INTERSIL

IM6100 CMOS FAMILY SAMPLER

CONTENTS

Introduction .................................................. 1
Power ON Procedure ........................................... 4
Functional Description ....................................... 4
20mA Current Loop Interface ................................. 5
RS232C Interface ............................................... 7
IM 6402 UART Timing ........................................ 8
Single Clock .................................................... 11
Single Instruction ............................................. 12
Parallel Data I/O .............................................. 14
Appendix A, Octal Debugging Technique ................. 15
Appendix B, (Deleted) .........................................
Appendix C, 6960—Sampler Board .......................... 32
Appendix D, Teletype Modifications ....................... 35
INTRODUCTION

An all-CMOS IM6100 based microprocessor system with 256 X 12 random access memory, 1K X 12 read only memory and a serial interface port can be fabricated utilizing seven (7) CMOS LSI devices.

The IM6100 is a single chip, 12-bit microprocessor implemented in silicon gate complementary metal oxide semiconductor (SiG CMOS) technology. It recognizes the instruction set of the Digital Equipment Corporation PDP-8 minicomputer.

The IM6561 is a high speed, low power silicon gate CMOS 1024 bit static RAM organized 256 words by 4 bits. Data-In and Data-Out are multiplexed on the same pins.

The IM6312 is a 1K X 12 mask programmable CMOS ROM with address and data multiplexed on the same pins. The IM6312-001 is 'masked' at the factory with the ODT program (Appendix A). This program allows the user to enter user programs into the RAM via the Teletype and then to edit and execute them.

A Parallel Interface Element, IM6101, provides the universal means of interfacing peripheral devices to the IM6100. It is programmable and configured by the IM6100 for a specific interface during system initialization.

The Universal Asynchronous Receiver/Transmitter, IM6402/03, is a CMOS LSI programmable subsystem for interfacing processors to an asynchronous data channel.

Additional requirements to enable the system to be fully operational are a single 5 volt power supply, two crystals, a 3.3 MHz crystal for the microprocessor and a 3.60 MHz crystal for a 110 baud serial port, control logic to reset and run the system and serial I/O level shifters to communicate with a Teletype or other ASCII terminals (Figure 1).

The dynamic power dissipation of the system (excluding that of the serial I/O level shifters) will be less than 50 mW at 4 MHz and 5 volts.
Fig. 1 — System Hookup Diagram
POWER ON PROCEDURE

1. Connect up the CMOS/LSI devices as shown in Figure 1.

2. Choose the level shifting network for the serial I/O device (for example a 20 mA current loop for Teletype) and make the appropriate connections.

3. Turn the power on. Press and release RESET pushbutton. Put the Teletype "ON LINE".

4. Press and release RUN/HLT pushbutton. The ODT program (Appendix A) responds with a CARRIAGE RETURN/LINE FEED and then waits for user commands.

FUNCTIONAL DESCRIPTION

The IM6312-001 ROM is mask programmed to be enabled when DX(0) = DX(1) = 1 at the falling edge of LXMAR to respond to addresses 6000-77778.

The RSEL output defines an area in the 4096 word addressing space dedicated to RAM. The IM6312-001 RSEL is programmed to be enabled when DX(0) = DX(1) = 0 at the falling edge of LXMAR to respond to addresses 0000-17778. If only 256 words of RAM are used, DX(2) and DX(3) bits are ignored during addressing. For 1K word RAM systems, DX(2) and DX(3) become the high order address bits.

The IM6402/03 UART is hardwired for eight data bits, two stop bits and no parity. If the IM6403 is used, the eleven stage divider chain is selected. This on-chip divider and oscillator allows an inexpensive crystal to be used as a timing source for the IM6403 UART. UARTs require the clocks to be 16X the bit rate (baud). For Teletype operation, it is ten characters per second or 110 baud. Each character consists of eleven bits (one start bit + eight data bits + two stop bits). The UART clock frequency must be 1.76 KHz (110 X 16). If a standard color TV crystal of 3.579545 MHz is used in conjunction with the eleven stage divider chain, one obtains a baud rate of 109.2 (3.579545 X 10^6 ÷ 211 ÷ 16) which is well within the tolerance limits for Teletype operation. (The crystal frequency should be 3.6044 MHz for 110 baud.) The same type of crystal can also be used to clock the IM6100. Refer to the section on UART timing for the circuits to be used for the IM6402.
The PIE is wired to respond to the select code 01110 on SEL3-SEL7 and to inhibit priority vectoring. The PIE control registers CRA and CRB are programmed by the ODT to be 0000b and 0060b, respectively. The PIE control signals are then defined as shown below:

- **WRITE1 and WRITE2** Active Low
- **SENSE1 and SENSE2** Active on 0 to 1 transition
- **SENSE3 and SENSE4** Active on 1 to 0 transition
- **FLAG1, FLAG2, FLAG3 and FLAG4** Initialized Low

Refer to the IM6100, IM6101, IM6312, IM6402/03 and IM6551/61 data sheets for detailed descriptions of the CMOS/LSI devices used in the IM6100 CMOS FAMILY SAMPLER.

The RESET pushbutton is for system initialization. When the IM6100 is reset, it halts with the Program Counter set to 7777b. The RUN/HLT pushbutton is used to alternatively run and halt the CPU. C1 and SKP lines are pulled up to VCC with 2.7K resistors, since these lines are driven by open drain outputs from the PIE.

The section on level shifting networks details the circuits for 20 mA current loop and RS232 serial I/O interface. User modifications that may be required to select 20 mA current loop and full duplex operation for Teletype are described in Application Bulletin M006: "Teletype Interface for the IM6100".

**20 mA CURRENT LOOP INTERFACE**

Figures 2-A, 2-B and 2-C describe the discrete circuits for 20 mA current loops. For reliable Teletype operation, the receive current loop must be established by a voltage source of 15 volts or more.

FLAG1 output of the PIE can be used to control the Reader Relay, if one is installed. Note that the ODT program initializes FLAG1 to be 0 and, therefore, the reader relay to be OFF. The user program must explicitly control the relay by programming FLAG1 output.

The pin connections to a 20 mil 10 pin header are shown.
20mA Current Loop Interfaces For Teletype

Fig. 2A — XMT Loop

Fig. 2B — RCVE Loop

Fig. 2C — RDR RUN Loop
RS232C INTERFACE

The level shifting networks to conform to the RS232C formats are shown in Figures 3-A and 3-B. The pin connections shown are for an AMP 205858-1 connector as used on the 6960-SAMPLR Board.

![Fig. 3A](image)

![Fig. 3B](image)
IM6402 UART TIMING

ICM7213

ICM7213 is a fully integrated oscillator and frequency divider with buffered outputs. The circuit in Figure 4-A divides the crystal frequency by 211 to generate the 16X clocks for the IM6402.

Fig. 4A—IM6402 UART Timing with ICM 7213

555 TIMER

The analog timer circuit shown in Figure 4-B can be used for the UART clocks. The resistors and capacitors in the timing circuit must be chosen to be temperature stable.

Fig. 4B—IM6402 UART Timing with 555
BAUD RATE GENERATOR

The 34702 CMOS Baud Rate Generator circuit can be programmed via $S_0$, $S_1$, $S_2$ and $S_3$ select lines to generate 13 of the most commonly used baud rates.

![Diagram of 34702 CMOS Baud Rate Generator circuit](image)

**XTAL** 2.4576 MHz

**Fig. 4C—IM6402 UART Timing with Baud Rate Generator**

<table>
<thead>
<tr>
<th>XTAL 2.4576 MHz</th>
<th>$S_3$</th>
<th>$S_2$</th>
<th>$S_1$</th>
<th>$S_0$</th>
<th>Baud Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>50</td>
</tr>
<tr>
<td>0.1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>75</td>
</tr>
<tr>
<td>0.0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>134.5</td>
</tr>
<tr>
<td>0.0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>200</td>
</tr>
<tr>
<td>0.1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>600</td>
</tr>
<tr>
<td>0.0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2400</td>
</tr>
<tr>
<td>0.0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>9600</td>
</tr>
<tr>
<td>0.1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4800</td>
</tr>
<tr>
<td>0.1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1800</td>
</tr>
<tr>
<td>0.1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1200</td>
</tr>
<tr>
<td>0.1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2400</td>
</tr>
<tr>
<td>0.1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>300</td>
</tr>
<tr>
<td>0.1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>150</td>
</tr>
<tr>
<td>0.1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>110</td>
</tr>
</tbody>
</table>
ICM7209

An 8-stage divider chain along with the oscillator and 3-stage divider circuit in the ICM7209 with the appropriate XTAL frequency (Figure 4-D) can generate some of the more commonly used baud rates.

<table>
<thead>
<tr>
<th>XTAL MHz</th>
<th>DIVIDER STAGE</th>
<th>BAUD RATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.6044</td>
<td>$2^8$</td>
<td>110</td>
</tr>
<tr>
<td>3.5795</td>
<td>$2^8$</td>
<td>109.2</td>
</tr>
<tr>
<td>2.4576</td>
<td>$2^6$</td>
<td>300</td>
</tr>
<tr>
<td></td>
<td>$2^5$</td>
<td>600</td>
</tr>
<tr>
<td></td>
<td>$2^4$</td>
<td>1200</td>
</tr>
<tr>
<td></td>
<td>$2^3$</td>
<td>2400</td>
</tr>
<tr>
<td></td>
<td>$2^2$</td>
<td>4800</td>
</tr>
<tr>
<td></td>
<td>$2^1$</td>
<td>9600</td>
</tr>
</tbody>
</table>

Fig. 4D—IM6402 UART Timing with ICM7209
SINGLE CLOCK

Since the IM6100 family of devices are static, the system may be single clocked. Figure 5 shows the circuit that stops the free running oscillator and introduces manually generated single clocks. ICM7209 is used as a buffered oscillator. The SC EN SW, Single Clock ENable, selects the ICM7209 output or the single clock as the input to the IM6100. The 74C74 is used to ensure integral clocking and to avoid clock slivers while switching from one mode to another. The single clocks are generated by the debounced SC, Single Clock, pushbutton.

Fig. 5—Single Clock
SINGLE INSTRUCTION

The IM6100 can be controlled to execute one instruction at a time using the circuit in Figure 6-A. When the SI EN SW, Single Instruction ENable, is in the free run mode, the WAIT signal is high, and the normal operation of the processor is not affected. When the SI EN SW is in the single instruction mode, the WAIT line is pulled down during IFETCH cycle, pausing the IM6100 in the instruction read state. Activating the SI pushbutton will clear the WAIT line, and the processor will go on to execute the instruction and then pause while reading the next instruction and so on.

![Diagram of IM6100 circuit](image)

Fig. 6A—Single Instruction

The simple display circuit in Figure 6-B can be used to display instruction addresses, or the state of the DX bus. If the DX EN SW is in the INSTR ADDRESS mode, the quad latches are clocked by IFETCH*LXMAR to display the instruction address. If the DX EN SW is in the DX mode, the DX data flows through the latch.

In the single clock mode, it is useful to have the DX EN SW in the DX state to read the state of the DX bus at every clock.

In the single instruction mode, first set the DX EN SW in the INSTR ADDR mode. The LEDs will then display the address of the instruction about to be executed. Then switch the DX EN SW to the
DX mode, and the display will then show the instruction that is about to be executed since the DX lines contain the instruction as the IM6100 is 'paused' in the read state. Flip the DX EN switch to the INSTR ADDR state before activating the SI push-button for the next instruction and so on to step through instructions, one at a time.

![Diagram of circuit with labels and connections]

**Fig. 6B—Program Counter and DX bus display**

![Diagram of LED display]
PARALLEL DATA I/O

The unused WRITE2 and READ2 control signals of the PIE may be used to implement the parallel data I/O circuit shown in Figure 7.

When the user program executes a READ2 instruction, the 80C95 tri-state buffers enable the data on the DX bus. The DX information can be latched in the 74C174 hex DFFs by executing a WRITE2 instruction. Note that the WRITE2 is programmed to be active low by ODT, and the data is latched by the second edge of the write pulse.

SENSE3-4 and FLAG 1-4 of the PIE are also available to the user.

---

**Fig. 7—Parallel Data I/O**
APPENDIX A

1. IM6312-001 CMOS OCTAL DEBUGGING TECHNIQUE (ODT) ROM

INTRODUCTION

This describes the use of the Intersil Octal Debugging Technique (ODT) program.

The commands for Intersil ODT are very similar to those for Digital Equipment Corporation's ODT for use on the PDP-8 in particular, and the DDT family of programs in general.

RESERVED LOCATIONS

The RAM locations in page zero with octal addresses 5, 6 and 20 through 77 are reserved for use by the ODT program. User programs should not modify these locations.

COMMANDS

ODT commands consist of a control character or an octal number followed by a control character. All commands are echoed, that is, each character is printed as it is typed in. Octal numbers may consist of octal digits only (no 8's or 9's) and may be from one to four digits long.

Note that ODT does not print a special prompt character (such as an asterisk) before commands may be typed, and does not require that commands be terminated by a special character, such as carriage return (in fact, carriage return is a special command). Commands may be typed in any time the Teletype is idle and are executed as soon as the control character is typed.

BINARY LOAD COMMAND

L - Load from the tape reader
Typing an L will load BIN tape (ignoring change field characters) from a reader. To use command, type L after a prompt, place tape in reader on the leader-trailer part, then start the tape reader. The BIN tape will be read into the current field and the checksum will be printed out on the Teletype following the end of the load. This should be 0000 for a proper load.

EXAMINE/MODIFY COMMANDS

/ (slash) - opens a location
Typing an octal number nnnn followed by a slash causes the location whose address is nnnn to be opened. When a location is opened, its content is printed out as an octal number. Typing a slash not preceded by a number causes the most recently opened location to be reopened.
(carriage return) - closes a location
When a location is open, typing an octal number, nnnn, followed by a carriage return causes the contents of the location to be changed to the number nnnn and closes the location. Once a location is closed, its contents cannot be changed without re-opening the location. Typing a carriage return not preceded by a number causes the location to be closed without modifying its contents. If a carriage return (possibly preceded by a number) is typed when no location is open, it has no effect.

(line feed) - closes and opens next
When a location is open, typing a line feed causes the location to be closed and the next memory location (that with an address one higher than the current location) to be opened. The address of the new location will be typed out, followed by a slash, followed by the contents of the new location. The effect is the same as if the user had typed a carriage return, followed by typing the address of the next location and the slash. Typing an octal number, nnnn, before typing the line feed causes the contents of the old location to be changed to nnnn. Then the old location is closed and the next location is opened, as described above.

+(back arrow) - closes location and opens indirect reference
When a Location is open, typing a back arrow causes the location to be closed. The contents of the location are then treated as an indirect reference. That is, the content of the old location is taken as an address, and the new location is opened. As with the line feed command, the address of the newly opened location is typed, followed by a slash, and the effect is the same as if the user had typed the address and slash. If while a location is open, an octal number, nnnn, is typed followed by a back arrow, the content of the open location is changed to nnnn and proceeds as above. Note in this case that the address of the new location opened will be the same as the number just typed in. On most Teletypes, the back arrow character is the same as shift 0.
† (up arrow) - closes location and opens memory reference
This command behaves identically to the back arrow command except that the contents of the location are treated as a memory reference instruction, and it is the location referenced by that instruction that is opened. The location opened is that immediately referenced by the instruction. If the instruction is indirect (bit 3 is set to 1), then typing the up arrow only opens the location containing the pointer to the operand of the instruction. To open the effective location referred to by an indirect instruction, type an up arrow (memory reference) followed by a back arrow (indirection). On most Teletypes, the up arrow character is the same as shift N.

Example

Say that the simple program

```
200   7200   CLA
201   1604   TAD I X
202   2204   ISZ X
203   7402   HLT
204  0205   X, Y
205   0000   Y, 0
```

is stored in memory. Then the following might be the result of a session with ODT. (Note: The underlined portion is typed by the user, and the remainder is typed by the computer. The symbol CR stands for carriage return, and LF stands for line feed.)

```
200/7200 LF
0201/1604 LF
0202/2204 LF
0203/7402 LF
0204/0205 LF
0205/0000 CR
200/7200 7201 CR
/7201 LF
0201/1604 †
0204/0205 ”
0205/0000 CR
202/2204 2205 LF
0203/7402
```

list the program in octal
change CLA to CLA IAC
verify the change
find the location referenced by TAD I X
change ISZ X to ISZ Y
PROGRAM CONTROL AND BREAKPOINT COMMANDS

G - go to
Typing an octal number, nnnn, followed by a G causes ODT to begin executing the program stored in memory, starting at location nnnn.

B - breakpoint
A breakpoint should be set at some location that will be executed by the user's program, upon the execution of which he wishes control to return to ODT. Typing an octal number, nnnn, followed by a B causes ODT to set a breakpoint at location nnnn. Only one breakpoint can be set at a time, therefore, setting a breakpoint clears the previous breakpoint. Typing a B without preceding it by a number causes the current breakpoint to be cleared. When execution of a program with a breakpoint resumes (via the G or C commands) the content of the breakpoint location is replaced with a special trap instruction that returns control to ODT. When control returns to ODT, the trap instruction is replaced by the original contents of the location. Then ODT prints out the location at which the trap occurred followed by a left parenthesis followed by the contents of the accumulator when the trap occurred. ODT then waits for the user to type in a new command.

C - continue
After a breakpoint causes control to return to ODT from a user program, typing C causes the program to resume execution where it left off. Execution will resume from the location that the program left off, even if the breakpoint was reset to a new location. If an octal number, nnnn, is typed before the C, then after the resumed program has executed the breakpoint once it will continue execution (rather than returning control to ODT) until the breakpoint has been executed nnnn more times; then control returns to ODT just as it does after a normal breakpoint.

A - examine/modify accumulator, link, MQ
Three consecutive ODT RAM locations are reserved for storing the contents of the AC, link and MQ registers when a breakpoint occurs. When execution of the user's program resumes (via the G or C command), the contents of these registers are restored from these locations. Typing A causes the first of these locations, containing the contents of the AC, to be opened. Following this by a line feed opens the next location, containing the contents of the link (either a one or a zero). Following this by a line feed opens the last location, containing the contents of the MQ, to be opened. All of the commands of the previous section (/,
\(+, -, \text{carriage return, line feed})\) can be used to examine and modify these registers. If one of the registers, AC, is changed during the breakpoint, then the AC will have the new value when program execution resumes. This can be used to change the contents of the AC. The same facility can also be used to set the initial contents of the registers before the first G command begins the program's execution.

**Example**

Say that the simple program

\[
\begin{align*}
300 & \quad 7001 \quad \text{START, IAC} \\
301 & \quad 7440 \quad \text{SZA} \\
302 & \quad 5300 \quad \text{JMP START} \\
303 & \quad 7402 \quad \text{HLT}
\end{align*}
\]

is stored in memory. Then the following might be the result of a session with ODT.

```
A1764 OFL  accumulator contains garbage, clear it
0050 /0001 OCR  same for link
302B  set breakpoint at JMP START
300G  execute program
0302 (0001  breakpoint occurs; accumulator has been incremented
breakpoint occurs
7774C Go past breakpoint 1 + 7774 times
0302 (7776 reset breakpoint to HLT instruction
303B breakpoint occurs
C program stops when AC reaches 0 again
0303 (0000 link has been changed by overflow
A000OLF clear all breakpoints
0050 /0001 CR B
```

**WORD SEARCH COMMANDS**

**M - open search mask, lower_bound, upper_bound**

The mask, lower bound and upper bound for word searches are kept in that order in three consecutive reserved ODT locations. The first of these locations, the mask, can be opened by typing M. This will cause the current value of the mask to be printed out. The user can enter a new value by typing it followed by any of the word modifying commands (carriage return, line feed, up arrow or back arrow). Typing a line feed causes the next location, containing the lower search bound, to be opened. Its contents will be printed out and it can be modified in the same way. Typing a line feed again causes the final location, containing
the upper search bound, to be opened. Initially, the value of all three locations is unspecified. The user should set these locations to the desired value before giving the first \textit{W} command.

\textbf{W} - word search command

Typing an octal number, \textit{nunn}, followed by a \textit{W} causes a word search to occur. The search proceeds as follows: The number, \textit{nunn}, that was typed is masked and remembered as the quantity which is being searched for. (The operation of masking is to take the bitwise boolean AND of the given word with the contents of the mask word.) Then each location, beginning with the location whose address is stored in the lower bound word, is masked and compared with the quantity being searched for. If the two are equal, then the address of the word, followed by a slash and the (unmasked) contents of the word are printed out. Then the next location is examined and so on until (and including) the location whose address is stored in the upper bound word is reached. The word search command does not change the contents of any word in the user's programs.

\textbf{Example}

Say that a program is stored between locations 200 and 377 in memory. Then typing in

\begin{verbatim}
M0000 7000LF
0054/0000 200LF
0055/0000 377CR
1000W
\end{verbatim}

causes all TAD instructions in the program (those words beginning with the octal digit 1) to be printed out, for example

\begin{verbatim}
0210 /1305
0265 /1305
0354 /1711
\end{verbatim}

if these were the only words in the program containing TAD instructions. Then typing in

\begin{verbatim}
M7000 7777CR
\end{verbatim}

\textit{OW}

would cause all words in the program whose contents were exactly 0000 to be printed out.
Typing in

M7777 OLF  
0054 /0200 250LF  
0055 /0377 300CR  
0W

will cause the contents of all words whose addresses are  
between 250 and 300 (inclusive) to be printed out.

TAPE PUNCHING COMMANDS

The following commands can be used to punch out paper  
tapes that can be read in by the BIN loader.

T - punch leader/trailer
Typing a T will cause about four inches of leader/trailer  
tape (tape punched with 200 octal) to be punched. The T  
command also causes the accumulated checksum to be set  
to zero (cleared). The tape punch should be turned on  
immediately after the T command is given, and turned off  
when the Teletype stops punching. Note that as long as  
the punch is turned on, anything (including commands)  
typed into the Teletype will be punched onto the tape, so  
the punch should always be turned off before giving a  
command.

P - punch tape
Typing an octal number, nnnn, followed by a semi-colon  
(;) followed by a second octal number, mmmm, followed  
by a P, causes a tape corresponding to the contents of  
the block of memory beginning at location nnnn and  
ending at location mmmm to be punched. No checksum is  
punched at the end of the block so that several blocks can  
be punched together with one inclusive checksum. The  
punch should be off when the P command is given (otherwise  
the P command will be punched on the tape, causing errors  
when the tape is read). ODT halts before punching the  
tape to give the user time to turn on the punch. ODT is  
restarted by pressing the RUN/HLT button. Thus, the  
sequence of actions is: Make sure the punch is off; give  
P command; ODT halts; turn on punch; press RUN/HLT switch;  
ODT punches tape; turn off punch.

E - punch checksum and trailer
Typing an E will cause the accumulated checksum to be  
punched, followed by about four inches of leader/trailer tape.  
The checksum is also reset to zero (cleared). Like the P  
command, ODT halts after the command is typed to give the  
user time to turn on the punch. The sequence of actions  
to use is: Make sure punch is off; give E command; ODT
halts; turn on punch; press continue switch; ODT punches checksum and trailer; turn off punch.

Example

Say that the user program occupies locations 100 through 170 and 200 through 377 of memory. To punch out the program, the user should give the following commands to ODT:

```
T
T00;170P
T200;377P
E
```

ERRORS

If an error occurs while typing a command, ODT types "?", ignores the erroneous command and waits for a new one. ODT detects that an error occurred if the user types more than four digits for an octal number or if a decimal digit (8 or 9) is typed. It also detects an error if some character other than a command character (/; carriage return, line feed, +, -, G, B, C, A, M, W, T, ;, P, E) is typed. ODT does not detect errors such as leaving the punch on while typing a command or giving a numerical argument for a command that should have none. For example, 100A is not detected as an error and is treated the same as typing A. If the user makes a mistake while typing a number, he can abort the command by purposely creating an error by typing too many digits or an illegal character.

Example

Here are some errors in ODT commands.

<table>
<thead>
<tr>
<th>Command</th>
<th>Error Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>12345?</td>
<td>too many digits</td>
</tr>
<tr>
<td>248?</td>
<td>decimal digit</td>
</tr>
<tr>
<td>101x?</td>
<td>illegal character</td>
</tr>
</tbody>
</table>

FIELD COMMANDS

Intersil ODT also contains commands for working with multiple fields. These commands have no effect in the SAMPLER system, which does not contain hardware for handling fields.

The additional command symbols are period (.) and Q. Hence, these are not illegal symbols and typing them does not cause ODT to recognize an error, but it may cause ODT to act strangely. If the user types one of these in by mistake, and ODT acts strangely, he should restart ODT.

Occasionally, ODT may print out 0. (a zero followed by a period) on the terminal. This just means that ODT thinks that you are operating in field zero (which you are) and may be ignored.

INTERRUPTS DURING ODT

The ODT program does not use any interrupt features. It does not disable the interrupt system if it is enabled by the user. The user must be cautious when enabling interrupt requests while the user program is 'controlled' by ODT via break points.
USEFUL SUBPROGRAMS

Some of the subprograms in the ODT ROM are general subroutines that the user may find it useful to call from his own program. These are described below.

CALLING AND RETURNING FROM A SUBROUTINE

For programs designed to exist in ROM, the JMS instruction cannot be used to call a subroutine because the first word of the routine cannot be used to store the return address. Instead, the ODT program makes use of a stack of return addresses kept in RAM by means of special call and return routines. (Refer to Applications Note M008: ROM Based Subroutine Calls With The IM6100.)

The user may make use of the ODT subroutine stack mechanism as follows. To call a routine requires two consecutive words; the octal number 4056 should be stored in the first word, and the address of the subroutine to be called should be in the second. Executing this piece of code causes the return address to be pushed onto the top of the stack and for control to be transferred to the given address. The AC, link and MQ are unaffected by executing the call instruction.

To return from the subroutine, execute a location containing the octal number 5461. This causes a return address to be taken from the top of the stack and returns control to that location. The AC, link and MQ are again unaffected.

The return address stack begins in location 75 and grows upwards (i.e. the first address pushed goes into location 75, the next into location 76, etc.) ODT may nest subroutines up to three deep, therefore, three stack locations in addition to what the user program needs should be reserved for use by ODT.

The stack pointer is stored in location 62, that is, location 62 contains the address of the location where the most recently pushed return address is stored.

IMPORTANT NOTE: The read and change commands in ODT involve calling a subroutine. Hence, giving the command "62/" will print out a value one greater than the true value of the stack pointer.

The user must be cautious when using the software stack for servicing interrupt requests since the software stack routine saves AC in a temporary location which could get overwritten by calling the stack routine again by an interrupt service routine before the first routine could exit.
TYPE SUBROUTINE

The TYPE subroutine prints on the Teletype the character whose ASCII code is stored in the AC. It also clears the AC before returning.

The subroutine may be called by using the call procedure described in the previous section. The starting address is 6102. Allocate one position on the stack for its use.

PNUM SUBROUTINE

The PNUM subroutine prints on the Teletype the contents of the AC as a four digit octal number followed by a space. It also clears the AC before returning. The starting address is 6107.

The subroutine may be called by using the call procedure described above. It makes use of the TYPE routine described above, therefore, two stack positions should be allocated for its use.

Example

Here is a subroutine that counts from 0000 to 7777 octal, printing each number followed by a space and an asterisk on the Teletype.

\[
\begin{align*}
\text{CALL} & = 4056 \\
\text{RETURN} & = 5461 \\
\text{TYPE} & = 6102 \\
\text{PNUM} & = 6107 \\
\end{align*}
\]

200 7200 COUNT, CLA /set count to zero
201 3214 DCA X /count stored in X
202 1214 LOOP, TAD X /get count
203 4056 CALL
204 6107 PNUM /print it and space
205 1213 TAD ASTRSK
206 4056 CALL
207 6102 TYPE /print asterisk
210 2214 ISZ X /increment count
211 5202 JMP LOOP /repeat loop
212 5461 RETURN /return
213 0252 ASTRSK, 252 /ASCII for asterisk
214 0000 X, 0 /count stored here, this must be a RAM location (all others can be ROM)
Three stack locations should be reserved for calling COUNT, one for the return address from COUNT itself, and two for using the PNUM subroutine. A stack location need not be allocated for calling TYPE because the two locations required for PNUM are unused when it is called. (In general, for each subroutine reserve one location plus the maximum number of locations required by all the subroutines called by it.)

Since three stack locations should also be reserved for ODT and the stack begins in location 75, locations 75 through 102 should be reserved for use by the stack.

PROGRAM LISTING:
APPENDIX C
INTERSIL 6960-SAMPLR BOARD

NOTE: 1. Square pads are used for pin 1 of all integrated circuits, the emitter of all transistors, the cathode of all diodes, and the common terminal of all switches.

2. Push button switches are assumed to have common as one end terminal, the center terminal normally open and the other end terminal normally closed (C & k type 8121). Toggle switches are assumed to have common in the center (C & k type 7101).

JUMPER OPTIONS

J1 C0 (6100) to VCC
J2 C2 (6100) to VCC
J3 INTREQ (6100) to VCC
J4 CPREQ (6100) to VCC
J5 DMAREQ (6100) to VCC
J6 WAIT (6100) to VCC
J7 (Deleted)
J8 SENSE4 (6101) to GND
J9 PRIN (6101) to GND
J10 INTGNT (6101) to GND
J11 SEL4 (6101) to GND
J12 SEL4 (6101) to VCC
J13 SEL7 (6101) to VCC
J14 SEL7 (6101) to GND
J15 Deleted
J16 Deleted
J17 OSC IN (6100) to pin 3 IC35 (74C00)
J18 OSC OUT (6100) to pin 11 IC35 (74C00)
J19 TRC (6402) to pin 2 (7213)
J20 RRC (6402) to pin 2 (7213)
J21 RESET (6100) to pin 1 IC26 (74C74)
J22 XTB (6100) to S16 common
J23 WAIT (6100) to pin 6 IC26 (74C74)
J24 IFETCH (6100) to pin 1 IC16 (74C00)
J25 TRO (6402/03) to RS-232-C XMT level shifter
J26 TRO (6402/03) to 20 mA loop XMT level shifter
J27 GND to 20 mA loop XMT level shifter
J28 RRI (6402/03) to RS-232-C RCVE level shifter
J29 RRI (6402/03) to 20 mA loop RCVE level shifter
J30 FLAG1 (6101) to 20 mA RDR RUN loop driver
J31 GND to -12V point of 20 mA loop interface
J32 -12V terminal to -12V point of 20 mA loop interface
J33 Pin 1 of EIA connector to GND
JUMPER STRAPPING

OPTION 1 - BASIC MICROCOMPUTER (See Figure 1)
Insert the following jumpers: J1, J2, J3, J4, J5, J6, J8, J9, J10, J11, J13

Note: J11 and J12 should NEVER be in circuit at the same time.
Similarly, J13 and J14 should NEVER be in circuit at the same time.
If both jumpers were in, a direct short circuit between VCC and GND would result.
XTAL 11 and XTAL 12 should be used.

OPTION 2 - 20 mA CURRENT LOOP INTERFACE (See Figure 2)
Insert jumpers J26 and J29

OPTION 3 - PROGRAM CONTROLLED TELETYPewriter PAPER TAPE READER
Insert jumper J30.

OPTION 4 - EIA RS-232-C INTERFACE (See Figure 3)
Insert jumpers J25, J28, J27.

Note: Options 2 and 4 are mutually exclusive, so insert and delete jumpers as necessary.

OPTION 5 - IM6402 UART TIMING (See Figure 4)
Use XTAL14 (delete XTAL21), insert jumpers J19 and J20

OPTION 6 - SINGLE CLOCK OPERATION (See Figure 5)
Use XTAL15 (delete XTAL11), insert jumpers J17 and J18

OPTION 7 - SINGLE STEPPING OPERATION (See Figure 6)
Delete jumper J6 and add jumpers J21, J22, J23 and J24
APPENDIX C
6960 SAMPLR BOARD
APPENDIX D
TELETYPERS MODIFICATIONS FOR THE INTERCEPT SYSTEM

The Intersil INTERCEPT systems have been designed to be used in conjunction with a Model ASR-33 Teletype. Before attempting to use your system inspect your Teletype for the following modifications and additions. If they have not yet been performed, you must complete them before using INTERCEPT.

To check for, or make, these modifications remove the cover of the Teletype. Loosen the three thumb screws in the back and remove the Platen that holds the roll of paper, the Mode Switch knob and the Face Plate. Remove the small screw on the Reader cover and the four screws under the Face Plate. You should now be able to lift the cover off. Use Figure D-1 to locate the various parts located below.

![Figure D-1]

The modifications are:

CURRENT LOOPS CHANGED FROM 60 TO 20 MILLIAMPS

The Current Source Resistor must be changed from 750 ohms to 1450 ohms. This is accomplished by moving the BLUE wire from Terminal #3 to Terminal #4 of the large power resistor shown in Figure D-2. The receiver current level is changed by moving the PURPLE wire of Terminal #8 on Terminal Strip 151411 to Terminal #9 on the same strip. Terminal Strip 151411 is shown in Figure D-3 with Terminal #1 at the far left.

TELETYPER WIRED FOR FULL DUPLEX OPERATION

The half duplex wiring must be changed by moving the BROWN/YELLOW wire from Terminal #3 to Terminal #5 and the WHITE/BLUE wire from Terminal #4 to Terminal #5 on Terminal Strip 151411.

THE READER RUN RELAY ADDED

The Reader circuit should have a 12 volt relay inserted to allow program control of the Reader. This Relay is shown along with the mode switch in Figure D-4. Mount the Relay with two 6-32 screws on the available bracket. A schematic diagram for the Relay and its connections is shown in Figure D-6. Locate the BROWN wire coming from the Distributor Trip Magnet which is connected to terminal J4—Pin 11 as shown in Figure D-5. Cut this BROWN wire and connect to the wire marked BROWN on the Relay circuit (note that this leaves J4—Pin 11 with no connection). Connect the wire marked LINE to terminal L1 and the wire marked LOCAL to terminal N of the mode switch as in Figure D-6. A preassembled Reader Relay Card is available from Intersil Inc., Model # 6908-RELAY.

LEVEL 8 OPTION WIRED TO ‘ALWAYS MARK’

The level 8 option must be changed from parity to ‘ALWAYS MARK’. This causes the keyboard to always output a 1 for the 8th bit, and the Reader to read the 8th bit as it was written. Locate the Left Contact Block and the Right Contact Block as shown in Figure D-7. It may be necessary to remove a clear plastic shield to gain access to the Left Contact Block. On the Left Contact Block remove the RED/GREEN wire from the upper left contact, leave the RED/GREEN wire open and connect the GREEN wire to the upper left contact. On the Right Contact Block connect the GREEN wire to the upper left contact. For a detailed reference see Teletype keyboard schematic 9334WD.

CONNECT CPUS TT OUTPUTS TO THE TELETYPER

The TTY outputs of the CPUS TT board are connected to Terminal Strip 151411 and the relay as shown in Figures D-6 and D-8.
FIGURE D-2
Current Loop Resistor

FIGURE D-3
Terminal Strip

FIGURE D-4
Relay Card

FIGURE D-5
Distributor Trip Magnet

FIGURE D-6
Reader Relay Circuit
The following changes will be required in order to use the new ROM
ODT : IM6312-001 marked 63S003 ( in place of 63S000 ):

**Figure 1 page 2:** "SEL 4" (pin 9) should be shown to ground, not +5V.  
"SEL 7" (pin 14) should be shown to +5V, not ground.

**Page 5:** The first paragraph should read, "The PIE is wired to  
respond to the select code 00111 on SEL3 - SEL7 ...".

**Page 6:** The "pin-out" on the 74CO0 should read as follows:

```
8 10
12 9
```

**Page 33:** OPTION 1, "Note: Should read, "XTAL 11 and XTAL 21  
should be used."

**Note:** The description of the jumpers used, shown on page 33,  
is correct.