  Processing: 0110                     .
.   Starting PASS Number 3  -  Xref lists: 0000                     .
.                                                                   .
.   Successful Disassembly -- Source File Created.                  .
.....................................................................
```

STEP 2 - View the disassembled listing (EXAMPLE.SRC).

```
;=================================================================;
; Disassembled Using DIS8051   -  (C)1989 Data Sync Engineering ;
;=================================================================;
;
;
        ORG     00000H
;
        LJMP    L0100             ;0000  02 01 00        ...
;
;
        ORG     00100H
;
;
L0100:  MOV     DPTR,#L0118       ;0100  90 01 18        ...
        LCALL   L0108             ;0103  12 01 08        ...
;
L0106:  SJMP    L0106             ;0106  80 FE           ..
```

A GUIDED TOUR (cont.)

```
L0108:  CLR     A                   ;0108  E4              .
        MOVC    A,@A+DPTR           ;0109  93              .
        INC     DPTR                ;010A  A3              .
        CJNE    A,#000H,L010F       ;010B  B4 00 01        ...
        RET                         ;010E  22              "
;
L010F:  JNB     TI,L010F            ;010F  30 99 FD        0..
        CLR     TI                  ;0112  C2 99           ..
        MOV     SBUF,A              ;0114  F5 99           ..
        AJMP    L0108               ;0116  21 08           !.
;
L0118:  ORL     A,058H              ;0118  45 58           EX
        AJMP    L024D               ;011A  41 4D           AM
;
        JNC     L016A               ;011C  50 4C           PL
        ORL     A,000H              ;011E  45 00           E.
;
;       Unresolved Address Reference list
;
;
L016A:          EQU     0016AH
L024D:          EQU     0024DH
;
;       Cross-references to LABELS
;
; L0100=  0000
; L0106=  0106
; L0108=  0103  0116
; L010F=  010B  010F
; L0118=  0100
; L016A=  011C
; L024D=  011A
;
;       Cross-references to Data Memory locations
;
; M: 00=  011E
; M: 58=  0118
; M: 99=  0114
;
;       Cross-references to BIT addressable locations
;
; B: 99=  010F  0112
;
;       Immediate Byte references
;
; #: 00=  010B
;
        END
```

A GUIDED TOUR (cont.)

When looking at the listing, notice that the area between locations 0118 to 011E contain ASCII text characters. It is also the same area that is generating the Unresolved Address References.

Hint: A complete disassembly should not produce any Unresolved Address References.

STEP 3 - Using your text editor, create a file called EXAMPLE.TAG. In this file, write the following lines:

```
0118=T
011F=B
FFFF
```

The above Tag File commands specify that; starting at location 0118, switch to TEXT mode disassembly. Then at location 011F switch to Define Byte disassembly. The FFFF is used to signal the end of the Tag file command list.

Each line of the Tag file contains only one command, but there is no limit to its length except that the "FFFF" address designation must be used to signal the end.

STEP 4 - Now disassemble the EXAMPLE.HEX program again using the Tag File parameters, lower-case character option and hexdump feature. Type:

```
DIS8051 EXAMPLE.HEX /TCL <RETURN>
```

```
.....................................................................
.                                                                   .
.   DIS8051  Cross Disassembler    Version 2.x    PC/MS-DOS         .
.   Copyright (C) 1989 Data Sync Engineering   All rights reserved.  .
.                                                                   .
.   Memory initialization ...                                       .
.   Tag File Processed                                              .
.   Starting PASS Number 1  -  Processing: 0110                     .
.   Starting PASS Number 2  -  Processing: 0110                     .
.                                                                   .
.   Successful Disassembly -- Source File Created.                  .
.....................................................................
```

STEP 5 - By the way, we used the lower-case option to satisfy the programmers who prefer that listing method. The final EXAMPLE.SRC result should now look like:

```
;===============================================================;
; Disassembled Using DIS8051   -  (C)1989 Data Sync Engineering ;
;===============================================================;
;
;
;
        org     00000h
;
        ljmp    L0100               ;0000  02 01 00        ...
;
;
        org     00100h
;
;
L0100:  mov     dptr,#L0118         ;0100  90 01 18        ...
        lcall   L0108               ;0103  12 01 08        ...
;
L0106:  sjmp    L0106               ;0106  80 FE           ..
;
L0108:  clr     a                   ;0108  E4              .
        movc    a,@a+dptr           ;0109  93              .
        inc     dptr                ;010A  A3              .
        cjne    a,#000h,L010F       ;010B  B4 00 01        ...
        ret                         ;010E  22              "
;
;
L010F:  jnb     ti,L010F            ;010F  30 99 FD        0..
        clr     ti                  ;0112  C2 99           ..
        mov     sbuf,a              ;0114  F5 99           ..
        ajmp    L0108               ;0116  21 08           !.
;
;
L0118:  db      'EXAMPLE'
        db      000h                ;011F  00              .
;
;        Unresolved Address Reference list
;
;
;
;
        end
```

2.7 DISASSEMBLY OUTPUT DESCRIPTION

The ORG assembly directives were inserted from information contained in the HEX file.

```
        ORG     00000H
;
        LJMP    L0100               ;0000  02 01 00        ...
;
        ORG     00100H


:03 0000 00 020100 FA
:10 0100 00 90011812010880FEE493A3B400012230 8C
    ----
      |
      +---> HEX file Load Address information




L0100:  MOV     DPTR,#L0118         ;0100  90 01 18        ...
        LCALL   L0108               ;0103  12 01 08        ...
;
L0106:  SJMP    L0106               ;0106  80 FE           ..
;
L0108:  CLR     A                   ;0108  E4              .
        MOVC    A,@A+DPTR           ;0109  93              .
        INC     DPTR                ;010A  A3              .
        CJNE    A,#000H,L010F       ;010B  B4 00 01        ...
        RET                         ;010E  22              "
;
L010F:  JNB     TI,L010F            ;010F  30 99 FD        0..
        CLR     TI                  ;0112  C2 99           ..
        MOV     SBUF,A              ;0114  F5 99           ..
        AJMP    L0108               ;0116  21 08           !.
;
L0118:  ORL     A,058H              ;0118  45 58           EX
        AJMP    L024D               ;011A  41 4D           AM
;
        JNC     L016A               ;011C  50 4C           PL
        ORL     A,000H              ;011E  45 00           E.
                                    -----  --------        ---
                                       |      |             |
      Program Counter  <---------------+      |             |
                                              |             |
               HEX bytes utilized  <----------+             |
                                                            |
                     ASCII equivalent of HEX bytes  <-------+
```

2.8 ALTERING THE DISASSEMBLER FORMAT

Because of variations in Assembler Formats and Directive statement syntax, it may become necessary to alter certain directive and delimiter sequences. The Tag File can be used to change the default setting of these formats. Listed below are the output formats and their default settings:

Format Type	Default Setting	Tag Function	Max Size
Origin Directive	ORG	O	4
Equate Directive	EQU	E	4
Define Directive for HEX Bytes	DB	D	4
Define Directive for ASCII Text	DB	A	4
Start-of-Label character	L	L	1
End-of-Equate-Name delimiter char	:	W	1
End-of-Label-Name delimiter char	:	X	1
Text delimiter characters	'	Y	1
Comment Field delimiter character	;	Z	1
Output File Extension	SRC	F	3
Mask 7-bits for ASCII hexdump	8-bit	M	-
Alternate HEX Notation	0--H	$	-

Valid examples of format changes:

Tag Command line	Change from:	To:
0000=O.ORG	ORG	.ORG
0000=EEQ	EQU	EQ
0000=DDFB	DB	DFB
0000=ATEXT	DB	TEXT
0000=LZ	L	Z
0000=X	X	(no delimiter)
0000=Y"	'	"
0000=Z*	;	*
0000=FTST	SRC	TST
0000=$	0--H	$--

2.9 CROSS REFERENCE LISTS

```
;       Cross-references to LABELS
;
; L0100=  0000
; L0106=  0106
; L0108=  0103  0116
; L010F=  010B  010F
; L0118=  0100
; L016A=  011C
; L024D=  011A
  -----   ----
    |      |
    |      +-------> Program address producing the reference.
    |
    +----------> Referenced Label (instruction address).


;       Cross-references to Data Memory locations
;
; M: 99=  0114
  -- --   ----
  |  |     |
  |  |     +--------> Program address producing the reference.
  |  |
  |  +---------> Data Memory Byte Address  (99 = address for SBUF).
  |
  +------> Indicates address is for Data Memory or byte memory.


;       Cross-references to BIT addressable locations
;
; B: 99=  010F  0112
  -- --   ----  ----
  |  |     |     |
  |  |     +-----+--> Program addresses producing the reference.
  |  |
  |  +---------> Bit Addressable Memory  (99 = bit address for TI).
  |
  +------> Indicates address is for a BIT Addressable memory.


;       Immediate Byte references
;
; #: 00=  010B
  -- --   ----
  |  |     |
  |  |     +--------> Program addresses producing the reference.
  |  |
  |  +---------> Immediate Values  (usually called CONSTANTS).
  |
  +------> Indicates immediate byte value.
```

2.10 OPERAND TEXT FILES

When a disassembly process is started, the DIS8051 loads in two text files, OPERAND.BYT and OPERAND.BIT. These files contain the text substitution for the BYTE and BIT address mnemonics. They are used to adapt the DIS8051 disassembler to other members of the 8051 family.

The rules, on the following page, should be followed when modifying the OPERAND.BYT and OPERAND.BIT files.

Entries within the files begin with Byte or Bit address "00" and end with address "FF". Each line contains a total of ten characters including the carriage-return and line-feed. An operand mnemonic is terminated by the "=" character. Any text after this character is ignored.

The following examples show the difference between the 8052 and 80C152 SFR register assignment:

Special Function Register BYTE address (OPERAND.BYT)

Byte Address	8052	80C152
C8	T2CON===	IEN1====

Special Function Register BIT addresses (OPERAND.BIT)

Bit Address	8052	80C152
C8	T2CP====	EGSRV===
C9	T2C=====	EGSRE===
CA	TR2=====	EDMA0===
CB	EXEN2===	EGSTV===
CC	TLCK====	EDMA1===
CD	RCLK====	EGSTE===
CE	EXF2====	0CEH====
CF	TF2=====	0CFH====

2.11 TEXT FILE MODIFICATION RULES

1. All text should be entered in upper case characters.

2. The "=" is used to end the mnemonic text word.

3. Only up to a seven character mnemonic can be used.

4. Each line must contain eight characters plus CR and LF. Any fill character can be used as long as the "=" character precedes it.

5. All characters after the "=" will not be inserted into the assembly output file.

6. Non applicable mnemonics, such as BIT addresses CE & CF of the 80C152, must contain at least one printable character. Hex notations (0--H) are generally recommended.

7. Although, address locations 00 to 7F are not part of the SFR register range, they are set to the hex notations 000H to 07FH. Symbol names can be used instead.

2.12 ERROR MESSAGES

Error -- No Input File Specified

Filename was missing in the command line.
Command Line syntax: DIS8051 <filename>[.ext] [/options]

Error -- Input File Did Not Open

Input File was not found.
Check Drive, Path or Filename.

Error -- Input File Empty

No data found in input file.
Check File contents.

Error -- Insufficient Disk Space

Disk or Directory Full.
Insert a new disk or delete unused files.

Load Error In HEX File

Checksum Error in HEX File.
Try another HEX file.

Error When Loading Operand Text Files

Either wrong file length, Empty File, or File Not Found.
Check OPERAND.BIT and OPERAND.BYT files.

A large amount of memory has been allocated for Cross-reference tables. If in the event, which is unlikely, an overflow has occured, a warning message will be displayed and that particular reference list will be truncated at it's maximum capacity of 32,766 references.

3.0 INTEL HEX FORMAT

DATA RECORD ---

BYTE #	1	Colon (:), signifies start of a record.
	2 & 3	Number of data bytes in this record.
	4 & 5	Load address for this record, High Byte.
	6 & 7	Load address for this record, Low Byte.
	8 & 9	Record type, must be "00".
	10 to X	Data bytes, two ASCII hex characters each.
	X+1 & X+2	Checksum, two ASCII hex characters.
	X+3 & X+4	CR & LF, (carriage return & line-feed).

END RECORD ---

BYTE #	1	Colon (:), signifies start of a record.
	2 & 3	Record length, must be "00".
	4 to 7	Start address, "0000" = end record.
	8 & 9	Record type
	10 & 11	Checksum, two ASCII hex characters.
	12 & 13	CR & LF, (carriage return & line-feed).

The CHECKSUM is the two's complement of the 8-bit sum of the Record Length, the two byte Load Address, the Record Type, and all the Data bytes.

3.1 8051 INSTRUCTION SET - ARITHMETIC OPERATIONS

Assembly form	Byte/Cycle	Flags,Notes	Description
ADD A,Rn	1/1	AC,CY,OV	Add Register to Accumulator
ADD A,direct	2/1	AC,CY,OV	Add Direct Byte to Accumulator
ADD A,@Ri	1/1	AC,CY,OV	Add Indirect RAM to Accumulator
ADD A,#data	2/1	AC,CY,OV	Add Immediate Data to Accumulator
ADDC A,Rn	1/1	AC,CY,OV	Add Register to Acc with Carry
ADDC A,direct	2/1	AC,CY,OV	Add Direct Byte to Acc with Carry
ADDC A,@Ri	1/1	AC,CY,OV	Add Indirect RAM to Acc with Carry
ADDC A,#data	2/1	AC,CY,OV	Add Immediate Data to Acc with Carry
DA A	1/1	CY,4	Decimal Adjust Accumulator
DEC A	1/1		Decrement Accumulator
DEC Rn	1/1		Decrement Register
DEC direct	2/1		Decrement Direct Byte
DEC @Ri	1/1		Decrement Indirect RAM
DIV AB	1/4	CY=0,OV,5	Divide Accumulator by B Register
INC A	1/1		Increment Accumulator
INC Rn	1/1		Increment Register
INC direct	2/1		Increment Direct Byte
INC @Ri	1/1		Increment Indirect RAM
INC DPTR	1/2		Increment Data Pointer
MUL AB	1/4	CY=0,OV,7	Multiply Accumulator and B Register
SUBB A,Rn	1/1	AC,CY,OV	Subtract Register from Acc w/borrow
SUBB A,direct	2/1	AC,CY,OV	Sub Direct Byte from Acc w/borrow
SUBB A,@Ri	1/1	AC,CY,OV	Sub Indirect RAM from Acc w/borrow
SUBB A,#data	2/1	AC,CY,OV	Sub Immediate Data from Acc w/borrow

3.2 8051 INSTRUCTION SET - LOGICAL OPERATIONS

Assembly form	Byte/Cycle	Flags,Notes	Description
ANL A,Rn	1/1		AND Register to Accumulator
ANL A,direct	2/1		AND Direct Byte to Accumulator
ANL A,@Ri	1/1		AND Indirect RAM to Accumulator
ANL A,#data	2/1		AND Immediate Data to Accumulator
ANL direct,A	2/1		AND Accumulator to Direct Byte
ANL direct,#data	3/2		AND Immediate Data to Direct Byte
CLR A	1/1		Clear Accumulator
CPL A	1/1		Complement Accumulator
ORL A,Rn	1/1		OR Register to Accumulator
ORL A,direct	2/1		OR Direct Byte to Accumulator
ORL A,@Ri	1/1		OR Indirect RAM to Accumulator
ORL A,#data	2/1		OR Immediate Data to Accumulator
ORL direct,A	2/1		OR Accumulator to Direct Byte
ORL direct,#data	3/2		OR Immediate Data to Direct Byte
RL A	1/1		Rotate Accumulator Left
RLC A	1/1	CY	Rotate Acc Left through Carry
RR A	1/1		Rotate Accumulator Right
RRC A	1/1	CY	Rotate Acc Right through Carry
SWAP A	1/1		Swap Nibbles within the Accumulator
XRL A,Rn	1/1		Exclusive-OR Register to Accumulator
XRL A,direct	2/1		Exclusive-OR Direct Byte to Acc
XRL A,@Ri	1/1		Exclusive-OR Indirect RAM to Acc
XRL A,#data	2/1		Exclusive-OR Immediate Data to Acc
XRL direct,A	2/1		Exclusive-OR Acc to Direct Byte
XRL direct,#data	3/2		Exclusive-OR Imm Data to Direct Byte

3.3 8051 INSTRUCTION SET - DATA TRANSFER

Assembly form	Byte/Cycle Flags,Notes	Description
MOV A,Rn	1/1	Move Register to Accumulator
MOV A,direct	2/1	Move Direct Byte to Accumulator
MOV A,@Ri	1/1	Move Indirect RAM to Accumulator
MOV A,#data	2/1	Move Immediate Data to Accumulator
MOV Rn,A	1/1	Move Accumulator to Register
MOV Rn,direct	2/2	Move Direct Byte to Register
MOV Rn,#data	2/1	Move Immediate Data to Register
MOV direct,A	2/1	Move Accumulator to Direct Byte
MOV direct,Rn	2/2	Move Register to Direct Byte
MOV direct,direct	3/2	Move from Direct Byte to Direct Byte
MOV direct,@Ri	2/2	Move Indirect RAM to Direct Byte
MOV direct,#data	3/2	Move Immediate Data to Direct Byte
MOV @Ri,A	1/1	Move Accumulator to Indirect RAM
MOV @Ri,direct	2/2	Move Direct Byte to Indirect RAM
MOV @Ri,#data	2/1	Move Immediate Data to Indirect RAM
MOV DPTR,#data16	3/2	Load Data Pointer w/ 16-bit Constant
MOVC A,@A+DPTR	1/2	Move Code Byte relative DPTR to Acc
MOVC A,@A+PC	1/2	Move Code Byte relative to PC to Acc
POP direct	2/2	Pop Direct Byte from Stack
PUSH direct	2/2	Push Direct Byte onto Stack
XCH A,Rn	1/1	Exchange Register with Accumulator
XCH A,direct	2/1	Exchange Direct Byte with Accumulato r
XCH A,@Ri	1/1	Exchange Indirect RAM with Acc
XCHD A,@Ri	1/1	Exchange Low Nibble Ind RAM with Acc

3.4 8051 INSTRUCTION SET - BOOLEAN VARIABLE MANIPULATION

Assembly form	Byte/Cycle	Flags,Notes	Description
ANL C,bit	2/2	CY	AND Direct Bit to Carry
ANL C,/bit	2/2	CY	AND complement of Direct Bit to Cy
CLR C	1/1	CY	Clear Carry
CLR bit	2/1		Clear Direct Bit
CPL C	1/1	CY	Complement Carry
CPL bit	2/1		Complement Direct Bit
JB bit,rel	3/2		Jump if Direct Bit is set
JNB bit,rel	3/2		Jump if Direct Bit is Not set
JBC bit,rel	3/2	AC,CY,OV,6	Jump if Direct Bit set then Clear it
JC rel	2/2		Jump if Carry is set
JNC rel	2/2		Jump if Carry is Not set
MOV C,bit	2/1	CY	Move Direct Bit to Carry
MOV bit,C	2/2		Move Carry to Direct Bit
ORL C,bit	2/2	CY	OR Direct Bit to Carry
ORL C,/bit	2/2	CY	OR complement of Direct Bit to Carry
SETB C	1/1	CY	Set Carry
SETB bit	2/1		Set Direct Bit

3.5 8051 INSTRUCTION SET - PROGRAM BRANCHING

Assembly form	Byte/Cycle	Flags,Notes	Description
ACALL addr 11	2/2	1	Absolute Subroutine Call, 11-bit address
AJMP addr 11	2/2	2	Absolute Jump, 11-bit address
CJNE A,direct,rel	3/2	CY,3	Comp Dir Byte to Acc, Jmp not equal
CJNE A,#data,rel	3/2	CY,3	Compare Immed to Acc & Jump not equal
CJNE Rn,#data,rel	3/2	CY,3	Compare Immed to Reg & Jump not equal
CJNE @Ri,#data,rel	3/2	CY,3	Comp Imm to Indir Ri, Jump not equal
DJNZ Rn,rel	2/2		Decrement Register & Jump if not zero
DJNZ direct,rel	3/2		Decrement Direct Byte & Jump not zero
JMP @A+DPTR	1/2		Jump indirect relative to the DPTR
JNZ rel	2/2		Jump if Accumulator is Not Zero
JZ rel	2/2		Jump if Accumulator is Zero
LCALL addr 16	3/2		Long Subroutine Call, 16-bit Address
LJMP addr 16	3/2		Long Jump, 16-bit Address
NOP	1/1		No Operation
RET	1/2		Return from subroutine
RETI	1/2		Return from Interrupt
SJMP rel	2/2		Short Jump, relative address

NOTES

1. Starting with 11H as the opcode base, the final opcode is formed by placing bits 8, 9, and 10 of the target address in bits 5, 6, and 7 of the opcode. The 8 possible opcodes in hexadecimal are then:

   11, 31, 51, 71, 91, B1, D1, F1

2. Starting with 01H as the opcode base, the final opcode is formed by placing bits 8, 9, and 10 of the target address in bits 5, 6, and 7 of the opcode. The 8 possible opcodes in hexadecimal are then:

   01, 21, 41, 61, 81, A1, C1, E1

3. The Carry Flag is set if the Destination Operand is less than the Source Operand, otherwise the Carry Flag is clear.

4. The Carry Flag is set if the BCD result in the Accumulator is greater than decimal 99.

5. The Overflow Flag is set if the B Register contains zero (flags a divide by zero operation). Otherwise the Overflow Flag is cleared.

6. If any of the condition code flags are specified as the operand of this instruction, they will be reset by the instruction if they were originally set.

7. The high byte of the 16-bit product is placed in the B Register, the low byte in the Accumulator.

NOTES ON THE INSTRUCTION SET AND THE ADDRESSING MODES

addr 11 - 11-bit destination address. Used by ACALL and AJMP. Branch will be within the same 2K-byte page of Program Memory as the first byte of the following instruction.
addr 16 - 16-bit destination address. Used by LCALL and LJMP. A branch can be anywhere within the 64K-byte Program Memory address space.
bit - Direct Addressed bit in Internal Data RAM or Special Function Register.
direct - 8-bit internal data location's address. This could be an Internal Data RAM location (0-127) or an SFR [i.e., I/O port, control register, status register, ect.(128-255)].
rel - Signed (two's complement) 8-bit offset byte. Used by SJMP and all conditional jumps. Range is -128 to +127 bytes relative to first byte of the following instruction.
Rn - Register R0-R7 of the currently selected Register Bank.
@Ri - 8-bit internal data RAM location (0-255) addressed indirectly through register R0 or R1.
#data - 8-bit constant included in instruction.
#data 16 - 16-bit constant included in instruction.

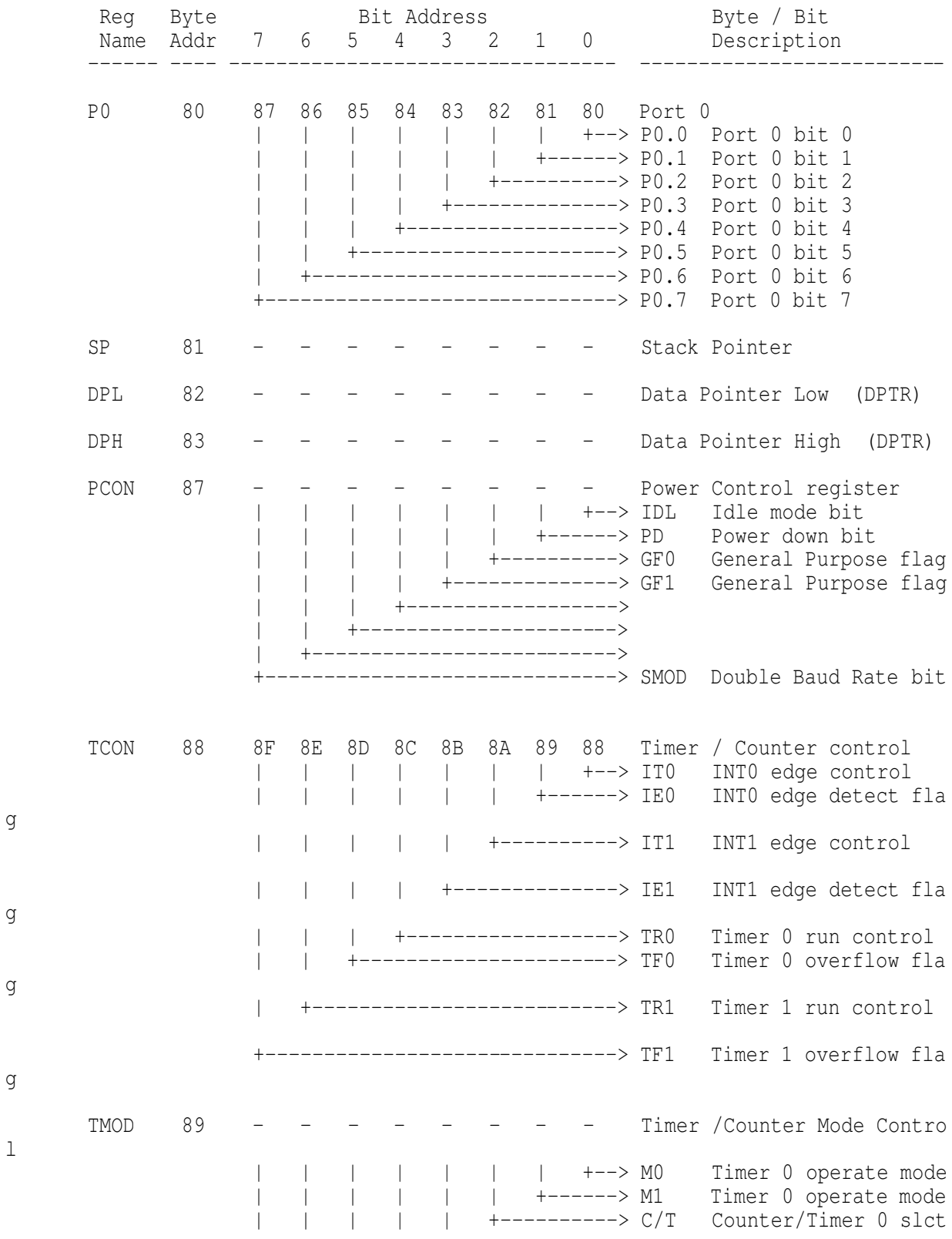
3.6 SPECIAL FUNCTION REGISTER MAP

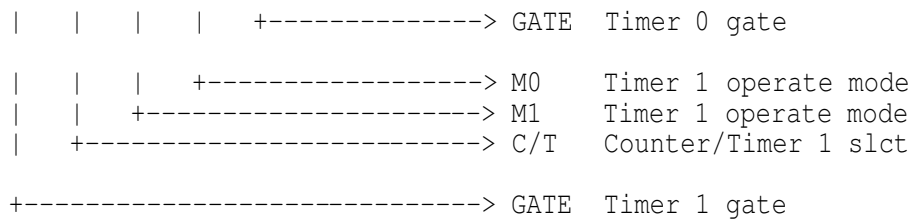
```
M: 80= P0       M: A0= P2       M: C0=          M: E0= ACC
M: 81= SP       M: A1=          M: C1=          M: E1=
M: 82= DPL      M: A2=          M: C2=          M: E2=
M: 83= DPH      M: A3=          M: C3=          M: E3=
M: 84=          M: A4=          M: C4=          M: E4=
M: 85=          M: A5=          M: C5=          M: E5=
M: 86=          M: A6=          M: C6=          M: E6=
M: 87= PCON     M: A7=          M: C7=          M: E7=

M: 88= TCON     M: A8= IE       M: C8= T2CON    M: E8=
M: 89= TMOD     M: A9=          M: C9=          M: E9=
M: 8A= TL0      M: AA=          M: CA= RCAP2L   M: EA=
M: 8B= TL1      M: AB=          M: CB= RCAP2H   M: EB=
M: 8C= TH0      M: AC=          M: CC= TL2      M: EC=
M: 8D= TH1      M: AD=          M: CD= TH2      M: ED=
M: 8E=          M: AE=          M: CE=          M: EE=
M: 8F=          M: AF=          M: CF=          M: EF=

M: 90= P1       M: B0= P3       M: D0= PSW      M: F0= B
M: 91=          M: B1=          M: D1=          M: F1=
M: 92=          M: B2=          M: D2=          M: F2=
M: 93=          M: B3=          M: D3=          M: F3=
M: 94=          M: B4=          M: D4=          M: F4=
M: 95=          M: B5=          M: D5=          M: F5=
M: 96=          M: B6=          M: D6=          M: F6=
M: 97=          M: B7=          M: D7=          M: F7=

M: 98= SCON     M: B8= IP       M: D8=          M: F8=
M: 99= SBUF     M: B9=          M: D9=          M: F9=
M: 9A=          M: BA=          M: DA=          M: FA=
M: 9B=          M: BB=          M: DB=          M: FB=
M: 9C=          M: BC=          M: DC=          M: FC=
M: 9D=          M: BD=          M: DD=          M: FD=
M: 9E=          M: BE=          M: DE=          M: FE=

M: 9F=          M: BF=          M: DF=          M: FF=
```

3.7 BIT ADDRESSABLE REGISTER MAP

```
B: 80= P0.0      B: A0= P2.0      B: C0=           B: E0= ACC.0
B: 81= P0.1      B: A1= P2.1      B: C1=           B: E1= ACC.1
B: 82= P0.2      B: A2= P2.2      B: C2=           B: E2= ACC.2
B: 83= P0.3      B: A3= P2.3      B: C3=           B: E3= ACC.3
B: 84= P0.4      B: A4= P2.4      B: C4=           B: E4= ACC.4
B: 85= P0.5      B: A5= P2.5      B: C5=           B: E5= ACC.5
B: 86= P0.6      B: A6= P2.6      B: C6=           B: E6= ACC.6
B: 87= P0.7      B: A7= P2.7      B: C7=           B: E7= ACC.7

B: 88= IT0       B: A8= EX0       B: C8= T2CP      B: E8=
B: 89= IE0       B: A9= ET0       B: C9= T2C       B: E9=
B: 8A= IT1       B: AA= EX1       B: CA= TR2       B: EA=
B: 8B= IE1       B: AB= ET1       B: CB= EXEN2     B: EB=
B: 8C= TR0       B: AC= ES        B: CC= TCLK      B: EC=
B: 8D= TF0       B: AD= ET2       B: CD= RCLK      B: ED=
B: 8E= TR1       B: AE=           B: CE= EXF2      B: EE=
B: 8F= TF1       B: AF= EA        B: CF= TF2       B: EF=

B: 90= P1.0      B: B0= P3.0      B: D0= P         B: F0= B.0
B: 91= P1.1      B: B1= P3.1      B: D1=           B: F1= B.1
B: 92= P1.2      B: B2= P3.2      B: D2= OV        B: F2= B.2
B: 93= P1.3      B: B3= P3.3      B: D3= RS0       B: F3= B.3
B: 94= P1.4      B: B4= P3.4      B: D4= RS1       B: F4= B.4
B: 95= P1.5      B: B5= P3.5      B: D5= F0        B: F5= B.5
B: 96= P1.6      B: B6= P3.6      B: D6= AC        B: F6= B.6
B: 97= P1.7      B: B7= P3.7      B: D7= CY        B: F7= B.7

B: 98= RI        B: B8= PX0       B: D8=           B: F8=
B: 99= TI        B: B9= PT0       B: D9=           B: F9=
B: 9A= RB8       B: BA= PX1       B: DA=           B: FA=
B: 9B= TB8       B: BB= PT1       B: DB=           B: FB=
B: 9C= REN       B: BC= PS        B: DC=           B: FC=
B: 9D= SM2       B: BD= PT2       B: DD=           B: FD=
B: 9E= SM1       B: BE=           B: DE=           B: FE=

B: 9F= SM0       B: BF=           B: DF=           B: FF=
```

3.8 SPECIAL FUNCTION REGISTER ASSIGNMENTS




GATE Timer 0 gate
M0 Timer 1 operate mode
M1 Timer 1 operate mode
C/T Counter/Timer 1 slct
GATE Timer 1 gate

SPECIAL FUNCTION REGISTER ASSIGNMENTS (cont.)

Reg Name	Byte Addr	Bit Address 7	6	5	4	3	2	1	0	Byte / Bit Description
TL0	8A	-	-	-	-	-	-	-	-	Timer / Counter 0 low byte
TL1	8B	-	-	-	-	-	-	-	-	Timer / Counter 1 low byte
TH0	8C	-	-	-	-	-	-	-	-	Timer / Counter 0 high byte
TH1	8D	-	-	-	-	-	-	-	-	Timer / Counter 1 high byte
P1	90	97	96	95	94	93	92	91	90	Port 1
								+-->	P1.0	Port 1 bit 0
							+------>		P1.1	Port 1 bit 1
						+---------->			P1.2	Port 1 bit 2
					+-------------->				P1.3	Port 1 bit 3
				+------------------>					P1.4	Port 1 bit 4
			+---------------------->						P1.5	Port 1 bit 5
		+-------------------------->							P1.6	Port 1 bit 6
		+----------------------------->							P1.7	Port 1 bit 7
SCON	98	9F	9E	9D	9C	9B	9A	99	98	Serial Control
									+--> RI	Rcvr interrupt flag
								+------> TI		Xmit interrupt flag
							+----------> RB8			9th bit rcvd/stop bit
						+--------------> TB8				9th bit transmission
					+------------------> REN					Receiver enable
				+----------------------> SM2						Serial Port mode sltt
			+--------------------------> SM1							Serial Port mode sltt
		+-----------------------------> SM0								Serial Port mode sltt
SBUF	99	-	-	-	-	-	-	-	-	Serial Data Buffer
P2	A0	A7	A6	A5	A4	A3	A2	A1	A0	Port 2
									+--> P2.0	Port 2 bit 0
								+------> P2.1		Port 2 bit 1
							+----------> P2.2			Port 2 bit 2
						+--------------> P2.3				Port 2 bit 3

```
|    |    |    +-------------------> P2.4  Port 2 bit 4
|    |    +-----------------------> P2.5  Port 2 bit 5
|    +---------------------------> P2.6  Port 2 bit 6
+-------------------------------> P2.7  Port 2 bit 7
```

SPECIAL FUNCTION REGISTER ASSIGNMENTS (cont.)

Reg Name	Byte Addr	Bit Address 7	6	5	4	3	2	1	0	Byte / Bit Description
IE	A8	AF	AE	AD	AC	AB	AA	A9	A8	Interrupt Enable register
									EX0	Enable interrupt INT 0
								ET0		Timer 0 overflow int
							EX1			Enable interrupt INT 1
						ET1				Timer 1 overflow int
					ES					Serial Port interrup t
				ET2						Timer 2 ovflow or Ca p
		EA								Enable all interrupt s
P3	B0	B7	B6	B5	B4	B3	B2	B1	B0	Port 3
									P3.0	Port 3 bit 0
								P3.1		Port 3 bit 1
							P3.2			Port 3 bit 2
						P3.3				Port 3 bit 3
					P3.4					Port 3 bit 4
				P3.5						Port 3 bit 5
			P3.6							Port 3 bit 6
		P3.7								Port 3 bit 7
IP	B8	BF	BE	BD	BC	BB	BA	B9	B8	Interrupt Priority Control
									PX0	INT0 priority
								PT0		Timer 0 priority
							PX1			INT1 priority
						PT1				Timer 1 priority
					PS					Serial Port priority
				PT2						Timer 2 priority
T2CON	C8	CF	CE	CD	CC	CB	CA	C9	C8	Timer / Counter 2 control
									T2CP	Capture/reload flag
								T2C		Internal/Ext select
							TR2			Timer 2 start/stop

```
                    |   |   |   |   +--------------> EXEN2 Timer 2 ext enable
                    |   |   |   +------------------> TCLK  Transmit clock flag

                    |   |   +----------------------> RCLK  Receive clock flag
                    |   +--------------------------> EXF2  Timer 2 external fla
g
                    +------------------------------> TF2   Timer 2 overflow fla

g
```

SPECIAL FUNCTION REGISTER ASSIGNMENTS (cont.)

Reg Name	Byte Addr	Bit Address 7	6	5	4	3	2	1	0	Byte / Bit Description
RCAP2L	CA	-	-	-	-	-	-	-	-	T/C 2 Capture reg. low byte
RCAP2H	CB	-	-	-	-	-	-	-	-	T/C 2 Capt reg. high byte
TL2	CC	-	-	-	-	-	-	-	-	Timer / Counter 2 low byte
TH2	CD	-	-	-	-	-	-	-	-	Timer / Counter 2 high byte
PSW	D0	D7	D6	D5	D4	D3	D2	D1	D0	Program Status Word
									P	ACC parity flag
									--	User definable flag
									OV	Overflow flag
									RS0	Reg bank select 0
									RS1	Reg bank select 1
									F0	Gen purpose Flag 0
									AC	Auxiliary carry flag
									CY	Carry flag
ACC	E0	E7	E6	E5	E4	E3	E2	E1	E0	Accumulator
									ACC.0	Accumulator bit 0
									ACC.1	Accumulator bit 1
									ACC.2	Accumulator bit 2
									ACC.3	Accumulator bit 3
									ACC.4	Accumulator bit 4
									ACC.5	Accumulator bit 5
									ACC.6	Accumulator bit 6
									ACC.7	Accumulator bit 7
B	F0	F7	F6	F5	F4	F3	F2	F1	F0	B Register
									B.0	B register bit 0
									B.1	B register bit 1
									B.2	B register bit 2
									B.3	B register bit 3
									B.4	B register bit 4
									B.5	B register bit 5
									B.6	B register bit 6
									B.7	B register bit 7

3.10 INTERRUPT VECTORS & PRIORITY LEVELS

Event	Vector	Priority level
Reset	000	
External INT0	003	- Highest priority
Counter / Timer 0	00B	
External INT1	013	
Counter / Timer 1	01B	
Serial RCV & XMIT flag	023	
Timer 2 & External 2	02B	- Lowest priority

3.11 USER RAM MEMORY MAP

Byte Addr	Bit Address 7	6	5	4	3	2	1	0	Byte / Bit Description
30 -> 7F	-	-	-	-	-	-	-	-	80 bytes general user RAM
2F	7F	7E	7D	7C	7B	7A	79	78	Bit addressable RAM location
2E	77	76	75	74	73	72	71	70	Bit addressable RAM location
2D	6F	6E	6D	6C	6B	6A	69	68	Bit addressable RAM location
2C	67	66	65	64	63	62	61	60	Bit addressable RAM location
2B	5F	5E	5D	5C	5B	5A	59	58	Bit addressable RAM location
2A	57	56	55	54	53	52	51	50	Bit addressable RAM location
29	4F	4E	4D	4C	4B	4A	49	48	Bit addressable RAM location
28	47	46	45	44	43	42	41	40	Bit addressable RAM location
27	3F	3E	3D	3C	3B	3A	39	38	Bit addressable RAM location
26	37	36	35	34	33	32	31	30	Bit addressable RAM location
25	2F	2E	2D	2C	2B	2A	29	28	Bit addressable RAM location
24	27	26	25	24	23	22	21	20	Bit addressable RAM location
23	1F	1E	1D	1C	1B	1A	19	18	Bit addressable RAM location
22	17	16	15	14	13	12	11	10	Bit addressable RAM location
21	0F	0E	0D	0C	0B	0A	09	08	Bit addressable RAM location
20	07	06	05	04	03	02	01	00	Bit addressable RAM location
18 -> 1F	-	-	-	-	-	-	-	-	Bank 3 registers R0 -> R7
10 -> 17	-	-	-	-	-	-	-	-	Bank 2 registers R0 -> R7
08 -> 0F	-	-	-	-	-	-	-	-	Bank 1 registers R0 -> R7
00 -> 07	-	-	-	-	-	-	-	-	Bank 0 registers R0 -> R7