COMPUCOLOR II and the
Multifunction Input/Output Controller

TMS 5501

by Dale Dewey - D2 ENGINEERING

During the middle '70s when microcomputers were being designed, TEXAS INSTRUMENTS came up with a design concept that was going to save everyone a lot of money. Their idea was to design a lot of functions into one package. Their goal was to reduce component cost, reduce packaging cost, decrease power requirements and a lot of other things. ISC was very interested in saving money on their computer for home use. You all know the rest of this story. History has proven it to be a bad policy to combine many functions into one module. You should keep unlike functions isolated from each other. This gives more flexability and fewer design problems to both the engineers and programmers. So here we are, stuck with a design that has limited flexability and lots of programming problems. I hope to detail some facts on the use of the 5501 controller and perhaps this will help explain why ISC is having problems with a very outdated computer design. Before going to far, it is necessary to lay some ground work on terminology. Remember, hardware is only software for electrical engineers. To begin with, lets define what the 5501 will do for any computer design.

First, it is capable of asynchronous serial data input/output. Most of you know that already but do you know what all of those adjectives mean? In computer designs, data is sent or received at any point in time without a clock (asynchronous) and a bit at a time (serial). Thus, you can't tell when the data is coming but you do know that once a start bit is detected, it will be one bit after another until a stop bit is detected. You must also assume that the bits are coming to you at a fixed rate since there is no clock (time) involved. This rate is known as the BAUD or bits per second rate.

Next, the 5501 can do parallel data input/output. In this case, data is sent or received without any timing and all at once as a single group of bits. For our microprocessor, a group of 8 bits is used and this group is called a byte.

Event timers are also included in the 5501. These timers can be set for a fixed interval of time. When that interval has elapsed, the computer will be notified. There are five timers, only four of which are usable if you want an external interrupt for the parallel input function.

The last function is called interrupt processing. All of the above functions are asynchronous in nature. You never know when an event will occur or which event will occur. The act of notifying the computer that an external (not in your program) event has occured is called interrupt processing.

Our next problem is to communicate with the 5501 so that it can be setup to perform all four of these functions. The COMPUCOLOR II (CC-II) is designed to treat the 5501 as an input/output or I/O port. What did he say? Well O.K., lets work on the idea of an I/O port. I think everyone who has done anything with BASIC has run into the PEEK function and POKE statement. These two tokens are used to examine memory. If you POKE into a memory location, you are addressing that location with a value and storing a second value into that location. For example:

```
10 POKE 33279,0
```

will place a value of zero in the KEYBOARD Character Ready Flag. Likewise, when you PEEK at a location, you are addressing that location with a value and getting a second value from that location. For example:

```
20 Y = PEEK(33278)
```

will set Y equal to the KEYBORD Character value. Some memory addresses in your machine are read only memory (ROM) while others are read/write memory (RAM) and yet others may be write only (non-existant) memory. In all cases, the values stored in memory are always in the range of 0 to 255 (8 bits).

This same concept applies to the I/O ports in your computer. The only limitation is that there are only 256 I/O port addresses (0 to 255). Aside from this, they are treated just like memory. There are two tokens in BASIC which allow you to access them. They are the INP function and OUT statement. There is a third statement (WAIT) which is a very special case of the INP function. If you are an assembler programmer, you have access to the I/O ports with the IN and OUT instructions. By the time we finish with this article, you will be able to use any of these to control the 5501 or any other I/O port in the CC-II.

At this point, I have to repeat a WARNING that ISC puts in all of their manuals. Be careful with some of the I/O ports. If you OUT to or INP from the wrong place or use a wrong value, the CRT controller chip could get very confused and it will actually destroy (SMOKE & FIRE) your computer!!!!! Enough of a warning? Now onto the important stuff.

The following table shows all of the I/O port addresses for the TMS 5501 in the CC-II. Note that single functions have two addresses. This is due to an incomplete decoding of the I/O addresses. This "feature" is another of the ISC cost savers. By only doing a partial decode, they were able to save a chip or two on the logic board. Anyway, use only the lowest addresses. The others work but why confuse yourself? This also means that the addresses 16 to 31 can't be used for anything else.

TMS 5501

Input/Output Port Assignments

I/O ADDRESS	FUNCTIONAL DESCRIPTION
0 & 16	Read serial data receiver
1 & 17	Read parallel data input
2 & 18	Read interrupt device address
3 & 19	Read 5501 status register
4 & 20	Load 5501 command register
5 & 21	Load BAUD rate register
6 & 22	Load serial data transmitter
7 & 23	Load parallel data output
8 & 24	Load interrupt mask register
9 & 25	Load interval timer #1
10 & 26	Load interval timer #2
11 & 27	Load interval timer #3
12 & 28	Load interval timer #4
13 & 29	Load interval timer #5
14 & 30	No function
15 & 31	No function

This is a complete list of the I/O ports. Only some of these are usable by the programmer. This is because your machine operates out of ROM. Many of the functions in the CC-II are controlled by the File Control System (FCS) or Screen (CRT) Mode software. These functions are "burned" into the system ROM and cannot be changed. Those functions which can be programmed, will be fully detailed. Those which are fixed, will be explained by function in the CC-II design.

The first function to discuss in detail is the interrupt processing. Whenever an event occurs (a timer "times-out"), the timer will set a flag inside the 5501. The 5501 will then check to see if the mask bit associated with that event is on. If it is, the 5501 will then notify the 8080 that an interrupt is waiting. The 8080 will complete the present instruction and then send a signal to the 5501. If the interrupt acknowledge bit is on in the 5501 command register, this signal will force a new instruction onto the data bus. This instruction causes the 8080 to jump to one of eight memory locations. These memory locations are called interrupt vectors. They typically contain a jump instruction to a service routine which will handle (in this case restart a timer) the interrupt.

Two more tables are required before going further. The first involves the command register in the 5501. The second is the interrupt mask register which enables or disables each interrupt connected with the 5501. These registers are 8 bits long and can be shown as a set of 8 blocks with each block representing a bit within the register.

TMS 5501

Command Register

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0

Bit Number	Function
0	RESET 1 => Reset serial transmitter and receiver. Clear interrupt mask register. Zero all interval timers. 0 => No action
1	BREAK 1 => Serial transmitter bit set to continuious MARKING (BREAK) condition. 0 => Serial transmitter set for normal (SPACING) operation.
2	INTERRUPT 7 SELECT 1 => XI7 of parallel input port. 0 => Interval timer #5.
3	INTERRUPT ACKNOWLEDGE ENABLE 1 => Enable response to interrupt acknowledge. 0 => Disable response to interrupt acknowledge.
4	HARDWARE TEST BIT
5	HARDWARE TEST BIT
6	NOT USED
7	NOT USED

NOTES: Bit 0 is normally off (leave it that way). This bit is used by the ROM only during a power up sequence.

Bit 1 is used to send a break. If your program detects the BREAK key, set Bit 1 for a short time then reset it.

Bit 2 is normally off (leave it that way).

BIT 3 is normally on (leave it that way).

Don't use this register unless you understand what each bit does. Check the 5501 specification sheet in the programmers manual for details on BITS 4 & 5. These bits are used to test the 5501.

TMS 5501

Interrupt Mask Register

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0

Bit Number	Function	Vector Address
0	INTERVAL TIMER #1 1 => ENABLE INTERRUPT 0 => DISABLE INTERRUPT	0
1	INTERVAL TIMER #2 1 => ENABLE INTERRUPT 0 => DISABLE INTERRUPT	8
2	EXTERNAL SENSOR (REAL TIME CLOCK) 1 => ENABLE INTERRUPT 0 => DISABLE INTERRUPT	16
3	INTERVAL TIMER #3 1 => ENABLE INTERRUPT 0 => DISABLE INTERRUPT	24
4	SERIAL CHARACTER RECEIVED 1 => ENABLE INTERRUPT 0 => DISABLE INTERRUPT	32
5	SERIAL CHARACTER SENT 1 => ENABLE INTERRUPT 0 => DISABLE INTERRUPT	40
6	INTERVAL TIMER #4 1 => ENABLE INTERRUPT 0 => DISABLE INTERRUPT	48
7	INTERVAL TIMER #5 OR PARALLEL INPUT (XI7) (SEE COMMAND REGISTER) 1 => ENABLE INTERRUPT 0 => DISABLE INTERRUPT	56

As another example, let's consider the real time clock in the CC-II. The clock (HOURS, MINUTES and SECONDS) is under control of the 5501 chip. There is a signal which comes from the power supply in your machine that is applied to the EXTERNAL SENSOR pin on the 5501. This signal is derived from the 60 HZ (50 HZ) power source and will cause an interrupt every 1/60th (1/50th) of a second. When this signal goes from GROUND to +5 VOLTS, it will set the internal interrupt flag in the 5501. If BIT 2 in the MASK REGISTER is set, the 5501 will notify the 8080 that an interrupt is waiting. When the 8080 completes the current instruction cycle, it checks to see if the programmer has interrupts enabled.

NOTE: Only assembler programmers can turn interrupts on and off with the EI and DI instructions.

If interrupts are on, the 8080 will send an acknowledge signal to the 5501. The 5501 now checks to see if BIT 3 in the COMMAND Register is on. If it is, the 5501 will place a restart (RST) instruction on the bus. This causes the 8080 to save the next instruction address for the presently running program and execute the instruction in one of the vector locations. In the case of our clock, that vector is located at memory address 16. The 8080 will execute this instruction, a JMP to UPDATE. This interrupt service routine then saves all registers; increment the fractions of a second counter in memory; update the HOURS, MINUTES and SECONDS memory locations as necessary; restore all registers and then return. The program which was interrupted now starts running again just as if nothing happened.

If you can follow all of that, you are off and running with the 5501. If you are not sure, go back and read the last two paragraphs again. Draw a picture or flow chart of each step. As you might expect, interrupt service routines can take a lot of time if they are very long. It is considered good programming practice to keep the routines as short and fast as possible. This lets your normal program do its thing.

Now that we have been through all of that, what can you the programmer do with these two I/O ports? The answer is not a lot. The programs in ROM will keep all bits in the MASK Register ON except when the disk is being accessed. If you turn one of them OFF it will stay OFF for a short period of time and then be turned ON again by another interrupt service routine in ROM. If you want complete control over this register, you must not allow any programs in ROM to execute. That means you must write your own FCS! The one thing you can do with these two ports is to send a BREAK character using the COMMAND Register. In addition, you can disable the interrupt acknowledge function by setting BIT 3 off. This is done only if you have another piece of hardware which can acknowledge an interrupt service request (such as an ANALOG to DIGITAL converter). Just turning it off will give unknown results.

Now lets look at the INTERVAL Timers in the 5501. These timers are very simple to use. You load a number (0 to 255) into them then every 64 microseconds one is subtracted from that number. When the number reaches zero, the timer generates an interrupt. If you load a zero, you will get an interrupt without any delays.

One useful function of these timers is as watchdogs. Let's say you have an input that must occur every 10 milliseconds and you want to know if it doesn't. Easy, set a timer for 175 counts (11 ms). Each time you get an input, restart the timer by loading 175 into it. You will never get a timer interrupt. If the inputs should stop, you will get an interrupt when the timer reaches zero (your input is 1 ms late). Most of the timers in the 5501 are dedicated to your CC-II. There are very few things you can do with all of them.

TIMER #1

This timer is not used in the CC-II, but its interrupt vector is used. If you start the timer, you will be reset to CRT mode when it times out. This timer makes a good watchdog timer and is a nice way to kill a runaway program. It will always bring you back to the CRT mode.

TIMER #2

This is the famous user timer. Its vector address, which contains a JMP to location 33224, is 8. You must put a jump to your service routine in location 33224. Be sure to save all registers on entry and restore all registers on return. You must also restart timer #2 if you want it to run again.

TIMER #3

This timer is used to "scan" the keyboard. Each time it times out, the keyboard is checked to see if any key is depessed. Changing this timer will only change the scan of the keyboard for one time interval since the scan routine will restart the timer with a fixed time constant.

TIMER #4

This timer is used by the nonexistant BELL. It controls the length of time that the bell sounds if you install one and fix the V6.78 ROM bug. The V8.79 bug was fixed and you would get a "beep" if you had a bell.

TIMER #5

This timer is not used in the CC-II, but its interrupt vector is used. If you do start it, you will end up in an "executive loop" on time out. This is a do-nothing loop which waits for anything else to happen. This loop is used by the CRT mode while it is waiting for some key to be typed on the KEYBOARD.

The parallel output port is 8 bits wide and serves two functions in the CC-II. When the disk drives are not being used, the port is used to address the keyboard. During disk drive use, the port controls the stepper motor, the read/write function and the drive select.

Whenever a disk drive is selected, the 5501 is a very busy chip. First, it controls the positioning of the read/write head in the floppy disk drive. It also selects the proper drive, places it in the read or write mode and transfers data for the floppy disk using the Serial Transmitter/Receiver. While this is going on, all interrupts except the Serial Character Sent or Serial Character Received are masked OFF. The keyboard is turned back on, the Mask bits are set ON and the modem is turned on when the disk access function is completed. The parallel input port is not used during disk access.

When the disks drives are off, the lower 4 bits of the parallel output port are used as a binary number to address the keyboard. There are 16 scan lines in the keyboard and a decode chip in the keyboard converts this address into a signal which will select a single line of keys. After setting the output port to address a line of keys, the parallel input port is tested. All 8 inputs are used, and the code read from the port will indicate which key(s) is(are) depressed. In addition, BIT 7 of the parallel output port may be set ON and the high 4 bits of the parallel input port will indicate the state of the special keys.

The following chart details the Parallel Output Port. The bit assignments are good for both the 3-phase (V6.78) and 4-phase (V8.79) disk drives. The hardware in the drive corrects for the difference in stepper motors.

The only way you as a programmer can control or scan for a unique key is to stop Timer #3. This can be done only by turning interrupts OFF. The following BASIC program demonstrates this operation for those of you who would like to try it. You must remember that while scanning the keyboard ALL interrupt driven events are disabled (no disk acess, no REAL TIME CLOCK, nothing). There is a better way to get input from the Keyboard. This is done by changing the KEYBOARD Flag value at location 33247. By doing this, you get to leave interrupts on and still select and sort which keys you want to use. See the Advanced Programmers Manual for more details on the use of the I/O Flags. The binary numbers for input and output to the keyboard are tabulated next. Notice that the Binary Code is not ASCII and that the Keyboard Address and Code are inverted. Another good reason to find some other way to read the keyboard.

TMS 5501

Parallel Output Port

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0

Bit Number	Function
0	Bit 4 and 5 => 0 Keyboard address Bit 0 Bit 4 or 5 => 1 Stepper Motor Phase Bit 0
1	Bit 4 and 5 => 0 Keyboard address Bit 1 Bit 4 or 5 => 1 Stepper Motor Phase Bit 1
2	Bit 4 and 5 => 0 Keyboard address Bit 2 Bit 4 or 5 => 1 Stepper Motor Phase Bit 2
3	Bit 4 and 5 => 0 Keyboard address Bit 3 Bit 4 or 5 => 1 0 => Read from disk 1 => Write to disk
4 & 5	Bit 4 and 5 => 0 Keyboard enable and Modem enable Bit 4 => 1 and Bit 5 => 0 Select internal drive (CD0) Bit 4 => 0 and Bit 5 => 1 Select external drive (CD1)
6	NOT USED
7	Bit 4 and 5 => 0 0 => Normal Keyboard operation 1 => Test Keyboard for SHIFT, CONTROL, REPEAT or CAPS LOCK key Bit 4 or 5 => 1 NOT USED

TMS 5501

Keyboard Scan Program

```
 10 INPUT "INPUT KEYBOARD ADDRESS >";N

 19 REM  Set for special keys (SHIFT, CONTROL, - - -)
 20 IF  N> 15 THEN  N= 128

 99 REM  Mask OFF all interrputs
100 OUT  8,0

109 REM  Reset TMS 5501
110 OUT  4,1

119 REM  Load Keyboard Address into output port
120 OUT  7,N

129 REM  Wait a little before reading keyboard
130 FOR  I= 1 TO  1000:NEXT  I

139 REM  Read Keycode from input port
200 A= INP (1)

199 REM  Report results if not a special key
210 IF  A< > 255 AND  N< 16 THEN  300

220 IF  N< 16 THEN  200

299 REM  Enable TMS 5501
300 OUT  4,8

309 REM  Mask ON all interrupts except serial transmitter
310 OUT  8,223

319 REM  Start TIMER #3 to scan keyboard
320 OUT  11,128

399 REM Print result of input
400 PRINT "KEYCODE > ";A

409 REM Start the porgram over
410 GOTO  10
```

TMS 5501

Keyboard Codes

Binary Code
Input Port

Address Output	254	253	251	247	239	223	191	127
15	0	@	P	F0	BREAK	BLACK	SPACE	
14	1	A	Q	F1	INSERT CHAR	RED		
13	2	B	R	F2	DELETE LINE	GREEN		
12	3	C	S	F3	INSERT LINE	YELLOW		
11	4	D	T	F4	DELETE CHAR	BLUE		
10	5	E	U	F5	AUTO	MAGENTA		
9	6	F	V	F6	CYAN			
8	7	G	W	F7	WHITE			
7	8	H	X	F8	HOME			
6	9	I	Y	F9	TAB	CURSOR RIGHT		
5	:	J	Z	F10	CURSOR DOWN	CURSOR LEFT	X	
4	;	K	[	F11	ERASE LINE	ESC	+	
3	<	L	\	F12	ERASE PAGE	CURSOR UP		
2	-	M	]	F13	CR	FG ON	=	
1	>	N	^	F14	A7 ON	BG ON		
0	/	O	_	F15	A7/BL OFF	BLINK ON		
128					CNTL	SHIFT	RPT	CAPS LOCK

The last function in the TMS 5501 which we need to cover is the Serial Transmiter/Receiver. When you want to send a byte of information out the serial port you OUT it to the transmitter. It will add a start bit to the beginning of your byte to let the remote receiver know that a byte is coming. Next it will send the zero bit, the one bit and so on until it gets to the seven bit. At this point the transmitter will add one or two stop bits to your byte to let the remote receiver know that the end of the byte has been reached. Most receivers require only one stop bit, however, it is best to check to see what is required. If you can't find out, use two stop bits to be safe.

The receiver works in a similar manner except in reverse. It is always looking for a start bit. When received, the zero bit is set up, the one bit is set and so on until the seven bit is set. The receiver now checks to see if a stop bit is received. If at least one is not received, this is called a framing error. Once a full byte is received, the Status Register is set to indicate that it is available. You can get the byte by doing an INP from the serial port.

There are four I/O ports which are used to control the serial port. The first is the BAUD or bit rate register. It is used to set the rate at which bits are sent or received. You can not have different rates for the transmitter and receiver. The chart below indicates the bit assignments for the BAUD Rate Register. If you use this register directly, be sure to also set location 33250 (CRATE). The best way to use this register is to use PLOT 27,18,R command in BASIC where R is the bit number plus one for the rate you want. BASIC will take care of updating location 33250.

The Status Register will indicate the "status" of the serial port and the interrupts. Its bit functions are shown in the following chart. Each bit will indicate something about the status of the 5501 and is generally used to find out what is going on within the chip itself.

The last two ports are the actual serial transmitter or receiver. In normal operation, the following steps would be used.

1) Set the BAUD rate
2) Get a byte to send
3) Test the status to see if the Transmitter Buffer is empty (transmitter is not busy)
4) OUT the byte to the transmitter
5) Test the status to see if the Receiver Buffer is full (byte waiting for pick up)
6) INP the byte from the receiver
7) Test the status for any error conditions
8) Repeat steps 2) to 7) as necessary

Notice that Modem control (Handshake) and parity are not features of the transmitter or receiver. If you need these, they must be taken care of in software or by other hardware means.

TMS 5501

BAUD Rate Register

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0

Bit Number	Function
0	Set for 110 BAUD
1	Set for 150 BAUD
2	Set for 300 BAUD
3	Set for 1200 BAUD
4	Set for 2400 BAUD
5	Set for 4800 BAUD
6	Set for 9600 BAUD
7	Number of STOP BITS transmitted 0 => two 1 => one

In general, 110 BAUD is assumed to be with 2 STOP BITS. This means that 11 bits are transmitted for every byte. If a byte represents one character, then it is correct to say that 110 BAUD is equivalent to 10 characters per second. At 110 BAUD, the maximum efficency of the serial port is 73% (8 bits received for every 11 sent). The other rates assume only 1 STOP BIT. The maximum efficency of these rates is 80%. If you use 2 STOP BITS, the only thing you are changing is the serial port through put. This may have no effect on overall system performance.

TMS 5501

Status Register

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0

Bit Number	Function
0	Framing Error When this bit is ON, it indicates that the last character did not have a stop bit after bit seven of the byte was received. This error is normally caused by unlike BAUD rates for the receiver and remote transmitter.
1	Overrun Error When this bit is ON, it indicates that a second character was received before the first one was picked up from the serial receiver buffer. This will happen when your interrupt service routine is too long. Try slowing down the BAUD rate to correct the problem.
2	Serial Data Received When this bit is ON, it indicates that no data is being received. This condition would indicate that a BREAK is being detected.
3	Receiver Buffer Full When this bit is ON, it indicates that a byte has been received by the serial receiver and that it is available for pick up.
4	Transmitter Buffer Empty When this bit is ON, it indicates that the serial transmitter buffer is empty and ready to receive another character for transmission.
5	Interrupt Pending When this bit is ON, it indicates that one of the interrupts has occured and that the corresponding interrupt Mask bit was ON. This status is used when interrupts are disabled.
6	Full Bit Detected This is a test bit.
7	Start Bit Detected This is a test bit.

As an example of using the serial transmitter, let's look at the following BASIC program.

```
10 REM Erase Page,A7 Off (2 SB),110 BAUD
11 PLOT 12,15,27,18,1

100 REM Send all printable ASCII characters to
101 REM the serial port then print them on the CRT
102 FOR I=32 TO 128

110 REM Test bit 4 to be sure buffer is empty
111 WAIT 3,16,16

120 REM Send to serial port
121 OUT 6,I

130 REM Print the character on the CRT
131 PRINT CHR$(I)

140 NEXT I
```

Statement 10 sets the BAUD rate and the number of stop bits to be used by the serial transmitter. Statement 100 sets up a loop to print all ASCII characters. Statement 110 tests port 3, bit 4. This is done using the WAIT statement. This statement takes the INP value of port 3 (the first argument), exclusive OR's it with 16 (the last argument), and then AND's it with 16 (the second argument). If the result of this test is zero, control will pass to statement 120 otherwise stetement 110 is executed again. This will cause the program to "loop" until the transmitter buffer is empty. Statement 120 sends I to the serial port. Statement 130 prints the value of I on the CRT. Statement 140 increments I and the program will run until I equals 129.

This is a very simple program that is very effective in demonstrating the use of the serial port. Set your printer to 110 BAUD and run the program. Notice the output is missing no characters. Now delete statement 111. See what happens? The transmitter buffer is being overwritten before a character can be sent. This condition would be even worse if you were writing your programs in ASSEMBLER.

One port which I have not covered is the Interrupt Device Address. This port is of no use to the CC-II. This is used when interrupts are turned OFF and you are testing Bit 5 in the Status Register. In this case, doing an INP from port 2 will cause your program to pass control to the Vector Address for the event which caused Bit 5 to be set in the first place. Since we are working with a ROM based system, this is a useless function.

Summary:

* Port 0 Serial Receiver
  Characters may be received without having to modify anything. Bit 3 of Port 3 will tell you if a character is waiting. See page 12 for details.

* Port 1 Parallel Data Input
  No use to programmers

* Port 2 Interupt Device Address
  No use to programmers

* Port 3 Status Register
  Lets you know what is going on. Of particular interest are Bit's 0,1,2,3,4. See Page 14 for details on each Bit

* Port 4 Command Register
  Use Bit 1 to send a Break. See Page 4 for details.

* Port 5 Baud Rate Register
  No real use. Use BASIC PLOT 27,18,R command to set this register.

* Port 6 Serial Transmitter
  Characters may be sent without having to modify anything. Bit 4 of Port 3 will tell you if a character may be sent. See page 12 for details.

* Port 7 Parallel Data Out
  No use to programmers

* Port 8 Interrupt Mask Register
  No use to programmers

* Port 9 Interval Timer #1
  Not used by CC-II. You can use it to return to CRT mode. See page 7 for details

* Port 10 Interval Timer #2
  This is your timer. See page 7 for details.

* Port 11 Interval Timer #3
  Used to scan keyboard. Not useable by you.

* Port 12 Interval Timer #4
  Not usable in CC-II. Bell timer.

* Port 13 Interval Timer #5
  Not used by CC-II. Will cause you to return to "do-nothing" loop if you use it.

References:

Programming Manual Intelligent Systems Corp.	999222
Advanced Programming Manual D2 Engineering.	
Maintemance Manual Intelligent Systems Corp.	999208
Specification Sheet Texas Instruments Inc.	TMS-5501
Application Report - TMS 5501 Texas Instruments Inc.	CA-185A