

INTERMIDI



HARDWARE

AND

SOFTWARE

© 1993 InterMidi Inc., All Rights Reserved

TABLE OF CONTENTS

INTRODUCTION.....	3
SECTION I - SPECIFICATIONS	3
SECTION II - HARDWARE.....	6
PTM UNIT.....	6
PTM SYSTEMS.....	12
PTM CONTROL.....	15
SECTION III - SOFTWARE.....	20
PROCESS	20
SOURCE FILE GENERATION	27
SOURCE FILE COMPILATION (RULE GENERATION).....	32
COMPILATION ERRORS.....	32

LIST OF FIGURES

FIGURE 1 - PTM Circuit Board Layout (MTM-1.00).....	6
FIGURE 2 - J12 PTM Number Assignment.....	7
FIGURE 3 - J12 Options Assignment.....	9
FIGURE 4 - J12 Self-Test LED B Patterns	10
FIGURE 5 - Typical PTM System	13
FIGURE 6 - MIDI Pipe Driver Boards	14
FIGURE 7 - PTM Sysex Control Format.....	15
FIGURE 8 - PTM Development System	20
FIGURE 9 - Sample Screen, PTM DOS Transactions.....	22
FIGURE 10 - Sample Screen, PTM Compiler Errors.....	23
FIGURE 11 - Sample Screen 1, PTM Error File.....	24
FIGURE 12 - Sample Screen 2, PTM Error File.....	25
FIGURE 13 - Sample Screen 3, PTM Error File.....	26

LIST OF TABLES

TABLE 1 - General PTM Specifications	4
TABLE 2 - PTM MIDI Implementation Chart	5
TABLE 3 - PTM Diagnostic Table.....	11
TABLE 4 - PTM Transaction Code Summary.....	17
TABLE 5 - Coupler Data Byte Definition.....	18
TABLE 6 - Stop Data Byte Definition	19
TABLE 7 - Compiler DOS Command Line.....	21
TABLE 8 - Specification Sequence	27
TABLE 9 - Delimiter Syntax.....	27
TABLE 10 - MIDI Note Numbers	30
TABLE 12 - Compiler Error Messages.....	33

Manual date: 2/93

PTM CPU code revision at this writing: 1.24

PTM compiler version at this writing: 1.00

INTRODUCTION

This manual describes the operation of InterMidi's Pipe Traffic Manager (PTM). PTM is a fully programmable and expandable MIDI traffic controller intended for use in Total MIDI Pipe Systems. It also adds traditional Pipe System control to modern synthesizers. This microprocessor based device implements stops, coupling and mixtures by responding to MIDI Sysex codes. Each PTM contains battery backed memory to retain programmed stops and mixtures rules. These rules can be changed to suit the user. InterMidi's PTM Programmer Software, written for PC compatibles, provides convenient programming features. Different system configurations can be saved, recalled and uploaded into PTM. This manual describes both PTM hardware and software.

SECTION I - SPECIFICATIONS

General PTM capabilities are summarized in TABLE 1. These capabilities are discussed more fully in the sections that follow. The PTM MIDI implementation chart is provided in TABLE 2.

If you have any questions regarding your particular system requirements, please call InterMidi. We have hardware solutions that permit MIDI to be added to all sizes and types of Pipe Systems.

UPGRADES, POLICY AND PLANS

Continuing product support is important. The specifications presented with this release are the results of customer inputs and represent a solution for the majority of users. Software revisions are currently under way, however, to add even more PTM capability. You may not need it now, but we want to make you aware of planned revisions. They will add:

- 1) more system flexibility
- 2) more stops processing power

InterMidi's first priority is to increase system flexibility. This includes such features as the added expression and tremolo assignment capabilities mentioned in the TABLE 1 notes. Several other system flexibility features are in the works, primarily to facilitate the integration of synthesizers into pipe systems.

As a second priority, InterMidi is addressing even more stops processing power. We have recently upgraded the PTM hardware to allow a substantial increase in the amount of stop definition memory. Even though initial surveys concluded that the majority of unified stop installations are accommodated with the current specifications, we plan more.

No revisions will change the basic Sysex control formats described in the following sections. PTM will be downward compatible so that console designers will always be able to use enhanced PTMs in their original systems. If a new revision adds something you like, InterMidi will upgrade you at cost. Upgrades usually involve a new compiler disk and a new PTM CPU. If you return the CPU, your cost will be minimal.

TABLE 1 - General PTM Specifications

FUNCTION	SPECIFICATION	NOTE
INPUT	1 MIDI INPUT	
OUTPUT	2 MIDI OUTPUTS (controls 2 ranks)	

OPERATING MODE OPTIONS	three jumper selections modify operating mode: 1) pipe/synth 2) intra-div coupling enable/disable 3) inter-div coupling enable/disable	4
DIVISION CONTROL	16 DIVISIONS (one for each MIDI chan)	
STOP CONTROL	14 STOPS PER DIVISION (224 stops)	1
COUPLER CONTROL	INTRA-DIVISIONAL (18): 4', Unison Off, 16' for six divisions INTER-DIVISIONAL (3): Division 1 -> Division 2 Division 1 -> Division 4 Division 2 -> Division 4	2
EXPRESSION CONTROL	Passes through designated PTM	3
TREMOLO CONTROL	Passes through up to two designated PTMs	3
SUSTAIN CONTROL	Passes through all PTMs	
ALL NOTES OFF CONTROL	Clears all PTMs and passes through to clear all equipment attached to PTM	
STOP DEFINITION	Programmable	
COUPLER DEFINITION	Internally Defined	2
EXPRESSION ASSIGNMENT	Programmable	
TREMOLO ASSIGNMENT	Programmable	

Note 1: This is an arbitrary limit. New PTM hardware permits more memory for rules and stops and this limit can be increased.

Note 2: TABLE 6 lists the couplers currently identified. Shaded cells indicate the couplers incorporated in the PTM version 1.24 software. The remainder of these functions are currently being added.

Note 3: This specification is currently being changed to permit this control to pass through multiple designated PTMs.

Note 4: Not incorporated into PTM revision 1.24.

TABLE 2 - PTM MIDI Implementation Chart

InterMidi PTM
Pipe Traffic Manager

Date: 2/7/93

Version 1.0

FUNCTION	TRANSMITTED	RECOGNIZED	REMARKS
BASIC CHANNEL	1-2	1-16	Rank1 output = ch1 Rank2 output = ch2
MODE MESSAGES	X	X	
NOTE NUMBER	24-120 0-127	36-96 36-96	pipe mode synth mode
VELOCITY note on note off	O O	O v = 1-127 O (also 9n v = 0)	
AFTER TOUCH	X	X	
PITCH BENDER	X (pipe mode) O (synth mode)	O	
CONTROL CHANGE 01 07 64	OX O O	OX O O	Modulation (hex) synth mode: Bn 01 nn = 0 - 7F pipe mode: Bn 01 01 = #1 off Bn 01 11 = #1 on Bn 01 02 = #2 off Bn 01 22 = #2 on Master Volume Sustain (hex) Bn 40 7F = on Bn 40 00 = off
PROGRAM CHANGE	OX	OX	pipe mode: X synth mode: O
SYSTEM EXCLUSIVE	O	O	see Section II, PTM Control
SYSTEM COMMON	X	X	
SYSTEM REAL TIME	X	X	
AUX MESSAGES Local On/Off All Notes Off Active sense Reset	X O X X	X O X X	
NOTES	O = yes X = no OX=selectable		

SECTION II - HARDWARE

PTM UNIT

The basic PTM unit is illustrated in Figure 1 below. MIDI IN receives MIDI data from the system console. MIDI THRU is provided to pass an unaltered, buffered copy of the MIDI input to another PTM or to other MIDI equipment. The RANK1 and RANK2 outputs connect to two MIDI to PIPE driver boards. Systems with more than two ranks of pipes can be created by using more than one PTM. Jumpers provide a unique system address for each PTM. Figure 1 illustrates the location of these jumpers (J12). Figure 2 illustrates the use of J12 to assign PTM system addresses. Although Figure 2 only illustrates addressing for a 32 rank system, the jumper area noted "not currently used" can provide addresses for hundreds of ranks.

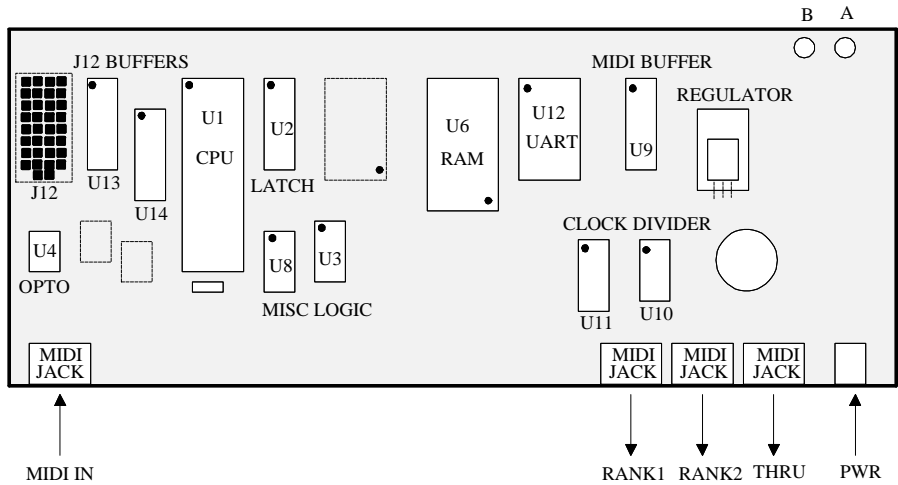
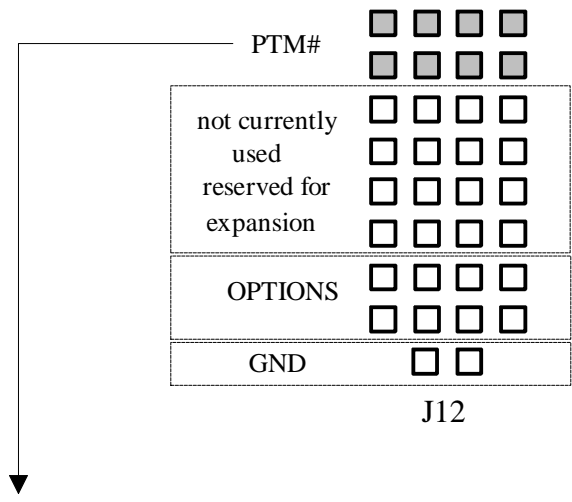


FIGURE 1 - PTM Circuit Board Layout (MTM-1.00)



<p>PTM#1 RANKS 1 & 2</p>	<p>PTM#9 RANKS 17 & 18</p>
<p>PTM#2 RANKS 3 & 4</p>	<p>PTM#10 RANKS 19 & 20</p>
<p>PTM#3 RANKS 5 & 6</p>	<p>PTM#11 RANKS 21 & 22</p>
<p>PTM#4 RANKS 7 & 8</p>	<p>PTM#12 RANKS 23 & 24</p>
<p>PTM#5 RANKS 9 & 10</p>	<p>PTM#13 RANKS 25 & 26</p>
<p>PTM#6 RANKS 11 & 12</p>	<p>PTM#14 RANKS 27 & 28</p>
<p>PTM#7 RANKS 13 & 14</p>	<p>PTM#15 RANKS 29 & 30</p>
<p>PTM#8 RANKS 15 & 16</p>	<p>PTM#16 RANKS 31 & 32</p>

FIGURE 2 - J12 PTM Number Assignment

Figure 3 illustrates the use of four jumpers to select PTM performance options. The first option adjusts the PTM output for either a pipe or synthesizer sound source. This is necessary because MIDI pipe control boards need a MIDI note number offset of -12 whereas a synthesizer needs no MIDI note offset. With the jumper installed, the synthesizer option is selected and MIDI note numbers pass through the PTM without modification. With no jumper, the pipe option is selected and all MIDI pipe traffic is output from the PTM twelve note numbers lower than they arrived. This is necessary in a MIDI system in order to accommodate a 97 pipe rank. The normal top of a MIDI keyboard generates MIDI note number 96. If the PTM was programmed to play pipe 97 when the top note of a division was played, it would need to output MIDI note number 133, which is illegal. Subtracting 12 from the output keeps PTM in the legal MIDI range (0 to 127). InterMidi Pipe Driver Boards are designed to be controlled over this range (24 to 121).

The second option provides a means of disabling intra-divisional coupling in a given PTM. This can be used to limit a potentially overwhelming resource, such as a rank of trumpets, or to conserve the polyphonicity of synthesizers. The third option provides a means of disabling inter-divisional coupling for similar reasons.

The final jumper provides a means of performing a PTM hardware self-test. This jumper should never be installed during normal operation. Self-test verifies all PTM input and output. It checks to insure that all jumper positions are active and being sensed by the microprocessor. It also checks to make sure that MIDI is being transmitted from both rank outputs and being received at the MIDI input.

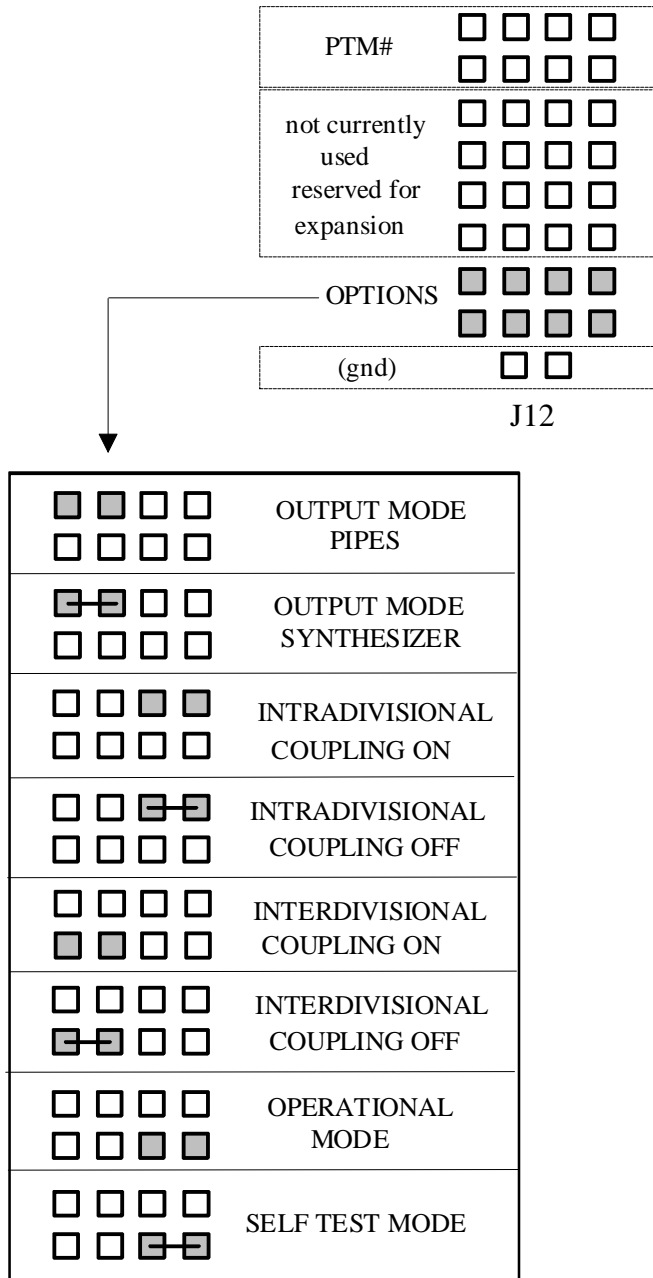


FIGURE 3 - J12 Options Assignment

To perform self-test, remove all jumpers from the PTM board except, of course, the self test jumper. Refer to Figure 1 throughout the following description to locate the parts under test. Proceed to short each of the PTM J12 inputs, one at a time and observe PTM LED A. LED A will flash a semaphore indicating its location. Figure 4 summarizes the pattern, which is a count of long and short flashes. These are arranged in such a way that you can detect a buffer IC failure in either U13 or U14. Table 1 provides diagnostic information. Multiple failures without pattern tend to point to the CPU. Use only one jumper and move it from position to position. If more than one test jumper is used, the self-test code scans to the first jumper it finds and ignores the rest.

To test the MIDI input and outputs, connect a MIDI cable between the MIDI IN input and either the RANK1 or RANK2 outputs. A sequential data pattern is transmitted at each rank output. If the same pattern is read at MIDI IN, the PTM LED B is illuminated. If the connection is broken, or the PTM MIDI path is not operational, LED B is extinguished. To check both outputs you simply leave MIDI IN connected and transfer the cable between the RANK1 and RANK2 outputs. Table 3 indicates the most likely components to be associated with a failure indication.

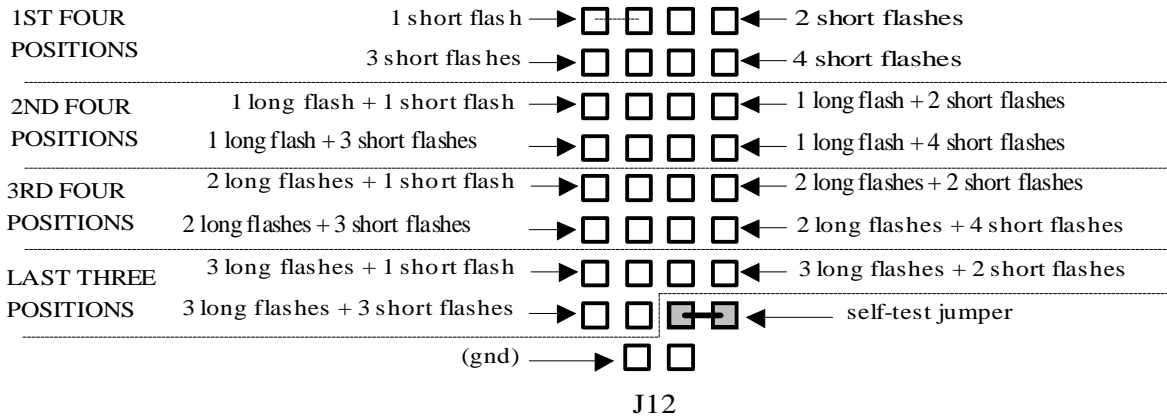


FIGURE 4 - J12 Self-Test LED A Patterns

TABLE 3 is not intended to provide exhaustive diagnostic information. It does enable you, however, to diagnose many simple failures yourself. If you have another PTM you can swap suspect parts, since InterMidi sockets all ICs. TABLE 3 should speed troubleshooting by providing a place to start, rather than leaving you to randomly swap various combinations of parts.

TABLE 3 - PTM Diagnostic Table

FAULT INDICATION	ASSOCIATED PARTS	RECOMMENDATION
1, 2 or 3 of first four positions does not indicate correctly, but at least 1 position works.	First 1/2 of buffer U13	Replace U13
All of first four positions do not indicate correctly	First 1/2 of buffer U13 Enable 1 from CPU, U1-1	Try replacing U13 first, then U1. (Check for low going strobe at U1-1 if you can).
1, 2 or 3 of second four positions does not indicate correctly, but at least 1 position works.	Second 1/2 of buffer U13	Replace U13
All of second four positions do not indicate correctly	Second 1/2 of buffer U13 Enable 2 from CPU, U1-2	Try replacing U13 first, then U1. (Check for low going strobe at U1-2 if you can).
1, 2 or 3 of third four positions does not indicate correctly, but at least 1 position works.	First 1/2 of buffer U14	Replace U14.
All of third four positions do not indicate correctly	First 1/2 of buffer U14 Enable 3 from CPU, U1-3	Try replacing U13 first, then U1. (Check for low going strobe at U1-3 if you can).
1, 2 or 3 of the last three positions does not indicate correctly.	Second 1/2 of buffer U14	Replace U14. (Self-test jmptr is in this group. If you can run any self tests, the U1-4 enable is OK).
LED B illuminated for Rank 1, but not Rank 2.	Buffer, U9 Inverter, U3 UART, U12	Try replacing U9 first, then U3, and finally U12.
LED B illuminated for Rank 2, but not Rank 1.	Buffer, U9 Inverter, U3 CPU, U1	Try replacing U9 first, then U3, then U1.
LED B does not illuminate for either Rank1 or Rank2.	Opto-Isolator, U4 CPU, U1	Try replacing U4 first, then U1. If you can, observe U1-10 and look for activity when there's MIDI in. Activity exonerates U4.

PTM SYSTEMS

Total MIDI pipe systems typically include several PTMs. FIGURE 5 illustrates a small PTM system. The PTM is part of the MIDI equipment normally located in the loft. Here it controls Pipe Driver boards that operate the solenoids of individual pipes. FIGURE 6 illustrates InterMidi's Pipe Driver board set. This set includes a MIDI controlled Pipe Driver board for a 61 pipe rank and expansion boards. The main board includes a Swell Shade Control output as well as two discrete Tremolo Control outputs. It also includes a port for the attachment of an expansion driver board. Expansion driver boards can be daisy chained, as shown in FIGURE 6, to add either 12, 24 or 36 pipes. In this way, a pipe controller for ranks containing 61, 73, 85 or 97 pipes can be constructed.

The console typically includes divisions, expression pedals and tremolo controls capable of generating MIDI note on/off, volume and tremolo data. The console also includes stop and coupler controls capable of generating MIDI Sysex codes appropriate for PTM. As shown in FIGURE 5, several PTMs are connected together to control multiple ranks of pipes. Each PTM controls two ranks and the "THRU" output of one PTM feeds the "MIDI IN" input of the next PTM. (Note: The "MIDI THRU" outputs are actively buffered and are inoperable if PTM is not energized).

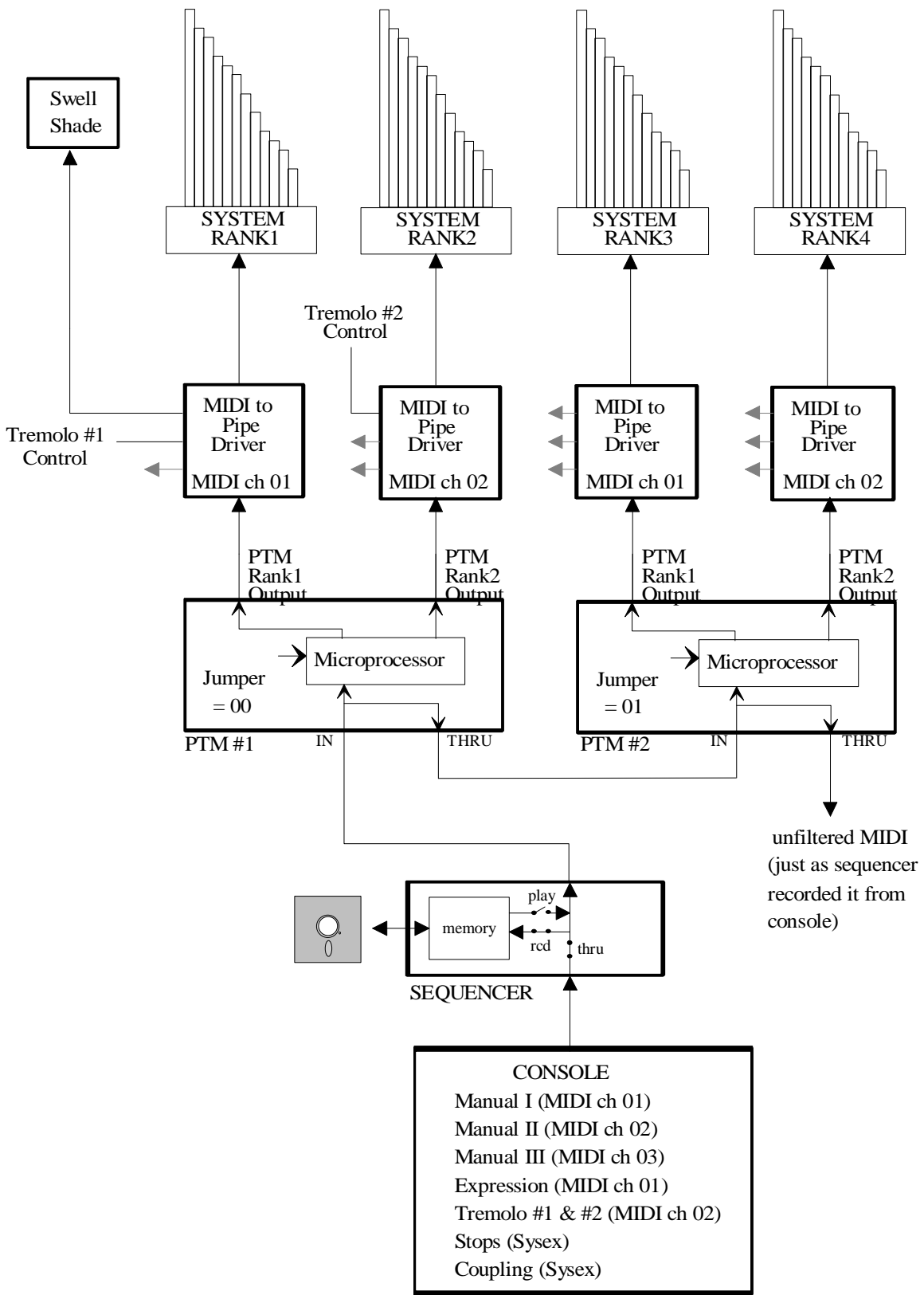


FIGURE 5 - Typical PTM System

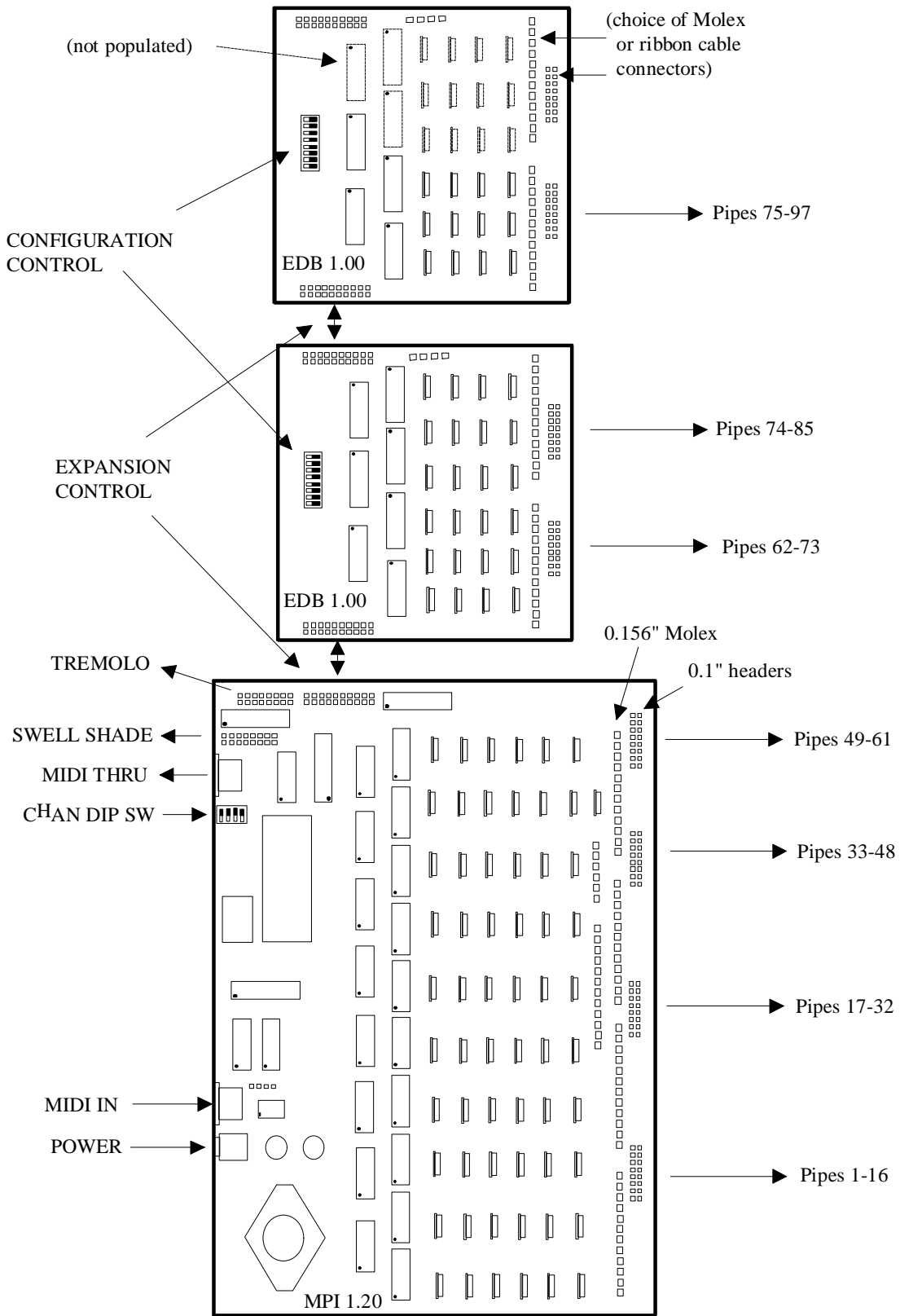


FIGURE 6 - MIDI Pipe Driver Boards

PTM CONTROL

PTM is controlled via MIDI Sysex codes. FIGURE 7, below, illustrates the basic command structure. The first byte, F0h (where the "h" indicates hexadecimal notation), signals the beginning of a MIDI Sysex transmission. The next three bytes are the InterMidi identification bytes (assigned by the International MIDI Association, IMA). The fifth byte is an InterMidi product code. This byte is required in large systems that may have more than one InterMidi product responding to Sysex controls (e.g. systems having both PTM and TCOP). The sixth byte provides the transaction type. TABLE 4 provides a summary of PTM transactions. Depending upon the transaction type, there may or may not be associated data. If there is, this data begins at the seventh byte position and continues until complete. The last byte is the MIDI EOX, or end of Sysex, byte.

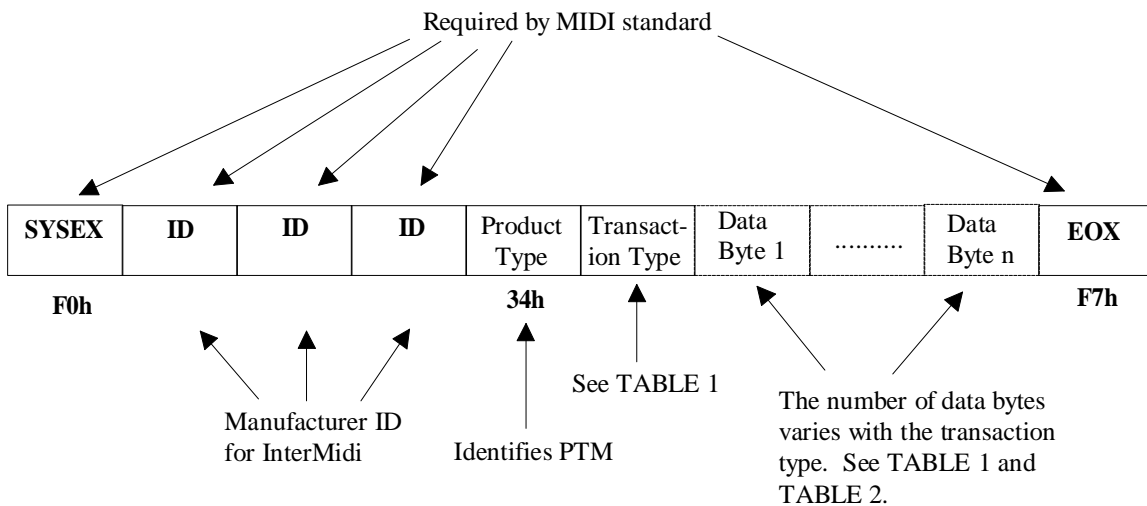


FIGURE 7 - PTM Sysex Control Format

The first transaction code in TABLE 4 is used to cause PTM to download its rules via its RANK 1 output. The rules are output complete with the MIDI Sysex codes required for a subsequent upload. In other words, the data is output:

```
F0 00 00 4F 34 05 dd dd ..... dd F7
```

Because the Sysex header, InterMidi ID, Product Code and upload transaction code are added in front and the MIDI EOX added in back, the rules are now in a form that can be turned right around and uploaded to another PTM. It can be captured with a sequencer or a PC with MIDI interface and saved for future use.

The second transaction code is used to upload a new set of rules to the PTM. The PTM compiler, for example, sends your latest rules as follows:

```
F0 00 00 4F 34 05 dd dd ..... dd F7
```

The remaining transaction bytes are associated with couplers and stops. The information is partitioned in a manner that optimizes data transmission. Couplers have a separate transaction byte from stops. Stops

are further partitioned by division. This allows as many as 70 coupler controls to go out in a coupler transmission and 14 stop controls for any one division. The overall Sysex transmission is limited to the seven overhead bytes for MIDI Sysex, ID, Product Code and PTM transaction byte plus the data byte count indicated in the table. Transmissions range from 7 to 18 bytes. The range reflects the fact that an intelligent console controller will not retransmit unchanged data bytes that occur after a byte containing a change. MIDI traffic of this duration will not significantly delay other performance traffic, such as note on, note off, volume and tremolo data. PTM can easily add more stops per division by inserting additional data bytes, each byte adding seven stops.

The coupler transaction byte is used to signal the transmission of up to 10 bytes (70 bits) of coupler status. TABLE 5 lists the interpretation of each bit. Not all bits are assigned and as of this publication date, not all bits are incorporated. The shaded cells indicate all of the active coupler functions for PTM software revision 1.24. The table presents InterMidi's plans, based upon research to date.

To transmit the command for inter-divisional coupling between Divisions 1 to Division 2 (8'), the following Sysex command string would be issued:

F0 00 00 4F 34 6F 10 F7

The sixth byte identifies a coupler transaction. The seventh byte contains the coupler data. Looking at TABLE 5, we can see that bit 4 of data byte 1 needs to be set (10 hex). Note that bytes 2 through 10 were not transmitted. It is not necessary to retransmit data beyond the point of change if it has not changed. PTM will assume that this information remains unchanged.

Stop Sysex transactions are divided by console division. To activate Division 2 stops 1 and 9, the following would be transmitted:

F0 00 00 4F 34 71 40 20 F7

The sixth byte identifies a Division 2 stop transaction. The seventh and eighth bytes contain the stop data. As shown in TABLE 6, stop 1 is bit 6 of data byte 1 (40 hex) and stop 9 is bit 5 of data byte 2 (20 hex). As in the case of coupling, unchanged data need not be retransmitted. For example, if stop 1 were now turned off, the transmission would be as follows:

F0 00 00 4F 34 71 00 F7

TABLE 4 - PTM Transaction Code Summary

TT hex	FUNCTION	# DATA BYTES
00-03	(reserved)	-
04	Configuration - download rules from PTM	8197
05	Configuration - upload rules to PTM	8197
0B-6E	(reserved)	-
6F	Control - couplers on/off	1 - 10
70	Control - stops on/off (for MIDI ch1 division)	1 - 2
71	Control - stops on/off (for MIDI ch2 division)	1 - 2
72	Control - stops on/off (for MIDI ch3 division)	1 - 2
73	Control - stops on/off (for MIDI ch4 division)	1 - 2
74	Control - stops on/off (for MIDI ch5 division)	1 - 2
75	Control - stops on/off (for MIDI ch6 division)	1 - 2
76	Control - stops on/off (for MIDI ch7 division)	1 - 2
77	Control - stops on/off (for MIDI ch8 division)	1 - 2
78	Control - stops on/off (for MIDI ch9 division)	1 - 2
79	Control - stops on/off (for MIDI ch10 division)	1 - 2
7A	Control - stops on/off (for MIDI ch11 division)	1 - 2
7B	Control - stops on/off (for MIDI ch12 division)	1 - 2
7C	Control - stops on/off (for MIDI ch13 division)	1 - 2
7D	Control - stops on/off (for MIDI ch14 division)	1 - 2
7E	Control - stops on/off (for MIDI ch15 division)	1 - 2
7F	Control - stops on/off (for MIDI ch16 division)	1 - 2

TABLE 5 - Coupler Data Byte Definition

DATA BYTE	BIT	COUPLER (1=coupler on;0=coupler off)	DATA BYTE	BIT	COUPLER (1=coupler on;0=coupler off)
1	7	(this bit always 0)	6	7	(this bit always 0)
1	6	(not used)	6	6	(not used)
1	5	DIV 1 -> DIV 2, 4'	6	5	DIV 1 -> DIV 4, 4'
1	4	DIV 1 -> DIV 2, 8'	6	4	DIV 1 -> DIV 4, 8'
1	3	DIV 1 -> DIV 2, 16'	6	3	DIV 5 -> DIV 4, 4'
1	2	DIV 5 -> DIV 2, 4'	6	2	DIV 5 -> DIV 4, 8'
1	1	DIV 5 -> DIV 2, 8'	6	1	DIV 3 -> DIV 4, 4'
1	0	DIV 5 -> DIV 2, 16'	6	0	DIV 3 -> DIV 4, 8'
2	7	(this bit always 0)	7	7	(this bit always 0)
2	6	(not used)	7	6	(not used)
2	5	DIV 3 -> DIV 2, 4'	7	5	(not used)
2	4	DIV 3 -> DIV 2, 8'	7	4	(not used)
2	3	DIV 3 -> DIV 2, 16'	7	3	DIV 2 -> DIV 4, 4'
2	2	DIV 6 -> DIV 2, 4'	7	2	DIV 2 -> DIV 4, 8'
2	1	DIV 6 -> DIV 2, 8'	7	1	DIV 6 -> DIV 4, 4'
2	0	DIV 6 -> DIV 2, 16'	7	0	DIV 6 -> DIV 4, 8'
3	7	(this bit always 0)	8	7	(this bit always 0)
3	6	(not used)	8	6	(not used)
3	5	DIV 3 -> DIV 5, 4'	8	5	DIV 1 4'
3	4	DIV 3 -> DIV 5, 8'	8	4	DIV 1 UNISON OFF
3	3	DIV 3 -> DIV 5, 16'	8	3	DIV 1 16'
3	2	DIV 1 -> DIV 5, 4'	8	2	DIV 5 4'
3	1	DIV 1 -> DIV 5, 8'	8	1	DIV 5 UNISON OFF
3	0	DIV 1 -> DIV 5, 16'	8	0	DIV 5 16'
4	7	(this bit always 0)	9	7	(this bit always 0)
4	6	(not used)	9	6	(not used)
4	5	(not used)	9	5	DIV 3 4'
4	4	(not used)	9	4	DIV 3 UNISON OFF
4	3	(not used)	9	3	DIV 3 16'
4	2	DIV 6 -> DIV 5, 4'	9	2	DIV 6 4'
4	1	DIV 6 -> DIV 5, 8'	9	1	DIV 6 UNISON OFF
4	0	DIV 6 -> DIV 5, 16'	9	0	DIV 6 16'
5	7	(this bit always 0)	10	7	(this bit always 0)
5	6	(not used)	10	6	(not used)
5	5	DIV 3 -> DIV 1, 4'	10	5	DIV 4 4'
5	4	DIV 3 -> DIV 1, 8'	10	4	DIV 4 UNISON OFF
5	3	DIV 3 -> DIV 1, 16'	10	3	DIV 4 16'
5	2	DIV 5 -> DIV 1, 4'	10	2	DIV 2 4'
5	1	DIV 5 -> DIV 1, 8'	10	1	DIV 2 UNISON OFF
5	0	DIV 5 -> DIV 1, 16'	10	0	DIV 2 16'

TABLE 6 - Stop Data Byte Definition

DATA BYTE	BIT	STOP NUMBER (1=stop on;0=stop off)
1	7	(this bit always 0)
1	6	1
1	5	2
1	4	3
1	3	4
1	2	5
1	1	6
1	0	7
2	7	(this bit always 0)
2	6	8
2	5	9
2	4	10
2	3	11
2	2	12
2	1	13
2	0	14

SECTION III - SOFTWARE

PROCESS

This section describes the process of developing your systems stop/coupling rules and placing them into PTM. The PTM development system is illustrated in FIGURE 8. The key components are an IBM PC compatible computer plus KEY MIDIator, which is an RS232 to MIDI interface. The process simply involves typing your specification into your favorite word processor and then invoking the PTM compiler by executing a single DOS command line.

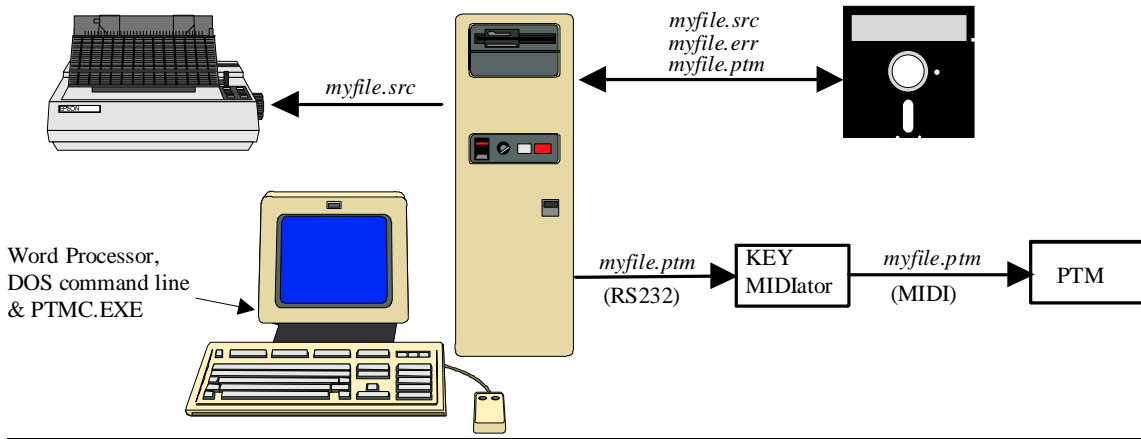


FIGURE 8 - PTM Development System

The DOS command line format has three variations, summarized in TABLE 7. In all cases, the command line requires the name of your source file, which is a plain ASCII text file with the extension .src. It's not necessary to include the extension on the command line.

TABLE 7 - Compiler DOS Command Line

VARIATION	DOS COMMAND LINE	FUNCTION
1	PTMC filename	Compile the file filename.src. If no errors place an uploadable rules file into filename.ptm. If there were errors, generate the error file filename.err .
2	PTMC filename /u	Compile filename.src. If no errors, place an uploadable rules file into filename.ptm <u>and</u> upload a copy of these rules into PTM. If there were errors, generate the rules file filename.err and do not upload any rules.
3	PTMC /u filename	Upload a copy of an existing rules file, filename.ptm to a PTM. Do not compile.

The first variation is typically used when you are simply designing the system and checking your entries for syntax errors with periodic compilations. If there are any errors, an error file (filename.err) is generated and a rules file (filename.ptm) is not. This error file has all of your original specifications, as entered, along with error messages located at the point of error. You may then correct your source file and recompile. When you are successful, the uploadable rules file (filename.ptm) is generated and placed on your disk in the current directory.

The second variation is useful when you have made very minor changes to your specification and would simply like to re-compile the PTM rules and upload. If there should be a problem, the compiler does not change the old rules file (filename.ptm) or upload anything. It will simply generate the error file.

The third and final variation simply uploads an existing rules file (filename.ptm). This is useful for initializing PTMs at an installation or experimenting with different, previously developed rules sets.

Input and response sequences, as seen on your computer screen, are illustrated in FIGURE 9 through 11. FIGURE 9 illustrates each of the DOS command line variations and the associated compiler response.

Typical information, when there are no errors, includes:

- compiler version
- confirmation that there were no errors
- percent of all PTM rules memory used
- confirmation that a rules file was created
- notification of Key MIDIator initialization (variation 2 or 3)
- confirmation that Key MIDIator was found (variation 2 or 3)
- notification of disk transfer of file to memory (variation 3)
- confirmation that the transfer completed (variation 2 or 3)

```
C:\PTM>ptmc myfile
PTM Compiler version 1.00
No errors.
(10 percent of PTM rules memory used).
myfile.ptm file created.

C:\PTM>ptmc myfile /u
PTM Compiler version 1.00
No errors.
(10 percent of PTM rules memory used).
myfile.ptm file created.
Initializing MIDI interface; looking for KEY MIDIator.
Using the MIDIator found on COM1.
Upload of rules to PTM completed.

C:\PTM>ptmc /u myfile
Initializing MIDI interface; looking for KEY MIDIator.
Using the MIDIator found on COM1.
Transferring myfile.ptm from disk to memory.
Upload of rules to PTM complete.
```

FIGURE 9 - Sample Screen, PTM DOS Transactions

FIGURE 10 illustrates the screen output associated with a compiler error. Typical information is:

- compiler version
- error messages
- notification that compilation completed
- number of errors
- notification and name of the error file created

If there are a lot of errors, the compiler version and all but the last 23 errors will scroll off the screen. Notification of compilation complete, total number of errors and the name of the error file created are always visible as the last three lines.

```
C:\PTM>ptmc myfile
PTM Compiler version 1.00
>>>ERROR 10: illegal ZONE number.<<<
>>>ERROR 22: PITCH assignment out of sequence.<<<
>>>ERROR 08: unknown RANK.<<<
Rules compilation completed.
Total errors = 3
( See file myfile.err for error location. )
```

FIGURE 10 - Sample Screen, PTM Compiler Errors

The following three screens illustrate how the three errors indicated above in FIGURE 10 might appear. In this example, MSDOS 5.0 Edit program was used to examine myfile.err. These screens preview some of the rules associated with the generation of a source file. All rules are discussed in the next section.

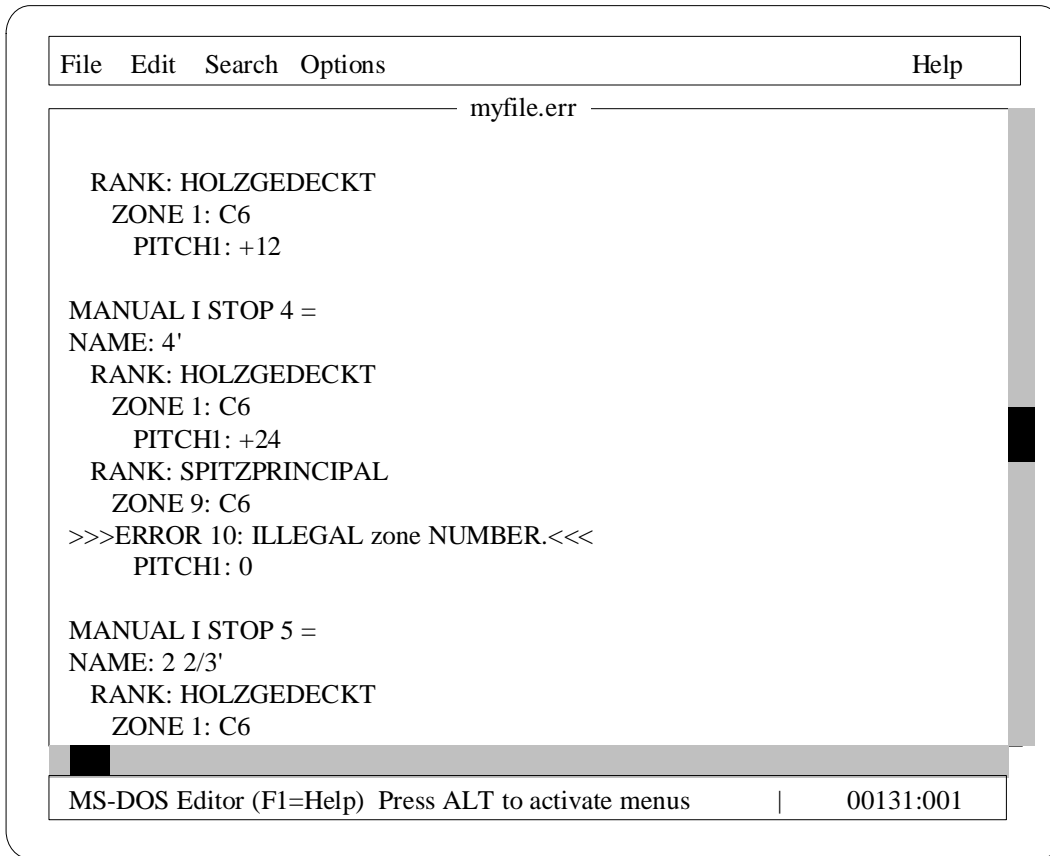


FIGURE 11 - Sample Screen 1, PTM Error File

In FIGURE 11, the error message appears immediately under the source code error. "ZONE 9:" is an illegal zone number since there can only be six zones. If you are careful, you can repair the .err file and "save as" a .src file. You have to remember to delete the error messages, however. Otherwise, this error file is an image of your source file, complete with your original comment lines.

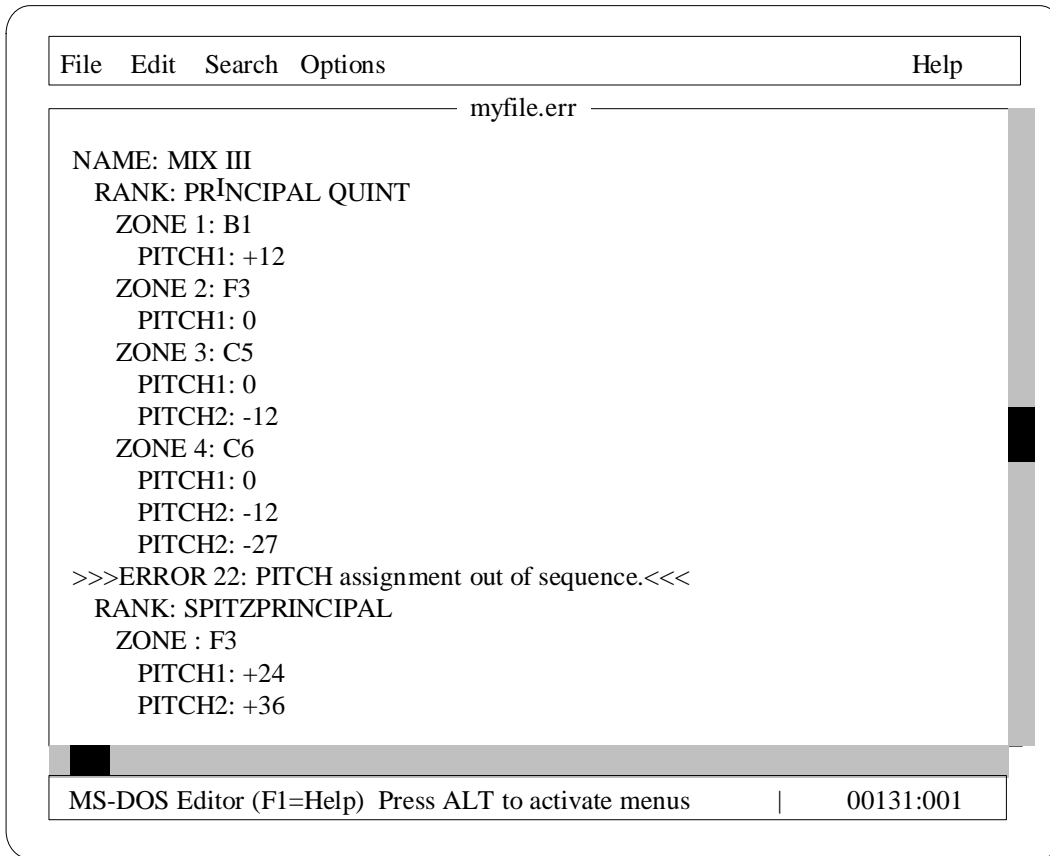


FIGURE 12 - Sample Screen 2, PTM Error File

The error message in FIGURE 12 appears directly below the offending pitch assignment. The compiler expects PITCH3 next and considers anything else to be "out of sequence". You can get this same kind of error if you assign a zone, for example, out of sequence.

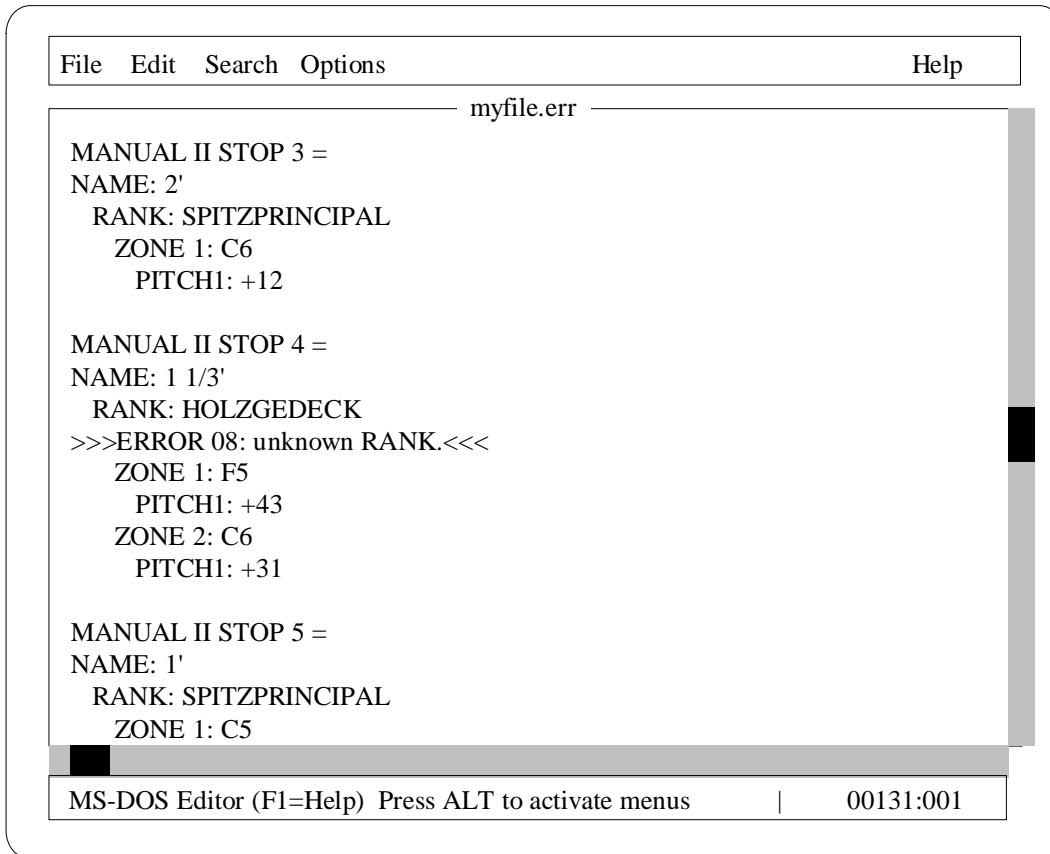


FIGURE 13 - Sample Screen 3, PTM Error File

FIGURE 13 illustrates a common error resulting from a simple "typo". As you will see in the next section, before you define stops, you tell the PTM compiler the names of all the ranks. This provides a list of legal rank names for use during rules definition. In FIGURE 13, the error stems from the fact that the "T" was left off of HOLZGEDECKT. The compiler doesn't know this was a typographical error, and assumes that you tried to assign an undefined rank.

Now that you have an overview of the process of generating and uploading stops rules for PTM, its time to look at how you specify your design in a source file. The following section discusses the sequence, syntax and meaning of source file entries. This is quite a bit simpler and more intuitive than learning a computer language.

SOURCE FILE GENERATION

As mentioned above, the source file is generated by any plain ASCII file editor. The MSDOS 5.0 editor, for example, works well. You can use your favorite word processor as long as you place it into a plain ASCII text file mode. This is sometimes referred to as "non-document" or "text only" mode. It does not add any of the word processor specific formatting codes to the text. The file you create should be saved with the extension .src . Most word processors have the ability to save a file with any extension you choose. This is usually under your file menu as the "Save As.." option.

The rules required to enter your specification have been kept minimal and logical. The rules must be entered in a sequence, since they can build upon each other, and there is a simple syntax involved. Table 5 summarizes the sequence of your entries and Table 6 summarizes the three syntax delimiters you must remember. Delimiters simply signal the rules generation program, the compiler, of an important event.

TABLE 8 - Specification Sequence

STEP	SPECIFY
1	System Title
2	Division Names and their associated MIDI channels
3	Rank names
4	Tremolo Rank and MIDI channel
5	Expression Rank and MIDI channel
6	Stop Division and Number
7	Stop Name
8	Rank
9	Division Zone
10	Zone Pitches
11	(repeat 6,7,8,9&10 for up to 16 divisions, 14 stops per division)

TABLE 9 - Delimiter Syntax

DELIMITER	USAGE
!	Comments
=	Declarations
:	Variable assignment

The System Title is a formality, but a good practice. You simply enter the following line into your text:

```
SYSTEM=  
    My system #1
```

In the following descriptions, it is always expected that the declaration will be on its own line and that the variables associated with it will be on the following lines. Generally, the names of variables can be up to 40 characters. Also, the exact position of text on a line is not critical, as long as you don't exceed an 80 character line length. People familiar with programming often use indentation to clarify hierarchical relationships. The number of spaces between words is also not critical. The compiler doesn't care and leaves you to place text and comments in the manner you find clearest.

Division Names and their MIDI assignments are declared in the following manner:

Rank Names are declared in the following manner.

Stop definition requires several declarations. As shown in Table 5, there are five declarations associated with the definition of a stop. The first declaration states the division with which the stop is associated and the stop number. The stop number is required by PTM. It wants to know which of 14 stops it's dealing with. The declaration for Swell division, stop 1, would appear as follows:

```
Swell Stop 1 =
```

Next, the Stop name is required. The compiler doesn't need this information to generate rules, however, the name is more meaningful to the designer and future report generation software will utilize this input. This declaration appears as follows:

```
Name: Celeste
```

Next, specify the name of the Rank controlled by this stop. The name must be one of the names previously defined in the Rank Name definition step. If the Swell Stop 1, Celeste, controls the Celeste rank, the declaration appears as follows:

```
Rank: Celeste
```

Finally, zone and pitch information is specified. These two pieces of information define how a keyboard will control a given rank and are best described together. Keyboard zones and pitch offsets are required to be definable in order to effectively utilize pipe resources. A 4' stop, for example, may want to play 12 pipes higher on a given rank. The pitch declarations allow this. A 49 pipe rank may run out of pipes at the top, but the designer may wish to repeat the last 12 pipes at the top of the keyboard. Zone declarations in combination with pitch declarations allow this. Several examples follow.

If you are not specifying zones, and wish the entire keyboard to play on pitch, simply specify one zone and terminate it at C6, as follows:

```
Zone 1: C6  
Pitch1: 0
```

The top end of a zone is up to and including the specified note. Table 6 presents keyboard number versus note versus MIDI note number. In the example where we had a 49 pipe rank and wished to play the last 12 keys of the division to wrap and replay the last 12 pipes, the declarations would be as follows:

```
Zone 1: C5  
Pitch1: 0
```

Zone 2: C6
Pitch1: -12

The above is interpreted to specify that the keyboard zone from C1 through C5 would play the rank on pitch, but the keyboard zone from C#5 through C6 would play down 12 semitones.

It is also possible to have multiple pitches per zone. PTM supports up to three pitches per zone. The following is an example of a three pitch declaration:

Zone 1: C6
Pitch1: -12
Pitch2: 0
Pitch3: +12

This declaration causes the division to play on pitch as well as up and down 12 semitones. This is similar to having sub and super coupling on.

TABLE 10 - MIDI Note Numbers

KEY#	NOTE	MIDI NOTE#	KEY#	NOTE	MIDI NOTE#
1	C1	36	31	F#3	66
2	C#1	37	32	G3	67
3	D1	38	33	G#3	68
4	D#1	39	34	A3	69
5	E1	40	35	A#3	70
6	F1	41	36	B3	71
7	F#1	42	37	C4	72
8	G1	43	38	C#4	73
9	G#1	44	39	D4	74
10	A1	45	40	D#4	75
11	A#1	46	41	E4	76
12	B1	47	42	F4	77
13	C2	48	43	F#4	78
14	C#2	49	44	G4	79
15	D2	50	45	G#4	80
16	D#2	51	46	A4	81
17	E2	52	47	A#4	82
18	F2	53	48	B4	83
19	F#2	54	49	C5	84
20	G2	55	50	C#5	85
21	G#2	56	51	D5	86
22	A2	57	52	D#5	87
23	A#2	58	53	E5	88
24	B2	59	54	F5	89
25	C3	60	55	F#5	90
26	C#3	61	56	G5	91
27	D3	62	57	G#5	92
28	D#3	63	58	A5	93
29	E3	64	59	A#5	94
30	F3	65	60	B5	95
			61	C6	96

Any given stop can control multiple ranks. This is required, for example, to construct mixtures. Mixtures will utilize pipes from multiple ranks and will usually require the definition of division zones of different pitch offsets. The following is an example of controlling two ranks to create a mixture:

```

GREAT STOP 4 =
NAME: MIXTURE III
RANK: SPITZPRINCIPAL 4'           ! rank one of two, 73 pipe rank
  ZONE 1: F#3                     ! division control is keys 1-31
    PITCH1: +24                   ! play pipes 25-55
    PITCH2: +36                   ! play pipes 37-67
  ZONE 2: F4                       ! division control is keys 32-42
    PITCH1: +12                   ! play pipes 44-54
    PITCH2: +24                   ! play pipes 56-66
  ZONE 3: C5                       ! division control is keys 43-49
    PITCH1: +24                   ! play pipes 57-73
    PITCH2: +12                   ! play pipes 55-61
    PITCH3: 0                     ! play pipes 43-49
  ZONE 4: F5                       ! division control is keys 50-54
    PITCH2: +12                   ! play pipes 62-76
    PITCH3: +0                    ! play pipes 50-54
  ZONE 5: C6                       ! division control is keys 55-61
    PITCH1: +12                   ! play pipes 67-73
    PITCH2: 0                     ! play pipes 55-61
RANK: PRINCIPAL QUINT 2/3'       ! rank two of two, 42 pipe rank
  ZONE 1: B1                       ! division control is 1-12
    PITCH1: +12                   ! play pipes 13-24
    PITCH2: +0                    ! play pipes 1-12
  ZONE 2: F3                       ! division control is 13-30
    PITCH1: +0                    ! play pipes 13-30
  ZONE 3: C5                       ! division control is 31-49
    PITCH1: +0                    ! play pipes 31-49
    PITCH2: -12                   ! play pipes 19-37
  ZONE 4: C6                       ! division control is 50-61
    PITCH1: 0                     ! play pipes 50-61
    PITCH2: -12                   ! play pipes 38-49
    PITCH3: -27                   ! play pipes 13-34

```

Appendix A contains two simple example designs. It includes the source files example1.src and example2.src . These files are on your PTM Compiler diskette and you are encouraged to alter it, upload it to your PTM and listen to the changes.

SOURCE FILE COMPILATION (RULE GENERATION)

Basically, the compiler reads your source file in and interprets it. It creates all of the tables that tell the PTM microprocessor how to direct your pipe traffic in response to your controls. The output is an 8K binary file that is uploaded, via the MIDI interface, directly into the PTM battery backed RAM memory. These rules remain intact, even if power is removed from the PTM, until you upload a different set.

COMPILATION ERRORS

It's easy enough to make "typos" or inadvertently leave out a step in the creation of your specifications. During compilation, error messages are output to the screen and the total error count is indicated. Because there may be more errors than can be displayed and because you really need to know where they occurred, the PTMC creates the error file, which contains your source code with the error messages incorporated at the trouble spots. These messages contain an error number along with a short description of the problem. After a while, this is enough to tell you what's wrong. Table 12, below, lists all of the error messages you can see. A brief discussion of each error follows.

CAUTION: Don't be alarmed if there are a large number of error messages or if some of them don't seem positioned accurately. A very simple error can generate a startling error message response. For example, if you omitted the rank declaration:

```
RANK NAMES =  
  BOURDON 16'  
  PRINCIPAL 4'  
  LARIGOT 1 1/3'
```

The compiler will indicate >>>ERROR 04: no RANK name definition<<< plus >>>ERROR 08: unknown RANK.<<<. An ERROR 04 is indicated just after each stop declaration, because it is at this point the compiler realizes (repeatedly) that you left out step four, the rank definition. (This message reporting will be changed in the next compiler version such that ERROR 04 is indicated only once, just prior to stops definition). ERROR 08, of course, will indicate everywhere that you have attempted to assign a rank name. This generates a lot of errors, but it's correct and all errors will disappear as soon as you fix ERROR 04. A good approach to error correction is to fix all obvious errors first and then recompile to see what errors remain.

NOTE: IF YOU HAVE COMPILED SUCCESSFULLY AND YET FEEL YOUR PTM IS NOT RESPONDING CORRECTLY, YOU MAY ARRANGE TO TRANSMIT TO INTERMIDI, VIA MODEM, YOUR SOURCE (.SRC) FILE AS WELL AS YOUR UPLOAD FILE (.PTM). WE CAN ANALYZE THIS INFORMATION TO DETERMINE WHETHER THE PROBLEM AROSE BECAUSE THE COMPILER FAILED TO FLAG AN IMPROPER INPUT OR BECAUSE THE COMPILER OPERATED INCORRECTLY.

TABLE 12 - Compiler Error Messages

MESSAGE
ERROR 01: no SYSTEM definition.
ERROR 02: no DIVISION name definition.
ERROR 03: no DIVISION midi definition.
ERROR 04: no RANK name definition.
ERROR 05: no STOP name definition.
ERROR 06: illegal MIDI channel number.
ERROR 07: illegal STOP number.
ERROR 08: unknown RANK.
ERROR 09: illegal ZONE limit.
ERROR 10: illegal ZONE number.
ERROR 11: illegal PITCH limit.
ERROR 12: illegal PITCH number.
ERROR 13: (unused)
ERROR 14: (unused)
ERROR 15: no TREMOLO definition.
ERROR 16: no EXPRESSION definition.
ERROR 17: unknown STOP assignment operator.
ERROR 18: ZONE assignment out of sequence.
ERROR 19: label exceeds 40 characters.
ERROR 20: unknown process declaration.
ERROR 21: source line longer than 80 characters.
ERROR 22: PITCH assignment out of sequence.
ERROR 23: illegal TREMOLO assignment.
ERROR 24: duplicate MIDI chan assignment.
ERROR 25: duplicate DIVISION assignment.
ERROR 26: duplicate RANK assignment.

>>>ERROR 01: no SYSTEM definition.<<<

This error occurs if you leave out the first step, which is to assign a system title. In our example files, you would get this error if you omitted:

```
SYSTEM =  
    Example1
```

NOTE: Compiler version 1.0 places this message immediately after the next declaration, which would be " DIVISION MIDI ASSIGNMENTS = ".

>>>ERROR 02: no DIVISION name definition.<<<

This error occurs if you leave out the second step, which is to assign divisions and their MIDI channels. In our example files, you would get this error if you omitted:

```
DIVISION MIDI ASSIGNMENTS =  
    SWELL: 1  
    GREAT: 2  
    DUMMY: 4  
    PEDAL: 3
```

NOTE: Compiler version 1.0 places this message immediately after the next declaration, which would be " RANK NAMES = ".

>>>ERROR 03: not DIVISION midi definition.<<<

If you leave out the MIDI channel in a declaration, you would get this error. The highlighted lines, below, are examples of this omission.

```
DIVISION MIDI ASSIGNMENTS =  
    SWELL: 1  
    GREAT: ! error, no MIDI channel specified  
    DUMMY: 4  
    PEDAL: 3
```

```
EXPRESSION ASSIGNMENT =  
    CONTROL: ! error, no MIDI channel specified  
    RANK: PRINCIPAL 4'
```

>>>ERROR 04: no RANK name definition.<<<

This error occurs if you leave out the third step, which is to assign rank names. In our example files, you would get this error if you omitted:

```
RANK NAMES =  
  BOURDON 16'  
  PRINCIPAL 4'  
  LARIGOT 1 1/3'
```

NOTE: The compiler version 1.0 places this message immediately after the next declaration, which is a stops declaration. ERROR 08 messages will accompany ERROR 04 if any stop rank assignments were made.

>>>ERROR 05: no STOP name definition.<<<

This error occurs when you have omitted the name of a stop, as indicated below:

```
SWELL STOP 1 =
```

```
NAME: ! error, no stop name supplied
```

```
  RANK: BOURDON 16'  
  ZONE 1: C6  
  PITCH1: +12
```

>>>ERROR 06: illegal MIDI channel number.<<<

Legal MIDI channels are 1 through 16. Numbers outside this range will generate this error message.

>>>ERROR 07: illegal STOP number.<<<

Legal stop numbers are 1 through 14. Numbers outside this range will generate this error message.

>>>ERROR 08: unknown RANK.<<<

This message occurs if you use a rank name that was not defined using the rank declaration:

```
RANK NAMES =
```

Most typically it occurs because it has been inadvertently spelled differently. (Don't worry about spaces in your names. " Principal 4' " and " Principal 4 ' " will be treated the same by the compiler.)

>>>ERROR 09: illegal ZONE limit.<<<

This error occurs if you attempt to define a zone upper limit that is not on the keyboard. This would mean that you attempted to define a note outside the range of C1 to C6 as the upper limit of a zone.

GREAT STOP 1 =

NAME: PRINCIPAL 8'

RANK: BOURDON 16'

ZONE 1: A1 ! error, below C1

PITCH1: +12

RANK: PRINCIPAL 4

ZONE 1: C7 ! error, above C6

PITCH1: 0

>>>ERROR 10: illegal ZONE number.<<<

This error occurs if you attempt to define more than 6 keyboard zones.

GREAT STOP 4 =

NAME: MIXTURE III

RANK: SPITZPRINCIPAL 4'

ZONE 1: F#3

PITCH1: +24

ZONE 2: F4

PITCH1: +12

ZONE 3: F5

PITCH3: +0

ZONE 4: C5

PITCH2: +12

PITCH3: 0

ZONE 5: C6

PITCH1: 0

PITCH2: 0

ZONE 6: F5

PITCH3: -12

ZONE 7: C5

PITCH2: -24

>>>ERROR 11: illegal PITCH limit.<<<

An illegal pitch limit is one that would generate an illegal MIDI note number. The compiler checks the top of the current zone and adds the pitch offset to it and checks to see if you fall outside the MIDI note number range of 0 to 127. If you do, you get this error message.

>>>ERROR 12: illegal PITCH number.<<<

Legal pitch number assignments are PITCH 1, PITCH 2 and PITCH 3. If you attempt to assign more pitches, you generate this error message.

>>>ERROR 15: no TREMOLO definition.<<<

This message occurs if you omitted the fourth declaration, which is for tremolo:

```
TREMOLO ASSIGNMENT =  
CONTROL 1: 5  
RANK 1: PRINCIPAL 4'  
CONTROL 2: 5  
RANK 2: LARIGOT 1 1/3'
```

NOTE: Compiler version 1.0 displays this error message on the screen, but places it incorrectly in the error file. It places it after every stop declaration instead of at the point of omission.

>>>ERROR 16: no EXPRESSION definition.<<<

This message occurs if you omitted the fifth declaration, which is for expression:

```
EXPRESSION ASSIGNMENT =  
CONTROL: 5  
RANK: PRINCIPAL 4'
```

NOTE: Compiler version 1.0 incorrectly places this error in the error file. It places it after every stop declaration instead of at the point of omission. It also indicates this error multiple times on the screen (when, of course, it can only be omitted once!).

>>>ERROR 17: unknown STOP assignment operator.<<<

This error occurs if you use an undefined STOP assignment operator (or misspell a legal one). The STOP assignment operators are NAME: , RANK: , ZONE #: (where # is 1-7) and PITCH #: (where # is from 1-2) and are used as shown below:

```
SWELL STOP 1 =
NAME: FLUTE 8'
  RANK: BOURDON 16'      ! 97 pipe rank
  ZONE 1: C6             ! division control is all 61 keys
  PITCH1: +12           ! play pipes 13-73 of this rank
```

>>>ERROR 18: ZONE assignment out of sequence.<<<

This error occurs if you do not sequentially assign zones. The following source code would cause this error:

```
GREAT STOP 4 =
NAME: MIXTURE III
  RANK: SPITZPRINCIPAL 4'
  ZONE 1: F#3
  PITCH1: +24
  PITCH2: +36
  ZONE 3: F4             ! error, zone should be 2, not 3
  PITCH1: +12
  PITCH2: +24
  ZONE 2: F5
  PITCH1: +24
  PITCH2: +12
  PITCH3: +0
```

>>>ERROR 19: label exceeds 40 characters.<<<

Labels, which are the names of system, divisions, ranks and stops, must be less than 40 characters in length. This is an arbitrary limit, but it was felt that it would accommodate most labels.

>>>ERROR 20: unknown process declaration.<<<

This error occurs if you either attempt to use an undefined declaration or you inadvertently misspell one that's defined. The following source code would cause this error:

```
DVISION MIDI ASSIGNMENTS =      ! error, misspelled DIVISION
```

>>>ERROR 21: source line longer than 80 characters.<<<

This error occurs occasionally if you add comment lines that make any one line over 80 characters in length. This is an arbitrary limit, but it was felt that it most users would be listing in 80 column width and would want to be notified if they were about to wrap.

>>>ERROR 22: PITCH assignment out of sequence.<<<

This error occurs if you do not assign pitches sequentially. The following source code would cause this error:

GREAT STOP 4 =

NAME: MIXTURE III

RANK: SPITZPRINCIPAL 4'

ZONE 1: F#3

PITCH1: +24

PITCH2: +36

ZONE 2: F4

PITCH1: +12

PITCH2: +24

ZONE 3: F5

PITCH1: +24

PITCH3: +12 ! error, PITCH2 should have been next, not PITCH3

PITCH2: +0

ZONE 4: C5

PITCH1: +24

PITCH2: +12

PITCH3: 0

ZONE 5: C6

PITCH1: +12

PITCH2: 0

>>>ERROR 23: illegal TREMOLO assignment.<<<

This error occurs if you attempt anything other than a CONTROL 1: , CONTROL 2: , RANK 1: or RANK 2: assignment (e.g. CONTROL 3: 5 or RANK 3: Principal 4').

>>>ERROR 24: duplicate MIDI chan assignment.<<<

This error occurs when you have assigned the same MIDI channel to two divisions. The following source code would cause this error:

DIVISION MIDI ASSIGNMENTS =

SWELL: 1

GREAT: 2

DUMMY: 4

PEDAL: 2 ! error, channel 2 was already assigned to the GREAT

>>>ERROR 25: duplicate DIVISION assignment.<<<

This error occurs if you inadvertently assign the same division twice. The following source code would cause this error:

DIVISION MIDI ASSIGNMENTS =

SWELL: 1

GREAT: 2

DUMMY: 4

SWELL: 3 ! error, Swell division was previously defined

>>>ERROR 26: duplicate RANK assignment.<<<RANK NAMES =

This error occurs if you inadvertently assign the same rank twice. The following source code would cause this error:

RANK NAMES =

BOURDON 16'

PRINCIPAL 4'

LARIGOT 1 1/3'

BOURDON 16' ! error, BOURDON 16' was previously defined

APPENDIX A - SOURCE CODE EXAMPLES

```

! ----- example1.src ----- sht 1 of 4
! filename: example1.src
!
! This is an example PTM source file for a small pipe system having
! two manuals (Swell and Great), a pedal and three ranks of pipes.
!
! -----

! First, declare the system name.
SYSTEM =
    Example1

! Second, declare the division names and their associated MIDI channels.
! The first division named is Division 1, the second is Division 2 etc.
! Division number is important to coupling.
! Note that even though there are only three divisions in this system,
! four division names were used. Division three is a dummy division,
! or place holder. Currently available inter-divisional coupling is
!   Division 1 -> Division 2
!   Division 1 -> Division 4
!   Division 2 -> Division 4
! The use of the dummy division allows the following coupling to be
! established
!   Swell to Great
!   Swell to Pedal
!   Great to Pedal

DIVISION MIDI ASSIGNMENTS =
    SWELL: 1           ! Division 1 xmits on MIDI chan 1
    GREAT: 2           ! Division 2 xmits on MIDI chan 2
    DUMMY: 4           ! dummy division assigned unused MIDI chan
    PEDAL: 3           ! Division 4 xmits on MIDI chan 3

! Third, declare the names of the pipe ranks. The order of the pipe ranks
! determines the associated PTM output. These can be rearranged and
! grouped as desired.

RANK NAMES =
    BOURDON 16'        ! PTM#1, RANK1 output, 97 pipes
    PRINCIPAL 4'       ! PTM#1, RANK2 output, 61 pipes
    LARIGOT 1 1/3'    ! PTM#2, RANK1 output, 61 pipes

```

! ----- example1.src ----- sht 2 of 4

! Fourth, assign the tremolo location and its MIDI control channel.
! There are two tremolo controls. They can be controlled by any MIDI
! channel and located on any rank.

TREMOLO ASSIGNMENT =

CONTROL 1: 5 ! Bn 01 01 = off; Bn 01 11 = on; n=4
RANK 1: PRINCIPAL 4' ! location = Principal 4' = PTM#1, RANK2
CONTROL 2: 5 ! Bn 01 01 = off; Bn 01 11 = on; n=4
RANK 2: LARIGOT 1 1/3' ! location = Principal 4' = PTM#2, RANK1

! Fifth, assign the expression location and its MIDI control channel.
! This is done as it was for tremolo, except that there is only one control
! and therefore no numeric suffix added to CONTROL and RANK.

EXPRESSION ASSIGNMENT =

CONTROL: 5 ! Bn 07 vv; vv = 0-127, n = 4
RANK: PRINCIPAL 4' ! location = Principal 4' = PTM#1, RANK2

! Finally, you begin the entry of stop definitions. This is done division
! by division. Each division can have 14 stops. Each stop can associate
! multiple ranks. Each rank can respond to six keyboard zones. Each zone
! can control three different pitches (or pipes).

! ----- SWELL STOPS -----

SWELL STOP 1 =

NAME: FLUTE 8'

RANK: BOURDON 16' ! 97 pipe rank
ZONE 1: C6 ! division control is all 61 keys
PITCH1: +12 ! play pipes 13-73 of this rank

SWELL STOP 2 =

NAME: PRINCIPAL 2'

RANK: PRINCIPAL 4' ! rank one of two, 61 pipes
ZONE 1: C5 ! division control is lower 49 keys
PITCH1: +12 ! play pipes 13-61 of this rank
RANK: BOURDON 16' ! rank two of two, 97 pipes
ZONE 1: C6 ! division control is upper 12 keys
PITCH1: +36 ! play pipes 86-97 of this rank

! ----- example1.src ----- sht 3 of 4

SWELL STOP 3 =
NAME: LARIGOT 1 1/3'
RANK: LARIGOT 1 1/3' ! 61 pipe rank
ZONE 1: C6 ! division control is all 61 keys
PITCH1: 0 ! play pipes 1-61 of this rank

! ----- GREAT STOPS -----

GREAT STOP 1 =
NAME: PRINCIPAL 8'
RANK: BOURDON 16' ! rank one of two, 97 pipes
ZONE 1: C6 ! division control is all 61 keys
PITCH1: +12 ! play pipes 13-73 of this rank
RANK: PRINCIPAL 4 ! rank two of two, 61 pipes
ZONE 1: C5 ! division control is lower 49 keys
PITCH1: 0 ! play pipes 1-49 of this rank

GREAT STOP 2 =
NAME: FLUTE 8'
RANK: PRINCIPAL 4' ! rank one of two, 61 pipes
ZONE 1: C5 ! division control is lower 49 keys
PITCH1: +12 ! play pipes 13-61 of this rank
RANK: BOURDON 16' ! rank two of two, 97 pipes
ZONE 1: C6 ! division control is upper 12 keys
PITCH1: +36 ! play pipes 86-97 of this rank

GREAT STOP 3 =
NAME: OCTAVE 4'
RANK: PRINCIPAL 4' ! 61 pipe rank
ZONE 1: C6 ! division control is all 61 keys
PITCH1: 0 ! play pipes 1-61 of this rank

GREAT STOP 4 =
NAME: FLUTE 2'
RANK: BOURDON 16' ! 97 pipe rank
ZONE 1: C6 ! division control is all 61 keys
PITCH1: +12 ! play pipes 13-97 of this rank

! ----- example1.src ----- sht 4 of 4

! ----- PEDAL STOPS -----

PEDAL STOP 1 =

NAME: GEDECKT 16'

RANK: BOURDON 16'

ZONE 1: G3

PITCH1: 0

! 97 pipe rank

! division control is 32 pedals

! play pipes 1-32 of this rank

```
! ----- example2.src ----- sht 1 of 4
! filename: example2.src
!
! This is an example PTM source file for a small pipe system having
! two manuals (Swell and Great), a pedal and three ranks of pipes.
! This source file includes an example of a mixture.
!
```

```
! First, declare the system name.
SYSTEM =
    Example2
```

```
! Second, declare the division names and their associated MIDI channels.
! The first division named is Division 1, the second is Division 2 etc.
! Division number is important to coupling.
! Note that even though there are only three divisions in this system,
! four division names were used. Division three is a dummy division,
! or place holder. Currently available inter-divisional coupling is
!   Division 1 -> Division 2
!   Division 1 -> Division 4
!   Division 2 -> Division 4
! The use of the dummy division allows the following coupling to be
! established
!   Swell to Great 8'
!   Swell to Pedal 8'
!   Great to Pedal 8'
```

```
DIVISION MIDI ASSIGNMENTS =
    SWELL: 1           ! Division 1 xmits on MIDI chan 1
    GREAT: 2          ! Division 2 xmits on MIDI chan 2
    DUMMY: 4          ! dummy division assigned unused MIDI chan
    PEDAL: 3          ! Division 4 xmits on MIDI chan 3
```

```
! Third, declare the names of the pipe ranks. The order of the pipe ranks
! determines the associated PTM output. These can be rearranged and
! grouped as desired.
```

```
RANK NAMES =
    BOURDON 16'       ! PTM#1, RANK1 output, 97 pipes
    SPITZPRINCIPAL 4' ! PTM#1, RANK2 output, 73 pipes
    PRINCIPAL QUINT 2/3' ! PTM#2, RANK1 output, 42 pipes
```

! ----- example2.src ----- sht 2 of 4

! Fourth, assign the tremolo location and its MIDI control channel.
! There are two tremolo controls. They can be controlled by any MIDI
! channel and located on any rank.

TREMOLO ASSIGNMENT =

CONTROL 1: 5 ! Bn 01 01 = off; Bn 01 11 = on; n=4
RANK 1: SPITZPRINCIPAL 4' ! location = Principal 4' = PTM#1, RANK2
CONTROL 2: 5 ! Bn 01 01 = off; Bn 01 11 = on; n=4
RANK 2: PRINCIPAL QUINT 2/3' ! location = Principal 4' = PTM#2, RANK1

! Fifth, assign the expression location and its MIDI control channel.
! This is done as it was for tremolo, except that there is only one control
! and therefore no numeric suffix added to CONTROL and RANK.

EXPRESSION ASSIGNMENT =

CONTROL: 5 ! Bn 07 vv; vv = 0-127, n = 4
RANK: SPITZPRINCIPAL 4' ! location = Principal 4' = PTM#1, RANK2

! Finally, you begin the entry of stop definitions. This is done division
! by division. Each division can have 14 stops. Each stop can associate
! multiple ranks. Each rank can respond to six keyboard zones. Each zone
! can control three different pitches (or pipes).

! ----- SWELL STOPS -----

SWELL STOP 1 =

NAME: FLUTE 8'

RANK: BOURDON 16' ! 97 pipe rank
ZONE 1: C6 ! division control is all 61 keys
PITCH1: +12 ! play pipes 13-73 of this rank

SWELL STOP 2 =

NAME: FLUTE 4'

RANK: BOURDON 16' ! 97 pipes rank
ZONE 1: C6 ! division control is all 61 keys
PITCH1: +24 ! play pipes 25-85 of this rank

SWELL STOP 3 =

NAME: PRINCIPAL 2'

RANK: SPITZPRINCIPAL 4' ! 73 pipe rank
ZONE 1: C6 ! division control is all 61 keys
PITCH1: +12 ! play pipes 13-73 of this rank

! ----- example2.src ----- sht 3 of 4

! ----- GREAT STOPS -----

GREAT STOP 1 =

NAME: PRINCIPAL 8'

RANK: BOURDON 16' ! rank one of two, 97 pipes
ZONE 1: C6 ! division control is all 61 keys
PITCH1: +12 ! play pipes 13-73 of this rank

RANK: SPITZPRINCIPAL 4'
ZONE 1: C5
PITCH1: 0

! rank two of two, 73 pipes
! division control is lower 49 keys
! play pipes 1-49 of this rank

GREAT STOP 2 =
NAME: OCTAVE 4'

RANK: SPITZPRINCIPAL 4'
ZONE 1: C6
PITCH1: +12

! 73 pipe rank
! division control is all 61 keys
! play pipes 13-73 of this rank

GREAT STOP 3 =
NAME: FLUTE 2'

RANK: BOURDON 16'
ZONE 1: C6
PITCH1: +36

! 97 pipe rank
! division control is all 61 keys
! play pipes 37-97 of this rank

GREAT STOP 4 =

NAME: MIXTURE III

RANK: SPITZPRINCIPAL 4' ! rank one of two, 73 pipe rank
ZONE 1: F#3 ! division control is keys 1-31
PITCH1: +24 ! play pipes 25-55
PITCH2: +36 ! play pipes 37-67
ZONE 2: F4 ! division control is keys 32-42
PITCH1: +12 ! play pipes 44-54
PITCH2: +24 ! play pipes 56-66
ZONE 3: C5 ! division control is keys 43-49
PITCH1: +24 ! play pipes 57-73
PITCH2: +12 ! play pipes 55-61
PITCH3: 0 ! play pipes 43-49
ZONE 4: F5 ! division control is keys 49-54
PITCH2: +12 ! play pipes 61-76
PITCH3: +0 ! play pipes 49-54
ZONE 5: C6 ! division control is keys 55-61
PITCH1: +12 ! play pipes 67-73
PITCH2: 0 ! play pipes 55-61
RANK: PRINCIPAL QUINT 2/3' ! rank two of two, 42 pipe rank
ZONE 1: B1 ! division control is 1-12
PITCH1: +12 ! play pipes 13-24
PITCH2: +0 ! play pipes 1-12
ZONE 2: F3 ! division control is 13-30
PITCH1: +0 ! play pipes 13-30
ZONE 3: C5 ! division control is 31-49
PITCH1: +0 ! play pipes 31-49
PITCH2: -12 ! play pipes 19-37
ZONE 4: C6 ! division control is 50-61
PITCH1: 0 ! play pipes 50-61
PITCH2: -12 ! play pipes 38-49
PITCH3: -27 ! play pipes 13-34

! ----- PEDAL STOPS -----

PEDAL STOP 1 =

NAME: GEDECKT 16'

RANK: BOURDON 16' ! 97 pipe rank
ZONE 1: G3 ! division control is 32 pedals
PITCH1: 0 ! play pipes 1-32 of this rank

INTERMIDI, INC.

8 AM to 6 PM M-F, PST

PHONE 1-509-427-7999

OR

FAX 1-509-427-7277