The Speech Synthesizer module

Introduction

The speech synthesizer module is a stand-alone unit that fits inbetween the console and the peripheral connection cable (if any). If contains a TMS5220 speech synthesis chip and two TMS6100 serial ROMs that hold a fairly limited vocabulary. Here are some pictures.

The TMS5220 synthesizer chip can receive speech data either from the serial ROMs or directly from the CPU. It contains a 16-byte parallel in / 128-bit serial out FIFO buffer for the latter purpose. It interacts with the CPU via three registers: the Command register (input), the Data register (output) and the Status register (output).

The speech synthesis logic uses the serial data to generate digital speech that is accessible on the `I/O` pin. This signal is fed to an internal digital-to-analog converter to produce an analog signal on pin `SPEAKER`, that can be used to drive a speaker. In the case of the TI-99/4A, the analog signal is sent to the TMS9919 sound chip inside the console.

Pinout

```
        +----+--+----+
     D7 |1 o       28| RS*
   ADD1 |2         27| WS*
 ROMCLK |3    T    26| D6
    Vdd |4    M    25| ADD2
    Vss |5    S    24| D5
    OSC |6         23| ADD4
    T11 |7    5    22| D4
SPEAKER |8    2    21| ADD8/DATA
    I/O |9    2    20| TEST
PROMOUT |10   0    19| D3
   Vref |11        18| READY*
     D2 |12        17| INT*
     D1 |13        16| M1
     D0 |14        15| M0
        +------------+
```

<u>Power supply</u>
Vdd -5V (drain supply voltage)
Vss +5V (substrate supply voltage)
Vref 0V (ground reference voltage)

```
                               TMS5220
         Pin #1              +---------+
+5V---------> >--------+-----|Vss      |
                0.1 uF |     |         |
            Gnd---||---+     |         |
       Case gnd---||---'     |         |
                0.1 uF       |         |
           Gnd----||---,     |         |
      Case gnd----||---+     |         |
                0.1 uF |     |         |
-5V---------> >--+-----+-----|Vdd      |
        Pin #43  |           |         |
                 |  220 Ohm  |         |
                 '--www--+---|Vref     |
                         |   +---------+
             Gnd---|>|---'
                  1N914 diode

Gnd---------> >---------Gnd
   Pins #21,23,25,27

            :
  TI-99/4A  :     Synthesizer module
```

Upon power-up, an internal circuitery ensures a clear condition 95% of the time, provided Vss-Vdd reaches +10 volts in less than 2 millisecond. This is done by issuing an internal "Reset" command, which lasts 15 milliseconds. To ensure a 100% clear reset condition, the software can send nine >FF bytes to the synthesizer, followed with a "Reset" command (>Fx).

<u>System interface</u>
D0-D7 Data bus. D0 is the most significant bit (weight >80), D7 is the least significant bit (weight >01). In the TI module, these lines are connected to the data bus present on the side port, pins #34-40, and 43.

RS* Read select. This input pin goes low when the CPU wants to read data from the synthesizer.

WS* Write select. This input pin goes low when the CPU wants to write data to the synthesizer.

If both `RS*` nor `WS*` are high, the synthsizer outputs are in high impedance state.

If both `RS*` and `WS*` go low, results are unpredictable. This never occurs in the TI module, as `RS*` and `WS*` signals come from a 74LS138 decoder which by definition can never bring more than one output low.

```
         74LS138                                             74LS138        TMS5220
        +--------+                            Pin #5         +--------+   +---------+
A2------|A    Y0*|                     A5---------> >--------|A    Y0*|   |         |
A1------|B    Y1*|                                      Gnd--|C    Y1*|   |         |
A0------|C    Y2*|                                    ,------|B    Y2*|---|WS*      |
    1K  |     Y3*|           74LS138              #3  |      |     Y3*|---|RS*      |
+5V-www-|G1   Y4*|--,      +--------+  RESET*-----> >--------|G1   Y4*|   |         |
Memen*--|G2A* Y5*|  | A5---|A    Y0*|                 | Gnd--|G2A* Y5*|   |         |
   Gnd--|G2B* Y6*|  | A4---|B    Y1*|  A15--------> >--------|G2B* Y6*|   |     INT*|-nc
        |     Y7*|  | A3---|C    Y2*|             #19 |      |     Y7*|   |         |
        +--------+  |      |     Y3*|                 | 10K  +--------+   |         |
                    '------|G2A* Y4*|---,             +-www---+5V         |   READY*|--,
      A5----,         A15--|G2B* Y5*|---=|)o------> >-'   Gnd----, |      +---------+  |
      DBIN--=|)o-----------|G1   Y6*| 74LS03      #2              \|---------www-------'
```

```
           74LS03          |     Y7*|                             /|         10k
         (NAND gate)       +--------+            #12           V  2N3904
                                     READY------< <------------'  Transistor
                                                :
       TI-99/4A console                         :         Synthesizer module
```

The TI-99/4A console generates a signal on pin #2 of the side port, to be used when the speech synthesizer is accessed. This signal decodes A0-A5, A15, DBIN and MEMEN*, and is active (high) for read operations at even addresses in the range >9000-93FE and write operations at even addresses in the range >9400-97FE.

Inside the synthesizer module, there is a 74LS138 decoder that combines this signal it with A15/CRUOUT to react only to even addresses (although this was already taken care of in the console) and with the RESET* line. The decoder uses address line A5 to distinguish read operations from write operations and triggers WS* or RS* on the TMS5520.

READY* The speech synthesizer is a very slow device. All I/O operations require halting the CPU until the synthesizer is done with reading/writing data on D0-D7. The `READY*` pin is used for that purpose: it goes high 100 ns after `RS*` or `WS*` goes low, to signal the device is not ready. In the TI module, the `READY*` pin controls a 2N3904 transistor that connects the `READY*` line of the side port (pin #12) to the ground when passing.

INT* This output pin goes active low when the TS (talk status) status bit turns zero. If TS turns one during a read cycle, the `INT*` pin will go low after the cycle has been completed. It will also go low during a Speak External command if the BL (buffer low) or BE (buffer empty) bits turns zero. This pin is not connected in the TI module.

Speech memory interface
ADD1-ADD8/DATA These four output pins are used to send an address to the external speech memories. The address is sent as 5 nibbles, and includes a chip selection code. ADD8/DATA also serves as an input pin to read data from the memory.

M0, M1 These two output pins carry command bits to the speech ROMs. M1 is pulsed high five times to pass an address to the ROMs, M0 is pulsed high to read subsequent bits from the ROM. Pulsing high both M0 and M1 causes an internal readn-and-branch in the ROM.

ROMCLK This output pin is used to synchronize operations when accessing the speech ROMs. It is derived from the OSC signal, divided by 4 (`ROMCLK` corresponds to phi2).

```
   TMS5220             TMS6100
 +---------+         +--------+
 |     ADD1|---------|ADD1    |+
 |     ADD2|---------|ADD2    ||
 |     ADD4|---------|ADD4    ||
 |     ADD8|---------|ADD8    ||
 |         |         |        ||
 |       M0|---------|M0      ||
 |       M1|---------|M1      ||
 |   ROMCLK|---------|CLK     ||
 |         |    Gnd--|CS      ||
 |         |         +--------+|
 +---------+          '--------+
```

Sound interface
SPEAKER This is the output of the digital-to-analog converter. It carries an analog sound signal from 0 to 1.5 mAmp, with a resolution of 5.9 uAmp. In the TI module, this pin is grounded via a 1.8K resistor (so as to generate a voltage accros it), and is filtered with a parallel 0.22 uF cap. It also goes to the side port, pin #44, via a 1 uF serial cap. Inside the console, it goes through a 330 Ohm resistor to the AUDIOIN pin of the TMS9919 sound chip.

```
 +---------+     0.22 uF                          +-----------+
 | TMS5220 |   +---||----Gnd                      |  TMS9919  |
 |         |   | 1.8K                               |           |
 |  SPEAKER|---+---www---Gnd                        |           |
 |         |   |               pin #44              |           |
 |         |   +---||-------------> >---www-----| AUDIOIN   |
 |      I/O|-nc   1 uF                  330 Ohm  |           |
 |      T11|-nc                                   +-----------+
 +---------+                    :
 Synthesizer module             :         TI-99/4A console
```

I/O The digital sound data, upstream the D/A converter can be read on this pin. The data is in the form of a signed 10-bit value, synchronized by `ROMCLK`. This pin in not connected in the TI module.

T11 This pin signals that data will be available on the I/O pin. The signal remains active (high) for two pulses, and data appear on the `I/O` pin the next pulse after `T11` went low. This pin is not connected in the TI module.

```
  _   _   _   _   _   _   _   _   _   _   _   _   _   _
_| |_| |_| |_| |_| |_| |_| |_| |_| |_| |_| |_| |_| |_| |_ ROMCLK
      ___
_____|   |_______________________________________________ T11
             ___________________________________________
____________|lsb|___|___|___|___|___|___|___|___|msb|___ I/O
```

Clock interface
OSC This input pin provides the clock signal used by the synthesizer. It is generally connected to a RC circuit tuned to 640 kHz (which provides an internal sample rate of 8 kHz and a `ROMCLK` signal of 160 kHz) or to 800 kHz (which results in a sample rate of 10 kHz and a `ROMCLK` signal of 200 kHz). The manual recommends a 80-100 kOhm resistor to select 10 kHz and a 120-200 kOhm resistor to select 8 kHz. It also advises a 10 pF shunt capacitor in parallel with the resistor to filter out noise. In the TI speech module, there are 3 resistors in parallel that can be cut out to select the proper frequency (and no shunt cap). In my module one of the resistor has been removed and the remaining two add up to a resistance of 209 kOhm.

The OSC signal is internally divided by 4 to generate four phase clocks: PHI1 (major phase), PHI2 (major phase, `ROMCLK`), PHI3 (pre-load for PHI1) and PHI4 (pre-load for PHI2).

Alternatively, OSC could be connected to a 320 kHz ceramic resonator, whose other pin is connected to Vss (-5 Volts). However, this option must be enabled during manufacture of the device and is therefore not accessible to us. It is possible however to feed a 320 kHz squarewave (0 / +5 volts) clock signal to OSC if `PROMOUT` is connected to Vss.

PROMOUT This pin is for test purposes and is normally not connected. If it is forced to -5 V, it disables the internal oscillator, so that OSC accepts an external clock signal.

TEST This pin is for test purposes and must not be connected.

```
   TMS5220
 +---------+       243K
 |         |   +---www---+
 |         |   |    ??   |
 |      OSC|---+  -www-  -+---- -5V
 |         |   |   1.5M  |
 |         |   +---www---+
 |     TEST|-nc
 |  PROMOUT|-nc
 +---------+
```

Putting the synthesizer inside the PE-box

Having the speech-synthesizer and the "firehose" PE-box cable daisy-chained in the console side port is a bit tricky. First it takes space, then its prone to poor contact that may result in loosing all your work if you accidentally bump this assembly.

For this reason, a quick hack was designed by Joe Spiegel to let you install the speech synthesizer board inside the PE-box. It does require some additional circuitery though, since the PE-box bus does not match exactly the console side port.

```
            7805
       +----------+      +5V
+8V]---|Vin   Vout|---+----+-----+--------------[+5V
       |   Ref    |   |    |     '---To all
       +----------+   = 47 = 0.1     chips
            |         | uF | uF
           Gnd       Gnd  Gnd

             7905
        +----------+
-16V]---|Vin   Vout|---+----+-------------------[-5V
        |   Ref    |   |    |
        +----------+   = 47 = 0.1
            |          | uF | uF
           Gnd        Gnd  Gnd
```

First, the power supply is unregulated in the PE-box. So the connection card must carry two voltage regulators: a 7805 provides +5 volts (also used by the additional TTL chips) and an 7905 provides -5 volts.

Then the PE-box bus does not contain the SBE selection signal that indicates access to the speech synthesizer. The adapter board must thus carry the necessary logic.

Contrarily to TI requirements, many address lines are not buffered in the adapter board. This is because they only drive one chip, so it doesn't make any difference whether this chip is a buffer or something else... Lines that are used for more than one purpose (A5, A15 and DBIN) are buffered by three gates of a 74LS367 tri-state buffer. Obviously, these gates are permanently enabled. Finally, some lines are not used by the adapter board and go directly to the synthesizer: READY, RESET* and AUDIOIN.

In the schematics below, a `>--` denotes a line from the PE-box bus, whereas a `--<` denotes a pin of the speech-sythesizer board.

```
A5>-----|>---A5                       READY>--------------<READY
         |                           RESET*>--------------<RESET*
A15>----|>---A15                    AUDIOIN>--------------<AUDIOIN
         |
DBIN>---|>---Dbin
         | 74LS367
        Gnd

           74LS138                  74LS138
          +-------+                  +-------+
A2>-------|S0     |            A5----|S0     |
A1>-------|S1     |       A4>--------|S1     |
A0>-------|S2     |       A3>--------|S2     |
          |    Y4*|------------------|G2A*   |
Memen*>---|G2A*   |       2.2K A15---|G2B*   |
    Gnd---|G2B*   |  +5V---WWW---+---|G1  Y4*|-+---,
+5V--WWW--|G1     |  A5----,     |   |    Y5*|-|-+-=|)o------<SBE
     1K   |       |  DBin--=|)o--'   |       | | |
          +-------+       74LS00     +-------+ ^ ^ 1N914 diodes
                                               | |
                        RDBENA*]---<|---Gnd    | |   1K
                                   '-----------+-+---WWW--+5V
           74LS245              74LS367        |
          +-------+                            |
          |    OE*|----------------------------'
D0>-------|       |-----------------------------------------<D0
```

```
     .    |       |      .
     .    |       |      .
     .    |       |      .
D7>-------|       |-----------------------------------------<D7
          |    DIR|----Dbin
          +--------+
```

Address decoding is performed by two 74LS138 decoders that react to memory operations in the range >90xx-94xx. A15 is included to make sure that only even addresses are taken into account. A5 and DBIN are combined with a 74LS00 NAND gate to make sure that read operations do not reach >9400. This exactly mimics the circuitery found inside the console, and described above.

Two outputs of the second decoder react to >90xx and >94xx respectively. They are combined with a 74LS00 NAND gate to provide the active-high selection signal SBE to the speech synthesizer. Two 1N914 diodes mounted as a "wired-and" play the same role to provide an active-low signal that enables the 74LS245 data bus buffer (whose direction is set by DBIN) and a 74LS367 tri-state buffer. This buffer makes the DRBENA* line go low to activate the data bus buffers in the connection card and cable.

I haven't tried this circuit myself, but I was told that it works...

Internal structure

General organisation

```
            +----------+
RS*-------->|          |
WS*-------->| CPU I/O  |
INT*<-------| Control  |
READY*<-----|          |
            +----------+
D0-D7
 ||     +-------------+
 ||====>| Command reg |----------,
 ||     +-------------+          |                                 ............
 ||                              |   +--------------+              :          :
 ||     +-------------+          |   | Timing logic |---> ROMCLK ----->:  TMS6100 :
 ||<====| Status reg  |          |   +--------------+              :          :
 ||     +-------------+          V                                 :  Speech  :
 ||                        +----------------+                      :   ROMs   :
 ||     +-------------+    | Speech ROM     |<=======> ADD1-ADD8 ==>:          :
 ||<====| Data reg    |<===| control logic  |--------> M0 --------->:          :
 ||     +-------------+  || |                |--------> M1 --------->:          :
 ||                      || +----------------+                      :..........:
 ||     +-------------+  ||
 \|====>| 16-byte     |  ||  +-----------------+     +---------------+
        | FIFO buffer |=======>| Synthesis logic |--+-->| D/A converter |---> SPEAKER
        +-------------+       +-----------------+  |   +---------------+
                                                   '-----------------------> I/O
```

The status register
This output register contains only 3 relevant bits:

TS	BL	BE	..	..	..	..	..

TS Talk Status. Bit 0, weight >80. This bit is 1 when the synthesizer is processing data. This occurs immediately after a "Speak" command or 50 usec after nine bytes were loaded by a "Speak-External" command. TS goes back to 0 after the stop code (energy=1111) is encountered, when the FIFO becomes empty or in case of a reset. In the first two cases, the audio output is interpolating toward zero during the current frame and will only terminate at the next frame.

BL Buffer Low. Bit 1, weight >40. This bit becomes 1 during a "Speak-External" command, when the number of bytes in the FIFO decreases below 8. It reverts to zero 50 usec after a ninth byte is written into the FIFO.

BE Buffer Empty. Bit 2, weight >20. This bit becomes 1 during a "Speak-External" command, when there is no more byte in the FIFO. This clears TS, terminates speech (at some abnormal point) and redirects incoming bytes to the command register.

The data register
This output register is organised as a serial-in/parallel-out buffer. It serves to hold data transfered from the speech ROM and pass it to the CPU as a single byte ("Read" command). The last bit transfered is the rightmost, least significant one.

The FIFO buffer
This input buffer is organized as a 16-byte parallel-in/128-bit serial-out stack, obeying a first-in, first-out logic. It is used to hold data passed bytewise by the CPU for the "Speak external" command. The synthesizer shifts out bits as it needs them to create speech, once 8 bits have been used, the stack ripples down by one byte and begins shifting bits out of the second "first in" byte. An internal stack pointer keeps track of the "last in" byte, so that the synthesizer knows where to put incoming bytes. The position of this pointer whithin the stack is reflected in the status register BL and BE bits.

The command register
This input register is used to internally latch the command passed by the CPU. There are 7 possible commands: Reset, Load-address, Read-byte, Read-and-branch, Speak, Speak-external and Load-frame-rate. The later is only available on the more advanced TMS5520C synthesizer, on the TMS5520 it is considered as a NOP (no operation) command.

Speech synthesis logic

```
             +-----------+      +--------------+
        /===>| Coded    4|=====>| Parameter   5|==============\
        ||   | param RAM |      | look-up ROM  |             ||
        ||   +-----------+      +--------------+             ||
        ||                                                   ||
    +-------------+                                        6 ||
===>| Coded param |                                          ||
 1  | input reg  2|                                          ||
    +-------------+                                          ||
        ||                                                   \/
        ||   +-------------------+   ,---------------,    +--------------+
        \===>| Decode logic and 3|==>| Interpolation |--->| Parameter   7|
             | condition latches |   | controller    |    | interpolater |
             +-------------------+   `---------------'    +--------------+
                                                             |    |
                                            ,----------------' 8  |
                                            V                     V
                                  +-----------+        +-------------+
                                  | Signal    |=======>| LPC lattice |----> to D/A
                                  | generator |   9    | network     |  10  converter
                                  +-----------+        +-------------+
```

1. Speech data is send to the speech synthesis logic in the form of coded parameters, either from the FIFO (data sent by the CPU) or from the ROM control logic (data fetched from the ROMs).
2. Data are fed serially into the parameter input register.
3. Data are unpacked and various tests are performed: is the repeat bit set? Is pitch 0? Is energy 0 ?
4. Unpacked parameters are stored in a parameter RAM
5. Saved parameters are used as indexes to fetch the appropriate 10-bit values from the lookup ROMs.

6. The output of the lookup ROMs are the target values for the interpolation logic to reach during this frame period. A frame period lasts 25 msec with a 640 kHz oscillator, 20 msec at 800 kHz.
7. The interpolater reaches the new values in eight steps. Each one of these eight interpolation periods lasts 20 `ROMCLK` periods, which is 3.125 msec (assuming a 640 kHz oscillator). Speech data is read (1) on the first of the eight.
8. After each interpolation period, the interpolater sends new pitch and energy parameters to the signal generator, and new reflection parameters (K1-K10) to the LPC lattice network.
9. The signal generator produces the filter excitation sequence for both voiced and unvoiced frames.
10. At the end of each sample period, digitized speech data are available on the `I/O` pin and to the D/A converter. A sample period is defined as one `ROMCLK` period, which is 6.25 usec with a 640 kHz oscillator. This corresponds to a sampling rate of 8kHz.

Timing	10 kHz	8 kHz
Oscillator rate Osc period	800 kHz 1.25 usec	640 kHz 1.5625 usec
`ROMCLK` rate `ROMCLK` period	200 kHz 5 usec	160 kHz 6.25 usec
Sample rate Sample period	10 kHz 100 usec	8 kHz 125 usec
Interpolation rate Interpol. period	400 Hz 2.5 msec	320 Hz 3.125 ms
Frame rate Frame period	50 Hz 20 msec	40 Hz 25 msec

D/A conversion

The digital speech signal available on the I/O pin, is converted into an analog signal available on the `SPEAKER` pin, by an internal D/A converter with a 2% LSB linearity resolution. Every sample period (125 usec with a 640 kHz oscillator), the most-significant 10 bits of the 14-bit LPC lattice network output are sampled. The seven least-significant bits are sent directly to the D/A converter, together with the sign bit (most significant bit). The remaining two bits, YC and YB, are combined with the sign bit and used to clip the driver to a "full-on" or "full-off" condition.

The resulting ouput is a current from 0 to 1.5 milliamps with a resolution of 5.9 microamps, which is optimal to drive the TMS9919 sound chip. The 1.8K resistor in series to the ground, causes the `SPEAKER` pin to deliver 2.7 volts when the lattice output is less than -127. When the lattice output is greater than 128, `SPEAKER` is clipped to 0 volts. When no speech takes place, the lattice output is -1, which causes the `SPEAKER` ouput to drive 750 microamps, this is meant for an AC-coupled speaker.

LPC latice outputs															D/A input	Analog out (uA)
Value	YD	YC	YB	YA	Y9	Y8	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0		
> +127	0	1	1	x	x	x	x	x	x	x	x	x	x	x	11111111	0
> +127	0	1	0	x	x	x	x	x	x	x	x	x	x	x	11111111	0
> +127	0	0	1	x	x	x	x	x	x	x	x	x	x	x	11111111	0
+127	0	0	0	1	1	1	1	1	1	1	1	x	x	x	11111111	0
+126	0	0	0	1	1	1	1	1	1	1	0	x	x	x	11111110	5.86
etc.	0	0	0	.	.	.	.	.	.	.	.	x	x	x	...	...
+1	0	0	0	0	0	0	0	0	0	0	1	x	x	x	10000001	738.0
0	0	0	0	0	0	0	0	0	0	0	0	x	x	x	10000000	744.0
-1 (off)	1	1	1	1	1	1	1	1	1	1	1	x	x	x	01111111	750.0

-2	1	1	1	1	1	1	1	1	1	1	0	x	x	x	01111110	755.8
etc.	1	1	1	.	.	.	.	.	.	.	.	x	x	x	...	...
-128	1	1	1	0	0	0	0	0	0	0	0	x	x	x	00000000	1500
< -128	1	1	0	x	x	x	x	x	x	x	x	x	x	x	00000000	1500
< -128	1	0	1	x	x	x	x	x	x	x	x	x	x	x	00000000	1500
< -128	1	0	0	x	x	x	x	x	x	x	x	x	x	x	00000000	1500

Speech ROMs

The data manual for the TMS5020C specifies that the serial memories can be TMS6100, TMS6125 or custom speech ROM or EPROMs. In the TI module, there are two TMS6100 custom chips that are piggy-backed, i.e. mounted on the top of each other which each and every pin on one chip connected to the corresponding pin on the other.

Internally, the TMS6100 is organized as 16 Kbytes, but some logic was added inside the chip so that it appears as 128 Kbits. Firstly, the address can be latched into an internal counter. Since the bus is only 4-bit wide, the address is passed as five consecutive chunks, for a total of 20 bits. The logic inside the ROM remembers how many chunks were passed and where to place the next nibble. Reading operations reset this logic.

Bits are read one at a time from the speech ROM. To this end, the ROM contains a data register into which it copies the currently accessed byte. Each successive read operation causes a register shift and a bit of data is sent out. An internal counter keeps track of the read operations and loads the next byte when needed. This automatically updates the address latch.

Pinout

```
     +----+--+----+
 Vdd |1 o       28| nc
  nc |2         27| nc
ADD1 |3         26| nc
ADD2 |4         25| nc
ADD4 |5         24| nc
ADD8 |6         23| nc
 CLK |7         22| nc
  nc |8         21| nc
  nc |9         20| nc
  M0 |10        19| nc
  M1 |11        18| nc
  nc |12        17| nc
 CS* |13        16| nc
 Vss |14        15| nc
     +------------+
```

Power supply
Vdd -5 V
Vss +5 V

Synthesizer interface
CS* Chip select. In the TI module this pin is connected to Vss, thus constantly active. Selection is achieved by mean of a special selection code sent on the `ADD1-ADD8` pins, after the address.

ADD1-ADD8 Address pins. To be connected to the corresponding pins on the synthesizer. `ADD8` also serves as output pin for the serial data. The address is passed at 14 bits to the `ADD1-ADD8` pins. It is followed by a chip selection code of 4 bits. Each chip has its own internal selection mask (programmed during manufacture for ROMs). If the selection code matches the mask, the chip will be selected. This allows to select upto 16 chips without the need of an external decoder.

M0, M1 Control pins. There are four possible combination of states for these two pins:
M0=0, M1=0: Idle state. The ROM is passive.
M0=0, M1=1: Load address. The next 4 bits of address are loaded into the ROM's address latch.
M0=1, M1=0: Read. The next bit from the current byte is sent on ADD8. If necessary, fetches the next byte.
M0=1, M1=1: Read-and-branch. The ROM fetches 2 bytes at the current address and uses the least significant 14 bits as a new address.

<u>Non-connected pins</u>
Although not connected internally, many of these pins are connected externally in the TI module. Pins #2 and #9 are connected to Vss (+5V), pins #15 through 28 are connected together, probably for mechanical support.

Speech ROM operation

As explained above, the TMS6100 speech ROMs latch an address as 20 bits, passed as five nibbles of 4 bits (since the bus is 4-bit wide). To load an address, the speech synthesizer places a nibble on the bus, toggles M1 up for the duration of one `ROMCLK` period and repeats this operation five times. The first 14 bits make up an address inside the ROM chip. The next four bits form a selection code that must match the internal chip code to select this chip. This allows parallel connection of upto 16 chips and provides about 30 minutes of speech. The last 2 bits are ignored (d.c. = don't care bits)..

Once the address is latched, the synthesizer toggles `M0` high for two `ROMCLK` pulses. This is known as a "dummy read" operation. No data will be transfered, the goal is only to reset the internal load pointer so that the next "load-address" operation loads the first 4 bits. The synthesizer must wait at least 80 usec afterwards, before attempting to read data from the speech ROMs.

From that point on, the synthesizer can read data one bit at a time, by pulsing `M0` high for the duration of one `ROMCLK` period. On the next `ROMCLK` pulse, the selected ROM will send a bit on `ADD8`, and increment its internal bit counter. Once 8 bits have been read, the ROM increments the address counter, fetches the next byte, and passes its first bit.

In summary, for all practical purposes, the speech ROM can be considered as a countinuum of 128K bits, that can be addressed starting at every 8th bit.

Here is the timing diagram of a "load address" operation:

```
                                              _______
_____________________________________________|       |___ M0
  ___     ___     ___     ___     ___
_|   |___|   |___|   |___|   |___|   |___________________ M1
  _   _   _   _   _   _   _   _   _   _   _   _   _   _
_| |_| |_| |_| |_| |_| |_| |_| |_| |_| |_| |_| |_| |_| |_ ROMCLK (6.25 usec)
  ___     ___     ___     ___     ___
X|A3_|XXX|A7_|XXX|A11|XXX|CS1|XXX|d.c|XXXXXXXXXXXXXXXXXXX ADD8
  ___     ___     ___     ___     ___
X|A2_|XXX|A6_|XXX|A10|XXX|CS0|XXX|d.c|XXXXXXXXXXXXXXXXXXX ADD4
  ___     ___     ___     ___     ___
X|A1_|XXX|A5_|XXX|A9_|XXX|A13|XXX|CS3|XXXXXXXXXXXXXXXXXXX ADD2
  ___     ___     ___     ___     ___
X|A0_|XXX|A4_|XXX|A8_|XXX|A12|XXX|CS2|XXXXXXXXXXXXXXXXXXX ADD1

 |<-- Address loaded as 5 x 4 bits ->|       ^Dummy read
```

Now the synthesizer waits at least 80 microseconds before it starts reading bits:

```
  ___     ___     ___     ___     ___     ___     ___
_|   |___|   |___|   |___|   |___|   |___|   |___|   |___________ M0
```

```
_________________________________________________________________ M1
_|¯|_|¯|_|¯|_|¯|_|¯|_|¯|_|¯|_|¯|_|¯|_|¯|_|¯|_|¯|_|¯|_|¯|_|¯|_|¯|_ ROMCLK
XXXXX|__D0___|__D1___|__D2___|__D3___|__D4___|__D5___|__D6___|XXX ADD8
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX ADD4
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX ADD2
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX ADD1
```

In this example, 7 bits are read (D0 to D6), but the operation could continue for as many bits as desired.

Speech ROMs in the TI module

The TI Speech Synthesizer module has two custom ROM chips. They contains a "binary-tree" list of words in plain ascii, with the addresses where to find speech data for each word. If you want to view the content of these ROMs, you can use my Module Explorer program, and set the memory type as "s".

Texas Instruments originally intended to release more ROMs to extend the vocabulary of the module. That's what the little door on the top of the module is for. However, they realized that speech produced merely by concatenating words is of poor quality: After. All. Nobody. Speaks. Like. That! To some extent it is possible to improve it by reading speech data, modifying a frame here and there (e.g. stripping the last frame of a word to link it to the next), and feeding the result to the synthesizer. But even like this, the result it poor.

TI thus created a much more versatile program, that was integrated into the Terminal Emulator module (don't ask me why). The module GROM contains speech data for a list of allophons, i.e. all possible sounds in English. The module ROM contains two subprograms, the first one breaks plain english sentences into a list of allophons, the second creates the speech data from those allophons, adds accentuation and voice inflexion, and passes it to the speech synthesizer via the Speak-External command. This provides a much more satisfactory discourse.

Operating the Speech Synthesizer

The CPU controls the speech synthesizer by writing commands to the command register. If the command is "Speak-External" all following writing operations will be redirected to a FIFO buffer (First In-First Out). This buffer can accept 16 bytes of data and feed them serially as 128 bits to the speech synthesis circuits.

All read operations return the content of the status register, except immediately after a "Read-Byte" command: in this case the next byte of data comes from the data register. This is a serial-in, parallel-out register that accepts 8 bits from the speech memory and pass them as 1 byte to the CPU.

In the TI-99/4A, the speech synthesizer maps at >9000 for read operations and at >9400 for write operations.

Commands

Only one command at a time can be passed to the command register. If the CPU tries to send a second byte to the synthesizer before the current command is terminated, the synthesizer will activate the READY line to stall the CPU, until it is ready to accept the next byte. Note that this is not true for read operations.

There are seven possible commands, most are one byte long, but some may require additional bytes.

Command byte	Operation
x111xxxx	Reset

```
x100aaaa     Load-Address (needs 4 more bytes)
x101xxxx     Speak
x110xxxx     Speak-External (accepts an unlimited number of data bytes)
x001xxxx     Read-Byte
x011xxxx     Read-and-Branch
x0x0xvrr     Load-Frame-Rate (rr=rate code, v=0:use rr, 1: variable) TMS5520C only.
```

Load-Address
This command sends a 14-bit address and a 4-bit chip select code to the speech memories. The chip whose internal selection mask matches the selection code will set its internal pointer to the specified address. Since there are only 4 address lines (ADD1, ADD2, ADD4 and ADD8), we'll need five successive Load-Address commands to complete the operation. The following table explains how to compose the five required bytes:

```
Bit    : 0 1 2 3 4   5   6   7
Byte 1: x 1 0 0 A3  A2  A1  A0
Byte 2: x 1 0 0 A7  A6  A5  A4
Byte 3: x 1 0 0 A11 A10 A9  A8
Byte 4: x 1 0 0 CS1 CS0 A13 A12
Byte 5: x 1 0 0  x   x  CS3 CS2
```

Speak
This command causes the synthesizer to read data from the serial memory (through the ADD8/DATA line) and to use it to generate speech. The TS bit becomes 1 in the status register and speech begins on the next frame boundary. Speech continues until the stop code (energy = >F) is received, at which point the synthesizer interpolates towards zero. Speech will be terminated and the TS bit reset at the next frame. Alternatively, the Reset command can be used to interrupt speech and reset TS immediately, whithout interpolation to zero.

Speak-External
This commands allows the CPU to supply its own speech data, rather than reading them from the speech ROMs. Upon reception of the command, the FIFO is cleared (which sets the BL and BE status bits and activates the INT* pin) and all subsequent data from the CPU is sent to the FIFO. Nothing happens until the ninth byte is received. At that time, BL becomes 0, TS becomes 1 and speech begins 50 microseconds later. Speech continues until the stop code is processed (a frame with the energy set as 1111) or until the buffer is empty or the synthesizer is externally reset (by bringing RS* and WS* low together). The Reset command cannot be used to stop speech as it will be mistaken for speech data.

Although this is not mentionned in the manual, I found out by experimenting with the Speak-External command, that it is not possible to overflow the FIFO with the TI-99/4A: when it becomes full it activates the `READY` line and stalls the CPU. Therefore, a speech program only has to ensure that the FIFO is never empty (by monitoring BL), and doesn't need to care about passing bytes too fast.

Read-Byte
This command is used by the CPU to read the next 8 bits from the speech ROMs. This assumes that a Load-Address command has been previously issued. Only the next byte will be speech data, all subsequent reading operations access the status register, until another Read-Byte command is issued.

Read-and-Branch
This command causes the synthesizer to read two bytes at the current address in the ROMs memory. These two bytes are used to select a new address into the current ROM, i.e. they constitute a pointer. There is a 240 usec delay before the new address is set and the synthesizer is ready for a Speak command.

This command is usefull because it allows to create different versions of the ROMs, that can be operated by the same program. For instance, imagine we want to create a french version of the Speech Synthesizer module. It is very unlikely that speech data for the word "Bonjour" will be at the same address in the french ROMs than data for "Hello" in the english version. Thus a program using the Load-Address and Speak commands will produce gibberish, unless it is completely rewritten. If on the other hand the ROMs contain a vector table (i.e. a table with the addresses of the speech data for the various words), all we have to do is to make sure that the vector for "Bonjour" is at the same address than the vector for "Hello" in the english version. The calling program will just use Read-and-Branch without knowing which language the ROMs contain.

Load-Frame-Rate
This command is used to tell a TMS5520C synthesizer whether it should use a fixed frame rate, or use the first two bits of each frame as a frame rate (it is ignored by the older TMS5520). The command byte has the form: x0x0xvrr where v is the frame mode and rr the fixed frame rate. If v=0, rr will be used as a frame rate until further notice. If v=1, rr is irrelevant and the frame rate must be passed with each frame. The four possible frame rates are:

00:	200 samples/sec
01:	150 samples/sec
10:	100 samples/sec
11:	50 samples/sec

Reset
This command is used to interrupt the current Speak command. The synthesizer stops speaking briskly, without interpolation to zero, the TS status bit is reset and the FIFO is cleared (which sets the BL and BE bits and activates the INT* pin). This command also sends a "load address" command to the speech ROMs (using a dummy address), followed with a dummy read pulse.

Speech encoding

I don't have the pretention to understand all the electronic mumbo-jumbo in the TMS5220C manual, therefore I'll just quote excerpts here and let you figure it out by yourself.

"Linear Predictive Coding (LPC) synthesizes human speech by recovering from the original speech enough data to construct a time-varying digital filter model of the vocal tract. This filter is excited with a digital representation of either glottal air impulses (voiced sound) or the rush of air, which produces unvoiced sound. The output of this filter model is passed through a 8-bit digital-to-analog (D/A) converter to produce a synthetic speech waveform."

"The LPC analysis program begins with a set of digitized speech samples. These digitized samples are usually derived with an analog-to-digital (A/D) converter by sampling an analog wavefore at a rate of 8 or 10 kHz. Consecutive samples are grouped together to form a "frame" of digitized samples. The frame may contain 50 to 400 samples, but usually contains 200. The LPC analysis routine operates on these digitized samples, a frame at a time, by preemphasizing the samples, calculating the energy, pitch, and the spectral coefficients (K-parameters). Next, each value is coded according to a pre-selected coding table."

"This coded speech parameter data is fed serially from either the speech memory or the FIFO buffer to the parameter input register. Here the Controller unpacks the data and performs various tests (i.e., is the repeat bit set, is pitch zero, is energy zero). Once unpacked, the coded parameter data is stored in RAM to be used as the index value to select the appropriate value from the Parameter Look-Up ROM. The outputs of the Parameter Look-Up ROM are the target values for the interpolation logic to reach in this frame period. During each of the interpolation periods the interpolation logic sends new parameter values to the LPC lattice network which makes avalaible a new value of digitized speech to the D/A converter."

"The LPC method of speech encoding reduces the speech data rate from approximately 100,000 bits/sec (raw digitized speech) to about 4800 bits/sec. The analyzer reduces this rate further (to 2000 buts/sec or less) by encoding each of the 20-bit speech parameters as 3 to 6-bit codes. These coded values select a 10-bit parameter from the parameter look-up ROM in the processor. Depending on the influence of the parameter on speech quality, between 8 and 64 possible values are stored in the Look-Up ROM for decoding and use in synthesis calculation. Note that the parameter ROM in the TMS5220C is mask programmable and not touchable or alterable by the user".

What this boils down to is:

1. Special hardware is needed to produce LPC-encoded speech, and we don't have a good description of it.
2. Speech is coded by: Pitch, Energy and ten reflection parameters K1-K10.
3. However, the value of these parameters is meaningless to us because it is an index into look-up tables burried inside the synthesizer. These table contains the real 10-bits parameter values, but we cannot alter them. Thus we are stuck with the predefined parameter values.

I once wrote a test program that fetches an allophone from the Terminal Emulator module, displays the frames it contains, then lets me modify their parameters and listen to the result. My goal was to derive a set of allophones for French. I spent endless hours trying to understand the effect of a given K parameter and finally had to give up. I wasn't even able to come up with a single allophone: the "an" sound (as in "en passant") that I was using as a test.

I recently discovered a free speech-generation Windows software, WSDS, on the Texas Instruments website. Unfortunately, WSDS only works with a dedicated sound board (which is not free...). There may be a way around it, as the Speech Editor, which is part of the package, can read .RAW sound files. So it may be feasible to record words on .RAW files, read them with the Speech Editor, and pass them to WSDS to create the LPC data. I haven't tried yet.

Now, here is how speech is encoded:

```
Frame type  Energy Rpt Pitch   K1    K2   K3   K4   K5   K6   K7   K8  K9  K10
Voiced      xxxx   0  xxxxxx xxxxx xxxxx xxxx xxxx xxxx xxxx xxxx xxx xxx xxx
Unvoiced    xxxx   0  000000 xxxxx xxxxx xxxx xxxx
Repeated    xxxx   1  xxxxxx
Silence     0000
Stop code   1111
```

Energy sets the volume for a given sound, or part of sound.
Pitch is the frequency for that sound.
Rpt is the repeat bit.
K1-K10 are the reflection parameters index values.
With the TMS5520C, we don't have to use a fixed frame rate: we can also specify the rate for each frame, using two more bits at the beginning of each frame to set the frame rate (see the Load-Frame-Rate command).

As you can see, only voiced frames (i.e. vowels) require the whole set of parameters, that add up to 50 bits. Unvoiced frames (consonants and whispers) can dispense with K7-K10, and only require 31 bits per frame. The synthesizer recognises unvoiced frames because they have a pitch value of 00000.

Since the human vocal tract changes shape relatively slowly, it is often necessary to repeat the same frame several times, possibly with different pitches. To save memory space, a special repeat bit has been introduced between energy and pitch: if this bit is set the previous K parameters are retained and the resulting frame only needs 11 bits of data.

Finally there are two special cases: silence that has an energy of zero and is required for interword or intersyllabe pauses. And the stop code that signals the synthesizer to stop speaking and has an arbitrary energy value of 1111. Both these frames are only 4 bits long (6 bits with two leading frame rate bits).

Synthesizer lookup tables

Below are the values hidden in the parameter lookup tables used by the TMS5220.

Energy

Value	RMS
>00	0

>01	52
>02	87
>03	123
>04	174
>05	246
>06	348
>07	491
>08	694
>09	981
>0A	1385
>0B	1957
>0C	2764
>0D	3904
>0E	5514
>0F	7789

Pitch

Value	**Pitch**	**Value**	**Pitch**	**Value**	**Pitch**	**Value**	**Pitch**
>00	0	**>10**	30	**>20**	50	**>30**	91
>01	15	**>11**	31	**>21**	52	**>31**	94
>02	16	**>12**	32	**>22**	53	**>32**	98
>03	17	**>13**	33	**>23**	56	**>33**	101
>04	18	**>14**	34	**>24**	58	**>34**	105
>05	19	**>15**	35	**>25**	60	**>35**	109
>06	20	**>16**	36	**>26**	62	**>36**	114
>07	21	**>17**	37	**>27**	65	**>37**	118
>08	22	**>18**	38	**>28**	68	**>38**	122
>09	23	**>19**	39	**>29**	70	**>39**	127
>0A	24	**>1A**	40	**>2A**	72	**>3A**	132
>0B	25	**>1B**	41	**>2B**	76	**>3B**	137
>0C	26	**>1C**	42	**>2C**	78	**>3C**	142
>0D	27	**>1D**	44	**>2D**	80	**>3D**	148
>0E	28	**>1E**	46	**>2E**	84	**>3E**	153
>0F	29	**>1F**	48	**>2F**	86	**>3F**	159

Reflection coefficients

Value	**K1**	**K2**	**K3**	**K4**	**K5**	**K6**	**K7**	**K8**	**K9**	**K10**
>00	-0.97850	-0.64000	-0.86000	-0.64000	-0.64000	-0.50000	-0.60000	-0.50000	-0.50000	-0.40000
>01	-0.97270	-0.58999	-0.75467	-0.53145	-0.54933	-0.41333	-0.50667	-0.31429	-0.34286	-0.25714
>02	-0.97070	-0.53500	-0.64933	-0.42289	-0.45867	-0.32667	-0.41333	-0.12857	-0.18571	-0.11429
>03	-0.96680	-0.47507	-0.54400	-0.31434	-0.36800	-0.24000	-0.32000	0.05714	-0.02857	0.02857

>04	-0.96290	-0.41039	-0.43867	-0.20579	-0.27733	-0.15333	-0.22667	0.24286	0.12857	0.17143
>05	-0.95900	-0.34129	-0.33333	-0.09723	-0.18667	-0.06667	-0.13333	0.42857	0.28571	0.31429
>06	-0.95310	-0.26830	-0.22800	0.01132	-0.09600	0.02000	-0.04000	0.61429	0.44286	0.45714
>07	-0.94140	-0.19209	-0.12267	0.11987	-0.00533	0.10667	0.05333	0.80000	0.60000	0.60000
>08	-0.93360	-0.11350	-0.01733	0.22843	0.08533	0.19333	0.14667			
>09	-0.92580	-0.03345	0.08800	0.33698	0.17600	0.28000	0.24000			
>0A	-0.91600	0.04702	0.19333	0.44553	0.26667	0.36667	0.33333			
>0B	-0.90620	0.12690	0.29867	0.55409	0.35733	0.45333	0.42667			
>0C	-0.89650	0.20515	0.40400	0.66264	0.44800	0.54000	0.52000			
>0D	-0.88280	0.28087	0.50933	0.77119	0.53867	0.62667	0.61333			
>0E	-0.86910	0.35325	0.61467	0.87975	0.62933	0.71333	0.70667			
>0F	-0.85350	0.42163	0.72000	0.98830	0.72000	0.80000	0.80000			
>10	-0.80420	0.48553								
>11	-0.74058	0.54464								
>12	-0.66019	0.59878								
>13	-0.56116	0.64796								
>14	-0.44296	0.69227								
>15	-0.30706	0.73190								
>16	-0.15735	0.76714								
>17	-0.00005	0.79828								
>18	0.15725	0.82567								
>19	0.30696	0.84965								
>1A	0.44288	0.87057								
>1B	0.56109	0.88875								
>1C	0.66013	0.90451								
>1D	0.74054	0.91813								
>1E	0.80416	0.92988								
>1F	0.85350	0.98830								

Timing diagrams

(For the CPU interface. See above for the speech ROM interface)

Write cycle for commands and speech data

```
        _______________|________>12 us_________| Next command
RS*                                            \______
        _____           ________________________
WS*          \__>6 us__/                        \______
             |a|>200ns_||_>100ns|
D0-D7   XXXXXXX|_valid data in__|XXXXXXXXXXXXXXXXXXXXXX
               |b_|______________
READY*  _________/   18-26 us    \_____________________
                 |       see note  c           |
a) <7 us
b) <100 ns
c) <595 us for Read-and-Branch command
   <42 us for Load-Address command
   <56 us for Speak command
   <287 us for Speak command after Load-Address
   <42 us for Speak-External command
   <10 us for external speech data
   <300 us for Reset command
```

Read cycle for status transfer

```
       ____                    ____>12 us_____ Next command
RS*        \______>6 us_______/               \_______
       _______________________________________
WS*                                           \_______
            |a |_______|b |
READY* ________/ 6-11 us \____________________________
                   |2 us | ____________
D0-D7  ZZZZZZZZZZZZxxxxxxx|_valid data_|ZZZZZZZZZZZZZZ

a) <100 ns
b) 2 us (unstable data)
```

Read-Byte sequence

```
       ____Write cycle________________________| Read cycle        ________
RS*                   |           >12 us      \_____>8 us________/
       ____           ____________________________________________________
WS*        \__>6 us__/
            |a|>200ns||_>100ns_|                           _______|_c|
D0-D7  XXXXXXX|_valid data in__|ZZZZZZZZZZZZZZZZZZZZZZZZXX|_valid out|ZZZZ
            | b |_______________               |b|_______
READY* _________/    <26 us     \________________/       \________________
                               |    see note d          |

a) <7 us
b) <100 ns
c) 4-9 us
d) <440 us after Load-Address
   <320 us otherwise
```

Electrical characteristics

Absolute maximum ratings

Any pin with respect to Vss...............	-20V to +0.3V
Power dissipation..............................	600 mW

Recommended operating conditions

Parameter	Min	Nom	Max	Unit
Vdd	-5.5	-5	-4.5	Volts
Vss	4.75	5	5.5	Volts

Vref	-	0	-	Volts
High level input	Vss-0.6	-	Vss	Volts
Low level input	Vdd	0	Vss-4	Volts
Free-air temperature	0	-	70	`C
Storage temperature	-40	-	70	`C
Operational freq (RC)	576	640	704	kHz

Electrical characteristics under recommended conditions

Parameter	Test conditions	Min	Nom	Max	Unit
High level output voltage	0.1 mAmp	Vss-0.5	-	Vss	Volts
Ditto for D0-D7,WS*,RS*,INT*	0.4 mAmp	2.4	-	Vss	Volts
Low level output voltage	0.1 mAmp	-	-	Vss-4.5	Volts
Ditto for D0-D7,WS*,RS*,INT*	1.6 mAmp	Vref-0.5	0	Vref+0.5	Volts
Supply current from Vref	Ref to Vss	-	3	5	mAmp
Supply current from Vdd	Ref to Vss		10	35	mAmp
Data bus load capacitance	-	25	-	300	pF
Other pins input capacitance	-	-	15	-	pF
Other pins output capacitance	-	-	15	-	pF

Revision 1. 1/31/99 OK to release
Revision 2. 3/30/99 Polishing
Revision 3. 4/1/00 Got the TMS5220 manual! Added ROM tables, block diagrams, schematics, etc.
Revision 4. 8/16/00. Added intro, link to picture.
Revision 5. 8/26/01. Added link to a picture page.
Revision 6. 5/14/02. Added installation inside PE-box.

Back to the TI-99/4A Tech Pages