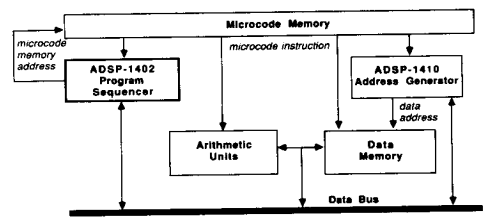
Word-Slice Program Sequencer

ADSP-1402

FEATURES
- **16-Bit Microcode Addressing Capability**
- **Look-Ahead™ Pipeline**
- **Extensive Interrupt Processing with Eleven On-Chip Interrupt Vectors**
- **Four Event Counters to Support Looping**
- **Absolute, Relative and Indirect Addressing**
- **50ns Cycle Time**
- **64-Word RAM for Storing:**
  - **Subroutine Linkage**
  - **Jump Addresses**
  - **Counters**
  - **Status Register**
- **1μm CMOS Technology**
- **84-Pin Grid Array Package**

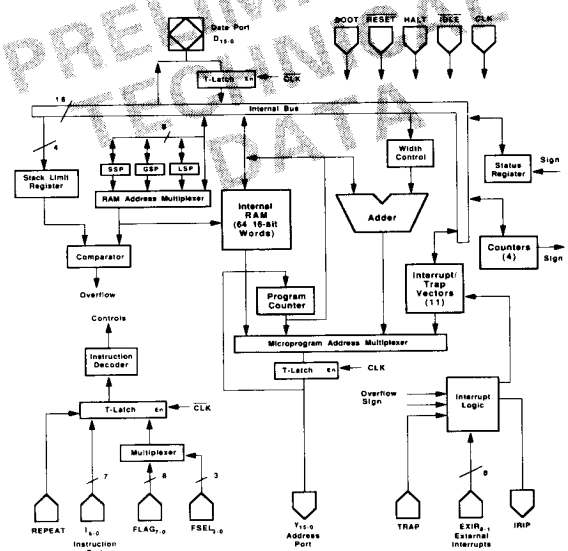


Word-Slice® Microcoded System with ADSP-1402

3

GENERAL DESCRIPTION

The ADSP-1402 Program Sequencer is an instruction-compatible upgrade to the ADSP-1401. It can be used with high speed arithmetic units and provides many features to simplify the design of microcoded systems. Among the devices it supports are the ADSP-3212 Floating-Point Multiplier, the ADSP-3222 Floating-Point ALU and the ADSP-3128A Register File. The ADSP-1402 is functionally identical to the ADSP-1401, except for the changes described in this section. A block diagram of the ADSP-1402 is shown below. For a detailed description of the architecture and instruction set of the ADSP-1402, see the *Word-Slice® User's Manual* and the *ADSP-1401 Data Sheet.*



ADSP-1402 Block Diagram

The ADSP-1402 is a high speed microprogram controller optimized for the demanding sequencing tasks found in digital signal processors and general purpose computers. In addition to high speed and large addressing range (64K of program memory), this Word-Slice component has unique features that make it highly versatile:

- On-chip storage and control of ten prioritized and maskable interrupts plus a nonmaskable trap,
- Four decrementing event counters,
- Absolute, relative and indirect addressing capability,
- Download capability (writeable control store) and
- A dynamically reconfigurable 64-word RAM.

The ADSP-1402 microprogram sequencer's main task is to provide the appropriate microprogram addressing to support programming requirements, such as looping, jumping, branching, subroutines, condition testing and interrupts. An internal Look-Ahead pipeline, controlled by both phases of the clock, allows the ADSP-1402 to satisfy these requirements at very high speed.

During each microinstruction, the ADSP-1402 monitors the conditions and instructions to determine the next microprogram address. This address can come from one of several sources: the stack, the jump address space in the on-chip RAM, the data port, the interrupt vectors or the microprogram counter. In all cases, the next address is available in a single cycle. An extensive set of conditional instructions is also available, including jumps, branches, subroutines, interrupts and writeable control store. Eight multiplexed flag inputs can be used as external conditions for these instructions.

The ADSP-1402's internal 64-word RAM is user configurable into three regions: subroutine stack, register stack and indirect jump address space. The subroutine stack is used for linking interrupts and subroutines and, during their execution, allowing the storage of system states. The register stack can be used to store sets of jump addresses; each set can be associated with a particular level of interrupt or subroutine (both local and global stacks are provided). Indirect jump capability is also supported, addressing for which is provided at the data port.

Interrupts are handled entirely on chip. The ADSP-1402's internal interrupt control logic includes registers for eight external (user) interrupt vectors, a mask register and a priority decoder. Two additional vectors are reserved for internally generated interrupts resulting from counter underflow and stack limit violation, and a special vector is provided for the nonmaskable trap interrupt. A stack limit violation is caused by stack overflow, underflow or collision. A mechanism is provided for recovering from stack violations. Trap interrupts have the highest priority of all interrupts, and the stack limit violation interrupt has the second highest priority.

The ADSP-1402's four decrementing 16-bit counters are used to track loops and events. These counters generate a signal when negative. This negative condition is available to several conditional instructions and can also trigger an internal interrupt.

CHANGES FROM THE ADSP-1401

TTR Input

The ADSP-1401 TTR (Trap/Tristate/Reset) input is eliminated in the ADSP-1402. In its place are separate $\overline{\text{RESET}}$ and TRAP control pins. The tristate function is implemented with the the $\overline{\text{IDLE}}$ pin as described under *Idle and Halt,* below.

Reset

The default reset function in the ADSP-1402 is similar to that of the ADSP-1401. While $\overline{\text{RESET}}$ is LO and $\overline{\text{IDLE}}$ is HI, the ADSP-1402 outputs H#0000 on its address port. When $\overline{\text{RESET}}$ goes HI, the address port remains at H#0000 for one clock cycle (the first cycle of normal operation). As with the ADSP-1401, the first ADSP-1402 instruction must be a CONT instruction.

The ADSP-1402 also provides an alternate reset function in which the address port is placed in a high impedance state. If the $\overline{\text{IDLE}}$ input is asserted LO during reset, the address port is tristated rather than outputting H#0000. Asserting $\overline{\text{IDLE}}$ during reset does not affect internal operation, only the address port. When $\overline{\text{RESET}}$ goes HI, the ADSP-1402 outputs H#0000 for one clock cycle.

Boot (WCS)

The ADSP-1401 and ADSP-1402 implement a WCS (Writeable Control Store) instruction. This instruction places the ADSP-1401 or ADSP-1402 in a mode in which an active FLAG input increments the program counter (PC), decrements the C_0 counter and outputs the PC to the address port. This operation is used to synchronize address sequencing for downloading microcode from a host. The usual way to exit this mode is by an interrupt, from either an external interrupt or the internal counter underflow (of C_0 in this case).

The ADSP-1402 also provides a pin that allows external hardware control of a download. The BOOT input of the ADSP-1402 controls the operation for downloading microcode in much the same way as the WCS instruction. The boot operation, although slightly more restricted compared to the WCS operation, requires no external circuitry.

Note: $\overline{\text{IDLE}}$ must be HI and TRAP must be LO while the boot function is being used. $\overline{\text{RESET}}$ must be active when BOOT is asserted and remain active until BOOT is deasserted.

In the cycle that BOOT is asserted, the ADSP-1402 outputs H#0000 on the address port and sets the PC to H#0000. When $FLAG_0$ is asserted, the PC is incremented and its new value is output on the address port. The ADSP-1402 remains in this mode until the BOOT pin is deasserted. Thus, no interrupt is required to end the download.

The system clock must be stable and $\overline{\text{RESET}}$ must be asserted for a minimum number of cycles before BOOT is asserted and after BOOT is deasserted. $\overline{\text{IDLE}}$ must be HI and TRAP must be LO for the entire time that BOOT is asserted. When BOOT is active, $FLAG_0$ is edge-sensitive (therefore, it cannot be asserted more than every other cycle). $FLAG_0$ must also meet minimum setup and hold times.

Trap
The ADSP-1402 trap function is controlled by the TRAP input. The TRAP signal must be asserted at least t_{TS} before the next rising clock edge and must be held at least t_{TH} after the rising clock edge. In addition, TRAP must not change state (HI or LO) t_{TS} before or t_{TH} after the rising clock edge, and it must meet a minimum pulse width specification.

The nonmaskable TRAP input on the ADSP-1402 has a dedicated interrupt vector (IV_{10}) that is separate from the IR_9 (stack over/underflow) vector (unlike the ADSP-1401, in which Trap and IR_9 share the same vector). As with the ADSP-1401, the TRAP signal may require a clock skip to allow time to fetch a new instruction. A block diagram of an example circuit for implementing a clock skip is shown in Figure 1. TRAP aborts the current instruction and pushes its address onto the subroutine stack.

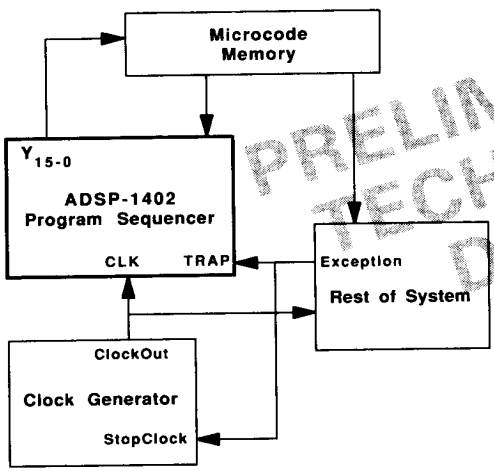


Figure 1. Example Clock Skip Circuit

External Interrupts
The eight external interrupts of the ADSP-1402 are input on eight separate pins, $EXIR_{8-1}$ (whereas the eight external interrupts of the ADSP-1401 are time-multiplexed into four inputs). All external interrupts are latched on the rising edge of the clock. The ADSP-1402 outputs the interrupt vector address in the same clock cycle in which the interrupt is latched.

Interrupt masking and enabling in the ADSP-1402 is the same as in the ADSP-1401. Interrupts on the ADSP-1402 are prioritized in descending numerical order; Trap has the highest priority, IR_9 has the next highest, and IR_0 has the lowest.

Interrupt In Progress (IRIP)
The ADSP-1402 has an internal Interrupt In Progress (IIP) bit that indicates when it is processing an interrupt (IR_9–IR_0). The ADSP-1402 also has an internal Trap In Progress (TIP) bit that indicates when a trap is being processed. The IRIP output flag is the logical OR of the IIP and TIP bits.

If TIP is set, the CCIR (Clear Current Interrupt) and RTNIR (Return From Interrupt) instructions reset TIP without affecting IIP. If TIP is not set, however, then executing one of these instructions resets IIP. Executing the CAIR (Clear All Interrupts) instruction resets both TIP and IIP. Thus, unlike in the ADSP-1401, a trap service routine can be nested inside an interrupt service routine; the return from the trap service routine will not eliminate the Interrupt In Progress status.

Flag Inputs
The ADSP-1402 has eight external flag inputs ($FLAG_{7-0}$). These eight input flags are multiplexed on-chip into one signal that is equivalent to the FLAG input on the ADSP-1401. Three external control bits select one of the eight input flags. The multiplexed flag signal is latched during clock HI and transparent during clock LO. During a Boot or Writeable Control Store operation, the multiplexer automatically selects $FLAG_0$.

Idle and Halt
The ADSP-1402 has two controls for stopping internal operation, one which tristates the address and data ports ($\overline{IDLE}$) and one which does not (HALT). Both perform functions similar to that of the ADSP-1401 IDLE instruction, which is functional but obsolete on the ADSP-1402.

Note: The IDLE instruction must *not* be input to the ADSP-1402 with either $\overline{IDLE}$ or HALT asserted; otherwise, the ADSP-1402 will not function properly.

The $\overline{IDLE}$ pin is useful for implementing multitasking in systems with multiple sequencers. $\overline{IDLE}$ removes the ADSP-1402 from the address and data buses, allowing another sequencer to drive them.

$\overline{IDLE}$ requires a minimum setup and hold time to the rising clock edge. When $\overline{IDLE}$ is asserted, the ADSP-1402 finishes executing the current instruction, and then the internal clock of the ADSP-1402 is stopped, freezing internal operation. At the next rising edge of the CLK input, both the address port and the data port are tristated, and the next instruction is latched but not executed. Fetching and execution of new instructions are inhibited until $\overline{IDLE}$ is deasserted. Interrupts are not latched, and traps are ignored as well. When $\overline{IDLE}$ is deasserted, normal operation continues at the next rising clock edge with the previously latched instruction.

The ADSP-1402 HALT input can be used to stretch the internal ADSP-1402 clock. HALT is primarily intended to implement wait states or to be used in conjunction with TRAP to handle exceptions.

HALT stops internal operation without tristating the address and data ports. It halts internal operation at the next rising edge of the CLK input after HALT is asserted. The address port and the data port are not updated; both ports maintain the states current at a time when HALT is asserted. No new instruction is latched. During HALT, fetching and execution of new instructions are inhibited. Interrupts are not latched; however, unlike during IDLE, active TRAP inputs *are* recognized and processed. The ADSP-1402 latches and executes its next instruction and updates the address and data ports at the next rising clock edge after HALT is deasserted.

Repeat

The REPEAT input causes the ADSP-1402 to repeat the next instruction (the one being set up at the same time as REPEAT) for one clock cycle. This input performs the same function as the ADSP-1401 IR_1 input in IHC (Instruction Hold Control) mode. The ADSP-1402 repeats the instruction as long as REPEAT stays asserted.

Interrupts cannot be serviced while the REPEAT pin is active because the ADSP-1402 ignores its instruction port; however, interrupt requests are still latched. Because TRAP is not latched, it should not be used while REPEAT is active.

The REPEAT input is dedicated to the repeat function; the ADSP-1402 has no IHC mode. In the ADSP-1401, the IHC instruction activates the IHC mode and selects an 8-bit relative jump offset width. For compatibility, the IHC instruction in the ADSP-1402 also selects an 8-bit relative jump offset width (the same effect as the REL8 instruction).

Data Port

The ADSP-1402 has a full-cycle data port rather than the half-cycle data port of the ADSP-1401. Instructions that write data out of the ADSP-1402 drive the bus for a full cycle. Instructions that read data into the ADSP-1402 require data to be valid a specified time before and after the clock rising edge. To avoid bus contention, therefore, an ADSP-1402 instruction that outputs data on the data port cannot be followed by an instruction that reads data from the port; a NOP cycle must occur between the two instructions.

The data port output drivers are tristated unless a data output is being performed.

Power and Ground

The ADSP-1402 has nine power pins and nine ground pins.

PIN LIST

Name	Type	Function
D_{15-0}	Bidirectional	Data Port
Y_{15-0}	Output	Address Port
IRIP	Output	Interrupt in Progress
I_{6-0}	Input	Instruction Port
$EXIR_{8-1}$	Input	External Interrupts
$FLAG_{7-0}$	Input	Flags
$FSEL_{2-0}$	Input	Flag Select
CLK	Input	Clock
TRAP	Input	Trap
$\overline{RESET}$	Input	Reset
$\overline{IDLE}$	Input	Idle
REPEAT	Input	Repeat Instruction
HALT	Input	Halt
BOOT	Input	Boot (WCS) Mode