Description of ts(.) array in AUTOSET

P 2. action

Purpose: Action is basically a continuation of the music processing program. It is called when the data for a measure is complete.

Inputs: @n = number of elements in data arrays
tvl(.) = element type

type  element
      ----   ------
  1    regular note
  2    extra regular note in chord
  3    regular rest
  4    cue note
  5    extra cue note in chord
  6    cué rest
  7    grace note or grace rest
  8    extra grace note in chord
  9    figured harmony
 10   bar line
 11   musical direction
 12   invisible rest
 13   backspace
 14   clef change
 15   time designation or other directive
 16   time change
 17   change in divspq
 18   key change
 19   print suggestion

tv2(.) = duration for types 1--9, 12,13
= measure number for type 10
= track number for type 11 (1 = default)
= new clef number for type 14
= 0 for type 15
= new time flag for type 16
= new divspq for type 17
= new key for type 18
= type of suggestion for type 19

if between 0 and 7
 0 = force slur 1 over
 1 = force slur 1 under
 2 = force slur 2 over
 3 = force slur 2 under
 4 = force slur 3 over
 5 = force slur 3 under
 6 = force slur 4 over
 7 = force slur 4 under

if between 8 and 9
8 = overhanded tie (tips down)
9 = underhanded tie (tips up)

if between 16 and 31 (03-21-97)
  bit 0: clear = square tuplet
        set = round tuplet
  bit 1: clear = broken tuplet
        set = continuous tuplet
  bit 2: clear = number outside tuplet
        set = number inside tuplet

if between 128 and 255
  font = type - 128

if between 0x100 and 0x1ff
  vert and/or horz adj to musical dir

if between 0x200 and 0x2ff
  vert and/or horz adj to sub-obj

if between 0x300 and 0x3ff
  vert and/or horz adj to note/rest/fig
  objects.

if between 0x400 and 0x4ff
  suggestion for barline or measure

  tv3(.) & 0x00ff = staff number  (0 or 1)

  For notes,
  & 0xff00 = value of repeater_flag
  For rests,
  & 0xff00 = value of restplace
  For musical directions
  & 0xff00 = value of optional forward
          offset for division counter

  tv5(.) used for flagging $ data that occurs
  at the beginning of a measure, but
  is not typeset immediately

  tcode(.) = pitch (rest) for types 1--8
  = number of figure fields for type 9
    (figured harmony)
  = bar type for type 10
  = musical direction code and position
    for type 11
  = "ires" for type 12
  = "back" for type 13
  = "0" or "128" for type 14 (clef change)
  = "" for types 15--18
  = for type 19 (print suggestions)
    a 4 byte code
byte 1: 0x01: active flag (0 = inactive)
        0xfe: various meanings
        (for ties only)
        length modification (+128)
        (0 = no data)
        (for start slurs (ff only)
        curvature modification (+128)
        (0 = no data)

byte 2: x-y active flags
        0x01: active flag (0 = inactive)
        0x02: 0 = x position is relative
              1 = x position is absolute
        0x04: 0 = y position is relative
              1 = y position is absolute

byte 3: x position data (+128) (0=no data)
byte 4: y position data (+128) (0=no data)

tdata(.) = additional data for types 1--9, 11, 19

Output:  ts(.,.)

Description of ts
---------------

Case I:  Notes, Rests, Grace Notes, Cue Notes, Cue Rests
        Extra Regular, Grace, and Cue notes in Chords
        (types 1--8)

        ts(1) = type:  1 = note
                     2 = extra regular note in chord
                     3 = rest
                     4 = cue note
                     5 = extra cue note in chord
                     6 = cue rest
                     7 = grace note or grace rest
                     8 = extra grace note in chord

        ts(2) = division number (starting with 1)
        ts(3) = clave    <100 = clave number
                   100 = rest
                   101 = movable rest
                   200 = irest

        ts(4) (used initially to store pointer to tcode(.) )
        ts(4) = accidental flag

        bits 0x0f: 0 = none       6 = natural-sharp
                   1 = natural      7 = natural-flat
                   2 = sharp        10 = double sharp
                   3 = flat         15 = double flat

        bit 0x10: 0 = regular     1 = "silent"
bits 0xff00: left shift (positioning)

\[ ts(5) = \text{note type} \]
\[
1 = 256th note \\
2 = 128th note \\
3 = 64th note \\
4 = 32nd note \\
5 = 16th note \\
6 = 8th note \\
7 = quarter note \\
8 = half note \\
9 = whole note \\
10 = breve \\
11 = longa \\
12 = eighth with slash
\]

\[ ts(6) = \text{dot flag} \quad 0 = \text{no dot}, \quad 1 = \text{dot}, \quad 2 = \text{double dot} \]

\[ ts(7) = \text{tuplet flag} \quad 0 = \text{no tuplet}, \quad # = \text{tuplet} \]

\[ ts(8) = \text{location on staff} \]

\[ ts(9) = \text{spacing number} \]

\[ ts(10) = \text{stem/chord flag} \quad \begin{array}{ccc}
\text{bit} & \text{clear} & \text{set} \\
0 & \text{no stem} & \text{stem} \\
1 & \text{step up} & \text{stem down} \\
2 & \text{single note} & \text{chord} \\
3 & \text{first note} & \text{extra note} \\
4-7 & \text{(note number in chord)} & \\
\end{array}
\]

\[ ts(11) = \text{beam flag} \quad 0 = \text{no beam} \\
1 = \text{end beam} \\
2 = \text{start beam} \\
3 = \text{continue beam} \]

\[ ts(12) = \text{beam code (up to six digits)} \]

This is an integer less than 1000000. The one's digit is the code for the eighth beam; the tens digit is the code for the sixteenth beam, etc.

\[ \begin{array}{ccc}
\text{digit} & \text{char} & \text{meaning} \\
0 & \text{blank} & \text{no beam} \\
1 & = & \text{continued beam} \\
2 & [ & \text{begin beam} \\
3 & ] & \text{end beam} \\
4 & / & \text{forward hook} \\
5 & \backslash & \text{backward hook} \\
6 & & \text{simple repeater} \\
7 & & \text{begin repeated beam} \\
8 & & \text{end repeated beam} \\
\end{array} \]

\[ ts(13) = \text{local x-offset (for chords)} \]

\[ ts(14) = \text{superflag} \quad \begin{array}{ccc}
\text{bit} & \text{set} \\
0 & \text{tie} \\
\end{array} \]
begin ~~~~ without tr.
begin ~~~~ with tr.
end ~~~~
begin tuplet
der tuplet
tuple has a bracket
bracket is continuous
(0 = broken)
number is inside
(0 = outside)
bracket is round
(0 = square)
tie is editorial (dotted)
~~~ is editorial
tuple is editorial

<table>
<thead>
<tr>
<th>ts(15) = slurflag bit set</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>start slur1 (new slur)</td>
</tr>
<tr>
<td>1</td>
<td>stop slur1 (from prev. note)</td>
</tr>
<tr>
<td>2</td>
<td>start slur2 (etc.)</td>
</tr>
<tr>
<td>3</td>
<td>stop slur2</td>
</tr>
<tr>
<td>4</td>
<td>start slur3</td>
</tr>
<tr>
<td>5</td>
<td>stop slur3</td>
</tr>
<tr>
<td>6</td>
<td>start slur4</td>
</tr>
<tr>
<td>7</td>
<td>stop slur4</td>
</tr>
</tbody>
</table>

for editorial slurs

| 16 | start slur1 (new slur)               |
| 17 | stop slur1 (from prev. note)          |
| 18 | start slur2 (etc.)                   |
| 19 | stop slur2                           |
| 20 | start slur3                          |
| 21 | stop slur3                           |
| 22 | start slur4                          |
| 23 | stop slur4                           |

for both kinds of slurs

| 8 | force slur1                          |
| 9 | 0 = up, 1 = down                     |
| 10 | force slur2                          |
| 11 | 0 = up, 1 = down                     |
| 12 | force slur3                          |
| 13 | 0 = up, 1 = down                     |
| 14 | force slur4                          |
| 15 | 0 = up, 1 = down                     |

for ties

| 24 | specify tie orientation              |
25 0 = overhand; 1 = underhand

ts(16) = subflag 1  

<table>
<thead>
<tr>
<th>bit</th>
<th>item</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-3</td>
<td>ornaments</td>
</tr>
<tr>
<td></td>
<td>0 = none</td>
</tr>
<tr>
<td></td>
<td>1 = turn</td>
</tr>
<tr>
<td></td>
<td>2 = trill(tr.)</td>
</tr>
<tr>
<td></td>
<td>3 = shake</td>
</tr>
<tr>
<td></td>
<td>4 = mordent</td>
</tr>
<tr>
<td></td>
<td>5 = delayed turn</td>
</tr>
<tr>
<td></td>
<td>6-15 (available)</td>
</tr>
</tbody>
</table>

4-9 accidental combinations  
with ornaments

| 4-6 accidental above ornament  |
| 7-9 accidental below ornament  |

Accidental code

| 0 = none  |
| 1 = sharp-sharp  |
| 2 = flat-flat  |
| 3 = sharp  |
| 4 = natural  |
| 5 = flat  |
| 6 = (not used)  |
| 7 = (not used)  |

10-13 dynamics

| 0 = none |
| 1 = p |
| 2 = pp |
| 3 = ppp |
| 4 = pppp |
| 5 = f |
| 6 = ff |
| 7 = fff |
| 8 = ffff |
| 9 = mp |
| 10 = mf |
| 11 = fp |
| 12 = sfp |
| 13 = sf |
| 14 = sfz |
| 15 = rfz |

14 upright fermata
15 inverted fermata
16 print note in cue size
17 editorial accidental
18 cautionary accidental

<table>
<thead>
<tr>
<th>ts(17) = subflag 2</th>
<th>bit</th>
<th>item</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>0</td>
<td>down bow</td>
</tr>
<tr>
<td>v</td>
<td>1</td>
<td>up bow</td>
</tr>
<tr>
<td>i</td>
<td>2</td>
<td>spiccato</td>
</tr>
<tr>
<td>.</td>
<td>3</td>
<td>staccato</td>
</tr>
<tr>
<td>=</td>
<td>4</td>
<td>line over dot</td>
</tr>
<tr>
<td>-</td>
<td>5</td>
<td>legato</td>
</tr>
<tr>
<td>&gt;</td>
<td>6</td>
<td>horizontal accent</td>
</tr>
<tr>
<td>A</td>
<td>7</td>
<td>vertical sforzando accent</td>
</tr>
<tr>
<td>V</td>
<td>8</td>
<td>vertical sforzando accent</td>
</tr>
<tr>
<td>o</td>
<td>9</td>
<td>harmonic</td>
</tr>
<tr>
<td>Q</td>
<td>10</td>
<td>thumb (*)</td>
</tr>
<tr>
<td>0</td>
<td>11</td>
<td>open string (0)</td>
</tr>
<tr>
<td>12-31</td>
<td></td>
<td>fingering (up to 5 numbers)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>12-14</th>
<th>first number</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>no number</td>
</tr>
<tr>
<td>1</td>
<td>finger 1</td>
</tr>
<tr>
<td>2</td>
<td>finger 2</td>
</tr>
<tr>
<td>3</td>
<td>finger 3</td>
</tr>
<tr>
<td>4</td>
<td>finger 4</td>
</tr>
<tr>
<td>5</td>
<td>finger 5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>substitution bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>no substitution</td>
</tr>
<tr>
<td>1</td>
<td>substitution</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>16-19</th>
<th>(second number, see 12 to 15)</th>
</tr>
</thead>
<tbody>
<tr>
<td>20-23</td>
<td>(third number)</td>
</tr>
<tr>
<td>24-27</td>
<td>(fourth number)</td>
</tr>
<tr>
<td>28-31</td>
<td>(fifth number)</td>
</tr>
</tbody>
</table>

ts(18) = used for sorting, later used to indicate position of virtual note head (for placing slurs and other articulations and signs). bit 24 set if modified

ts(19) = used for sorting, later used to indicate global x-offset for chord groups

ts(20) = index to ASCII tsdata

ts(21) = pass number

ts(22) = backtie flag (for regular, chord and cue notes)

| 0 | this note is not backward tied |
| # | this note is backward tied    |

Actually the BACKTIE flag has multiple uses.
(1) When the ts array is first being constructed, there may be a tie into this group of notes from a previous measure. In this case, a tiearr ROW element has already been constructed. The tiearr rows need to be searched and the proper one found. This index (+ INT10000) is then stored as the backtie flag.

(2) For all other row elements of the ts array, it is sufficient to store a back pointer to the ts row that originated the tie.

(3) When it comes time to process the ts array, three cases may be encountered.

(a) There is a non-zero backtie flag, and this flag is greater than INT10000. In this case, the backtie flag (- INT10000) points to a tiearr ROW element, and the tie may be processed.

(b) There is a forward tie from this note. In this case, the backtie flag has already been used to set a tie and the element is now free for other use. We can generate a new row element in tiearr, and place the pointer to this element in the backtie flag (the term "backtie" is now a misnomer).

(c) Now when we encounter a non-zero backtie flag in a new ts ROW, we know this points to a previous ts row, from which we can get the pointer to the relevant tiearr ROW in that ts(.BACKTIE).

For this method to work properly, it is necessary that backward ties be processed before forward ties. When a backward tie is processed it is important to set the backtie flag to zero.

\[
\begin{align*}
\text{ts(23)} & = \text{note duration (in divisions)} \\
\text{ts(24)} & = \text{increment distance flag} \\
0 & = \text{fixed distance (not to be modified by print)} \\
\# & = \text{variable distance; \# = time elapsed between this node and next node. (576 divisions = quarter note)} \\
\text{ts(25)} & = \text{virtual end of stem (bit 24 set if modified)} \\
\text{ts(26)} & = \text{editorial version of ts(16), subflag 1} \\
\text{ts(27)} & = \text{editorial version of ts(17), subflag 2} \\
\text{ts(28)} & = \text{staff number} \\
\text{ts(29)} & = \text{multi-track flag << 2 + mcat flag}
\end{align*}
\]

multi-track flag
0 = this note lies on a staff that has notes from only one pass (the simplest and most common situation).

1 = this note belongs to one of multiple passes on this staff and all notes on this pass have stems which point up

2 = this note belongs to one of multiple passes on this staff and all notes on this pass have stems which point down

3 = this note belongs to one of multiple passes on this staff and the notes for at least one of these passes have stem directions which are both up and down

mcat flag

0 = only one independent instrument represented in this measure (vflag = 1)

1 = more than one independent instrument (vflag > 1) but only one pass and without chords (either unison part, or single part)

2 = more than one independent instrument (vflag > 1) but only one pass but with chords (more than one part, but parts are isorhythmic)

3 = more than one independent instrument (vflag > 1) and more than one pass (two or more musically independent parts)

ts(30) = spacing parameter (1 <= spn <= 6913)
ts(31) = y position of object (saves time in proc. chords)
ts(32) = pointer to extra ts() row element for storing data on slurs. Elements 1-6 of new element are for storing global data on slurs entering and leaving the note. Elements 7-42 are taken in groups of three (expanded from two in 05/06/03 code revision), making a total of 12 such groups. Each group describes a slur entering or leaving this note. The first element in the group contains general information + the x-offset; the second element in the group contains the y-offset. The third element in the group contains the integer equivalent of the 4-byte print suggestion for the slur. See TS32 for more information.

ts(33) = node shift flag (positive and negative values)
ts(34) = tsr pointer

Case II: Figures

ts(1) = 9
ts(2) = division number (starting with 1)
ts(3) = number of figures in this chord
ts(4) = space parameter
ts(5) = first figure -- position one
ts(6) = first figure -- position two
ts(7) = first start/stop flag for continuation line
ts(8) = second figure -- position one
ts(9) = second figure -- position two
ts(10) = second start/stop flag for continuation line
ts(11) = third figure -- position one
ts(12) = third figure -- position two
ts(13) = third start/stop flag for continuation line
ts(14) = fourth figure -- position one
ts(15) = fourth figure -- position two
ts(16) = fourth start/stop flag for continuation line

figure field:  0 = blank
              1-19 = figure
              20 = +
              21 = x
              22 = 2+
              23 = sharp
              24 = 4+
              25 = 5+
              26 = 6\n              27 = 7\n              28 = natural
              29 = flat
              30 = short continuation line (-)

Adding 1000 to figure field (position one) indicates small parentheses around the field.
Adding 2000 to figure field (position one) indicates large parentheses this figure and the one below it.
Adding 3000 to figure field (position one) indicates large parentheses this figure and the two below it.
(Added 11/16/03)

start/stop continuation flag:  0 = none
                               1 = stop
                               2 = start
                               3 = continue

ts(20) = minimum space for figure group
ts(21) = pass number
ts(23) = figure duration in divisions (0 if not given)
ts(24) = increment distance flag (see notes)
ts(28) = staff number

Case III:  Bar Lines

ts(1) = 10
ts(2) = division number (starting with 1)
ts(3) = bar number (0 = none)
ts(4) = bar type
1 = regular 5 = double regular
2 = heavy 6 = regular-heavy
3 = dotted 9 = heavy-regular
10 = heavy-heavy

\( ts(5) \) = repeat flag

0 = no repeats 1 = forward repeat
2 = back repeat 3 = both repeats

\( ts(6) \) = backward ending flag

0 = no ending
# = ending number: positive = stop ending
    negative = discontinue ending

\( ts(7) \) = forward ending flag

0 = no ending
# = ending number

\( ts(8) \) = flags

<table>
<thead>
<tr>
<th>bit</th>
<th>set</th>
<th>clear</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>--------------</td>
<td>----------------</td>
</tr>
<tr>
<td>0</td>
<td>continue ~~~</td>
<td>stop ~~~</td>
</tr>
<tr>
<td>1</td>
<td>segno sign</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>fermata over bar</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>fermata under bar</td>
<td>0</td>
</tr>
</tbody>
</table>

\( ts(9) \) = space parameter (important for non-contr. bars)
\( ts(10) \) = number over previous measure: 0 = none

\( ts(20) \) = index to ASCII tsdata taken out
\( ts(21) \) = pass number
\( ts(28) \) = number of staves

Case IV: Signs, Words, Marks

\( ts(1) \) = type  sign = 11, words = 12, mark = 13
\( ts(2) \) = division number (starting with 1)
\( ts(3) \) = vertical position flag: 1 = below line
     2 = above line
\( ts(4) \) = sign number

0 = no sign
1 = segno
2 = ped
3 = *
4 = other letter dynamics
5 = D.S or D.C. (right justified string)
6 = fine (centered string)
7 = words (left justified string)  
8 = tie terminator  
(added 10-12-96)

\[ ts(5) = \text{super flag} \]

0 = no super-object  
1 = start wedge  
2 = stop wedge  
3 = start dashes (after words)  
4 = stop dashes  
5 = start 8ve up  
6 = stop 8ve up  
7 = start 8ve down  
8 = stop 8ve down  
9 = start 15 up  
10 = stop 15 up  
11 = start 15 down  
12 = stop 15 down  
13 = normal transposition (temporary flag)

\[ ts(6) = \text{parameter for words: optional font designation} \]

\[ ts(7) = \text{wedge offset} \quad \text{(for cases where a wedge begins after or stops at a letter dynamic)} \]

\[ ts(8) = \text{track number} \quad \text{(useful for multiple wedges, dashes or transpositions of the same type)} \]

\[ ts(9) = \text{spacing (for case of isolated mark)} \]

\[ ts(10) = \text{parameter for wedges: wedge spread} \]

\[ ts(11) = \text{parameter for musical directions which are objects: position shift} \]

\[ ts(12) = \text{special flag for case where this element is isolated on a division (possibly with other members of this same group).} \]

\[ ts(13) = \text{parameter for musical directions which are super-objects: position shift} \]

\[ ts(20) = \text{index to ASCII tsdata} \]

\[ ts(21) = \text{pass number} \]

\[ ts(22) = \text{backtie flag (for tie terminators)} \quad \text{(added 10-12-96)} \]

\[ ts(28) = \text{staff number} \]

Case V: Clef change in middle of a measure

\[ ts(1) = \text{type} = 14 \]
\[ ts(2) = \text{division number (starting with 1)} \]
ts(3) = clef number
ts(4) = clef font number
ts(5) = transposition flag:

   1 = notes written octave higher than sound
   0 = notes written at sound
  -1 = notes written octave lower than sound

ts(6) = position on staff
ts(9) = space parameter
ts(20) = index to ASCII tsdata
ts(21) = pass number
ts(28) = staff number

Case VI: Time designation in middle of a measure

   ts(1) = type = 15
   ts(2) = division number (starting with 1)
   ts(9) = space parameter
   ts(20) = index to ASCII tsdata
   ts(21) = pass number
   ts(28) = staff number

Case VII: Meter change in middle of a measure

   ts(1) = type = 16
   ts(2) = division number (starting with 1)
   ts(3) = time number (100 time numerator + denominator)
   ts(9) = space parameter
   ts(20) = index to ASCII tsdata
   ts(21) = pass number
   ts(28) = number of currently active staves

Case VIII: Change in number of divisions per quarter

   ts(1) = type = 17
   ts(2) = division number (starting with 1),
   ts(3) = divisions per quarter
   ts(9) = space parameter
   ts(20) = index to ASCII tsdata
   ts(21) = pass number

Case IX: Change in key signature

   ts(1) = type = 18
   ts(2) = division number (starting with 1)
   ts(3) = new key signature
   ts(4) = old key signature
   ts(9) = space parameter
   ts(20) = index to ASCII tsdata
   ts(21) = pass number
   ts(28) = number of currently active staves