DTMF Tone Generation. An Implementation using the TMS320C2xx

Literature Number: BPRA068
Texas Instruments Europe
October 1997

IMPORTANT NOTICE

Texas Instruments (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Certain applications using semiconductor products may involve potential risks of death, personal injury, or severe property or environmental damage ("Critical Applications").

TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

Inclusion of TI products in such applications is understood to be fully at the risk of the customer. Use of TI products in such applications requires the written approval of an appropriate TI officer. Questions concerning potential risk applications should be directed to TI through a local SC sales office.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used.

Contents

DTMF Tone Generation.

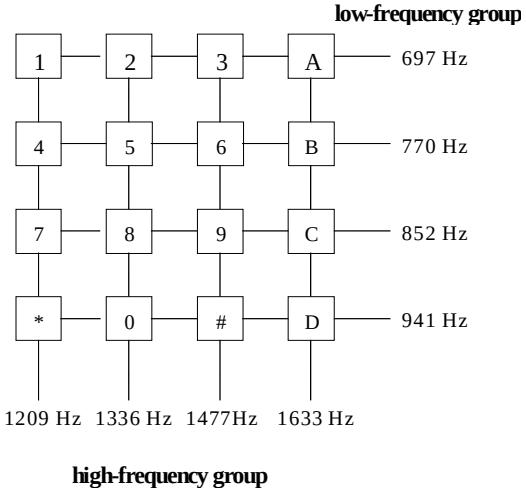
An implementation using the TMS320C2xx

ABSTRACT

This application report deals with the implementation of a dual-tone multiple frequency (DTMF) tone generator on a TMS320C2xx DSP. It describes, in detail, the algorithm, the way of using the routines which have been made for a multi-channel environment. Users can also find information on the process performance and its speed and memory requirements.

1. Introduction

A DTMF transmitter (encoder) generates a composite audio tone burst which comprises two frequencies f_l and f_h that are not harmonically related. Furthermore, the choice of DTMF frequencies has been guided by the need to avoid between a dialling tone and a conversation flow. For this reason, the probability of finding one of the possible combinations of frequencies and their levels in a conversation (or a room noise) is statistically extremely weak. The telephone keypad may be used to illustrate the sixteen possible DTMF frequency combinations representing sixteen separate digits.



The signal generated by a DTMF encoder is a direct algebraic summation, in real-time of the amplitudes of two sine (or cosine) waves of different frequencies.

$$x(t) = A_l \cdot \sin(2\pi f_l t) + A_h \cdot \sin(2\pi f_h t) .$$

2. DTMF generation recommendation

The **CCITT Q.23** has defined the requirements for a central office DTMF receiver to ensure reliable operations. For example, the receiver must tolerate slight variations (frequency bandwidths) in the eight frequencies and the relative signal amplitudes (twist) of the two frequencies comprising a valid digit. Also, the tone bursts must meet certain timing criteria such as on-off duration etc. The receiver must also reject speech signals and operate in the presence of certain noise levels, without incorrectly decoding the tone pairs.

2.1 Coding

Signalling frequencies are chosen in two groups of distinct frequencies in the range from 300 Hz to 3400 Hz. A signal is made up of one, and only one, frequency from each group which are simultaneously transmitted on the line.

Low Group	High Group
697 Hz	1209 Hz
770 Hz	1336 Hz
852 Hz	1477 Hz
941 Hz	1633 Hz

We can notice that they are, indeed, not harmonically related.

The tones are assigned as follows:

Keypad	Low frequency (Hz)	High frequency (Hz)
0	941	1336
1	697	1209
2	697	1336
3	697	1477
4	770	1209
5	770	1336
6	770	1477
7	852	1209
8	852	1336
9	852	1477
*	941	1209
#	941	1477
A	697	1633
B	770	1633
C	852	1633
D	941	1633

A typical DTMF receiver uses a special-purpose decoder chip (or chip set) to perform the decode function. Besides performing the main task of DTMF decoding, the TMS320C2xx can also perform a host of other telecom functions.

2.2 DTMF Transmitter

2.2.1 2.1 Frequency errors

Transmitted frequencies are in the range of ± 1.5 % of their nominal value.

2.2.2 Power levels

Transmission levels at a resistance of 600 Ω

1st option -9 dBm ± 2 dB for the high tone
-11dBm ± 2 dB for the low tone

2nd option -6 dBm ± 2 dB for the high tone
-8 dBm ± 2 dB for the low tone

The high-frequency has to be 2 ± 1dB above the low one. This 'offset' makes it possible to anticipate the attenuation undergone by the high frequencies in comparison with the low-tones on the telephone lines. The generator will transmit DTMF signal with **a 3 dB twist, and with -6dBm for the high tone and -9dBm for low (2nd option)**.

Parasitic signals must be 20 dB less energetic than the low-tone.

Furthermore, noise must fall within limits in relation to where it lies in the band. In particular, we ensure the specification in the band from 300 Hz to 4300 Hz, where it has to be less than -33 dBm.

2.2.3 Timing

Tone duration specified by the Q.23 recommendation is at least 65 ms. Pause duration (a pause is a -80dBm level signal, encountered between two digit) must be of at least 65 ms.

3. Data Format

3.1 Number representation

The fixed point mode consists of fixing the position of the binary comma in the binary word. The first bit of the word is dedicated to the sign expression. The addition of two fixed-point numbers on N bits is not erroneous but it requires an $(N+1)^{th}$ bit. By contrast, the multiplication of these words involves a rounding or a truncation error.

- The sign-magnitude representation divides the word into two fields: the sign field on one bit and the absolute value field on the (N-1) other bits.
  For instance, on 5 bits 0.375 becomes 0.0110
  -0.375 becomes 1.0110
  Thus, an N-bit word can take 2^N-1 different values. The zero is twice (0.0000 or 1.0000 on 5 bits). Furthermore, the sign-magnitude method is not really suitable for computations (particularly for additions).
- The individual inversion of all bits in the word can provide a number inversion. It is known as the 1's-complement method. Here, on N bits, 2^N-1 values are free. Keeping the last example, we have :
  0.0110 for 0.375 and
  1.1001 for - 0.375
- Using this method, one value is lost because of the double representation of zero (0.0000 and 1.1111 on 5 bits). To avoid this, another representation is generally used : the 2's complement method. To change the sign of a number, all bits are inverted and a 1 is added. And, so, 2^N combinations of an N-bit word become usable.
  Example: 0.375 is given by 0.0110 and
  -0.375 is given by 1.1010

3.2 Data sent to the D/A converter (AIC)

The AIC (TLC32046), which carries out the A/D and D/A conversions, receives input voltages between -3V and +3V. It quantifies those data on 16 bits at an 8kHz sampling rate. Therefore, the 2 LSB are reserved for dialogue between the DSP and the AIC.

$\begin{cases} x_q : \text{quantified data on a 16 bits length word with the 2's complement method.} \\ x_{an} : \text{corresponding analog input/output} \end{cases}$

$x_q = \frac{x_{an}}{V_{max}} \times D_{max}$ with, according to DSP feature, $\begin{cases} D_{max} = 32767 = 2^{15}\text{-}1 \\ V_{max} = 3V > x_{an} \end{cases}$

The MSB contains sign information. That format is usually called Q.15 format.

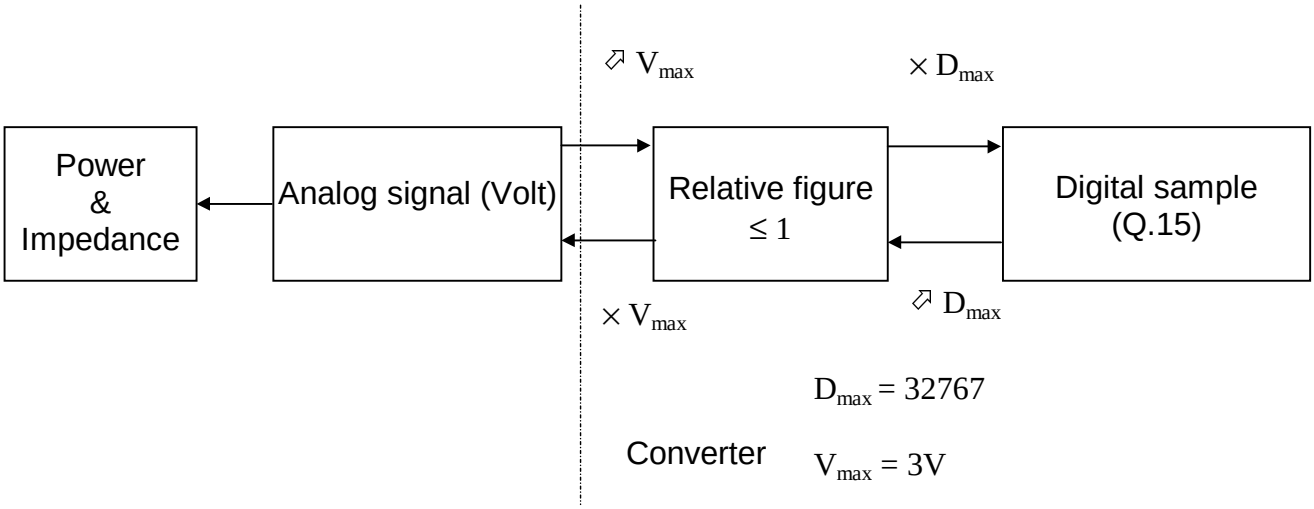
$x_{an} / V_{max} =$

S	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}	2^{-15}

$x_q =$

S	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

3.3 Conclusion



If Vmax from your converter is different, you would have to modify - by a multiply inside the routine, for example - values of transmitted samples.

4. Generator algorithm

The sinusoidal function, required by the recommendation, may be created by means of an oscillator.

4.1 Background on harmonic resonator

This is a direct implementation of the Z-transform of a direct sine function, sin(n.w.T), where T = sample rate and w= frequency to be generated in radians.

Let $h(n)=A.sin(n.w.T)$ *and* $x(n)=\delta(n)$, delta function.

$$Y(z)= H(z).X(z)$$

$$Y(z) = \frac{z.A.\sin(\omega.T)}{z^2 - 2.z.\cos(\omega.T)+1} \,.\, \delta(z) \quad \text{with } \delta(z) = \{1,0,0...\}$$

which leads to the equation:

$y(1) = 0$
$y(2) = A.\sin(wT)$
$\forall\ n \in [\ 3\ ,\ +\infty\ [\quad y(n) = 2.\cos(w.T).y(n-1) - y(n-2)$ because $x(n)=0$

4.2 Coefficient computations

Each coefficient is fractional, so the Q.15 format is useful to quantify it. “A” is the maximum value of the sine function:

- if frequency = 697, 770, 852, 941
  -9 dBm => V=0.3887 on 600Ω
  Vmax (from the A/D converter) = 3V
  => A= 0.3887/ 3
- if frequency = 1209, 1336, 1477, 1633
  -6 dBm => V=0.5490 on 600Ω
  Vmax (from the A/D converter) = 3V
  => A= 0.5490/ 3

$$P_{dB} = 10\log\left(\frac{U_{eff}^2}{1000 \times Z_0}\right) \text{with } Z_0 = 600\ \Omega$$

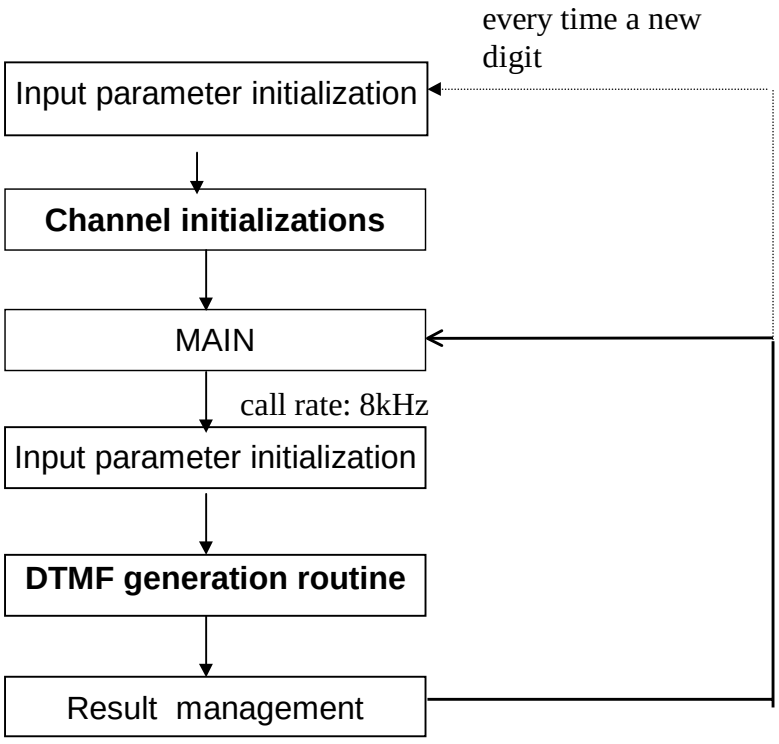
$$U_{eff}^2 = \frac{U^2{}_{\max}}{2}$$

Binary amplitude of each sine.

frequency	*cos(2 π f / fs) . 2^15*	*A . sin(2 π f / fs) . 2^15*
697	27980	2210
770	26956	2414
852	25701	2634
941	24219	2860
1209	19073	4876
1336	16325	5200
1477	13085	5498
1633	9315	5749

4.3 Process overview

This description is suitable for a single channel evaluation.



The following routines depend on your application, your way to manage a multi-channel environment:

- MAIN

- Input parameter initialization
- Result management

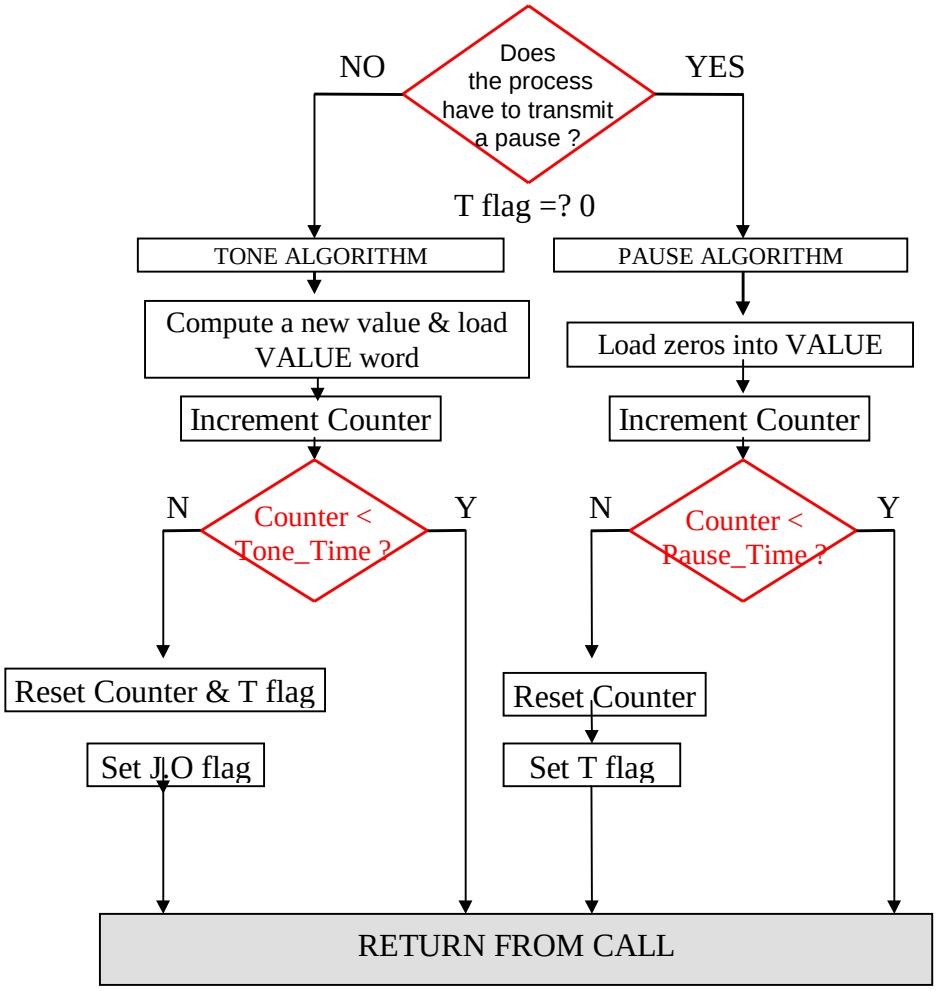
The other ones are given in _DTMF_GEN.asm file.

4.4 DTMF generation procedure

4.4.1 General overview

The channel initialization routine is described in “Program Organization” chapter. Flag definitions:

- T ⇔ Tone flag; if T=1 the process must compute a tone sample. else it must create a pause sample (0V)
- J.O ⇔ Job Over flag; by setting J.O, the process announces that the DTMF signal was totally transmitted and could then be initialized to transmit another digit.

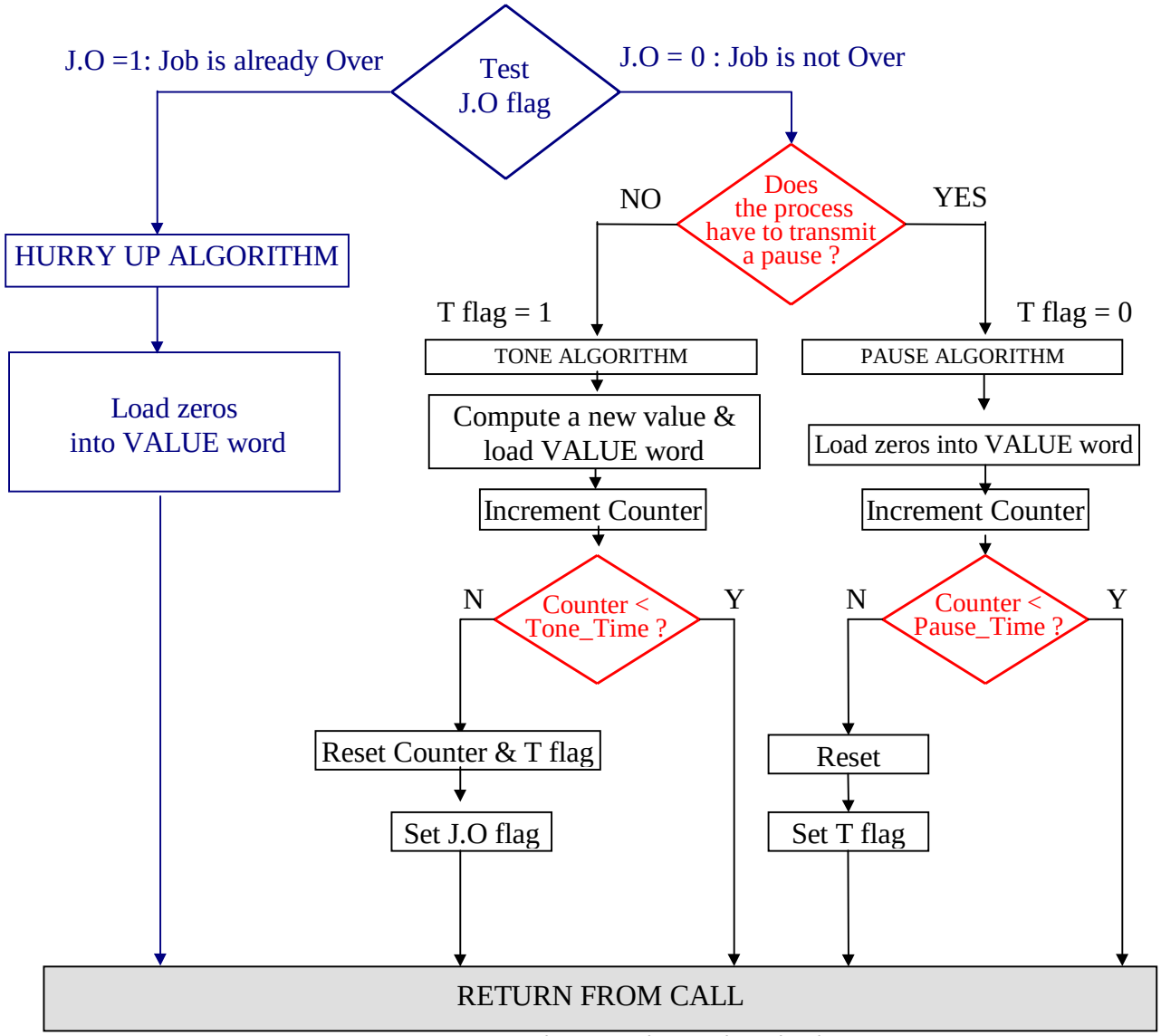


What will happen if the result manager does not, "in time", re-initialize the channel for a new digit transmission ?

It will transmit the same digit. As a result, the general scheme had been enhanced to avoid this trouble.

4.4.2 Enhanced scheme

After a digit transmission, if the channel is not re-initialized in time – which can happen when the new digit to transmit is unknown by the system manager– the process will transmit an unlimited pause, until J.O flag clears. Here is the new flow chart:



IMPORTANT NOTICE: The process always load VALUE with (y_low + y_high)(n), so (y_low + y_high)(1) and (y_low + y_high)(2) won't be transmitted.

5. Data Memory Organization

5.1 Requirements

The process needs two kinds of variables: the 'global' ones and the 'local' ones.

The 'global' variables are variables which do not depend on the channel under investigation: it could be resonator coefficients, or initial values, for example. These are shared between all the channels.

The 'local' variables are the variables specific to a channel: their values vary from one channel to another. They are useful to store an individual channel state, its filter nodes.

5.2 Global variables

These are allocated via a section called *'const'.*Some of them are constants such as resonator coefficients, while others are temporary variables such as TEMP, DIGIT.

Temporary variables are unitialized but users need not initialize them:

TEMP, DIGIT, _VALUE, CHANNEL_add are always overwritten during the process.

Why constants have not been declared as '.set' ?
In fact, you can change 'const' section by declaring SOME constants as '.set' and allow only temporary variables in this section. But if you decide to modify constant values, during algorithm evaluation for example, you will have to assemble and link again: actually, you will do it every time you wish to perform a change. To allow constants as .word xxxx permits you to change their figures by filling the right memory space in debugger tools, without having to assemble and link at every change.

If you decide to declare ALL constants as .set, you will have to adapt 'MPY' instructions, because they have 13-bits operands !

This section must be declared in your linker file and cannot be removed from 'DTMF_gen' file: the 'DTMF_gen' routine used both indirect and direct addressing modes when it calls variables from 'const' section.

This section cannot cross pages. It must be loaded in a page but in any one you choose.

5.3 Local variables

They are used **to store individual channel variables** such as COUNTER, coefficients currently used.

They do not need to be explicitly declared. It depends only on your channel management!

It was decided to create the ‘chan’ section and to declare it in mainC2xx.cmd and _DTMF_GEN.asm because these made for easier initialization of ‘_DTMF_GEN’ and ‘_INIT_CHAN’ input parameters.

The ‘chan’ section is used in a single channel environment, it would be tedious and unnecessary to create sections for each channel in a multi-channel environment.

The individual channel variables must be INITIALIZED before every DTMF digit transmission. A routine called ‘_INIT_CHAN’ had been created to do it easily (see program organization chapter for more information).

Space for a channel can cross pages: DP during ‘_DTMF_GEN’ does not depend on the channel under investigation.

5.4 Conclusion

Memory requirement for ‘const’ and for each channel is specified in another chapter.There is no constraint about where individual channel parameters must be saved: it can cross pages and begin anywhere: it depends on your free memory space. **You only have to**:

- declare in your files ‘const’ section and be sure that it does not cross pages.
- initialize a channel variables before DTMF digit generation.

5.5 Tables

base+... = address	length	Data name	Access type	Description
0h	1	STATUS	indirect via AR2	Contains channel state (its format is described below)
1h	1	coef_low	“	Is the lowband resonator coefficient currently used.
2h	2	Y1_low Y2_low	“	Is the lowband resonator delayed nodes y(n-1) and y(n-2).
1h	1	coef_high	“	Is the highband resonator coefficient currently used.
2h	2	Y1_high Y2_high	“	Is the highband resonator delayed nodes y(n-1) and y(n-2).

STATUS format:

BIT

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COUNTER														J.O	T

COUNTER counts the number of transmitted tone or pause samples. J.O & T flags are described in the previous chapter.

base+... = address	length	Data name	Access type	Description
0h	8*2	tab	indirect via AR3	Contains resonator coefficients and initial values for each DTMF frequency.
10h	1	CHANNEL_ add	DMA	Is used to store complete 16 bits address of channel base.
11h	1	DIGIT	DMA	Contains temporarily the digit to transmit.
12h	1	_VALUE	DMA	The process will load this word with the sample generated.
13h	1	TEMP	DMA	This is a temporary useful word.

6. Program Organization

Two routines are given into '_DTMF_GEN.asm' file:

- '_INIT_CHAN'
- '_DTMF_GEN'

The following description explains what these programs are supposed to do and what their input parameters and outputs are. At the end of this chapter is a brief explanation of how the packages –mainC2xx (.asm&.cmd), and mainC5x (.asm& .cmd)– work.

6.1 '_INIT_CHAN'

Before generating a digit, the channel memory space must be initialized.

- Its STATUS must be loaded with 0000h. (the job is not yet over as a Pause must first be sent and COUNTER filled with zeros)

- coef_low and coef_high from the lowband and highband resonators respectively must be found in the “KEY” table which had been loaded in program memory space.
- initial values of delayed nodes must also be found.
  ‘_INIT_CHAN’ has been created to facilitate this.

You have to specify the channel you want initialize by loading the start address of that space into AR2 and loading the digit to transmit into DIGIT word from the “const” section. Then you can call _INIT_CHAN.

Example:
I have to manage 2 channels.
I know that each channel requires 7 words and I’d like to use:
- 0x0100 to 0x0106h for the first channel.
- 0x087d to 0x0882h for the second one.

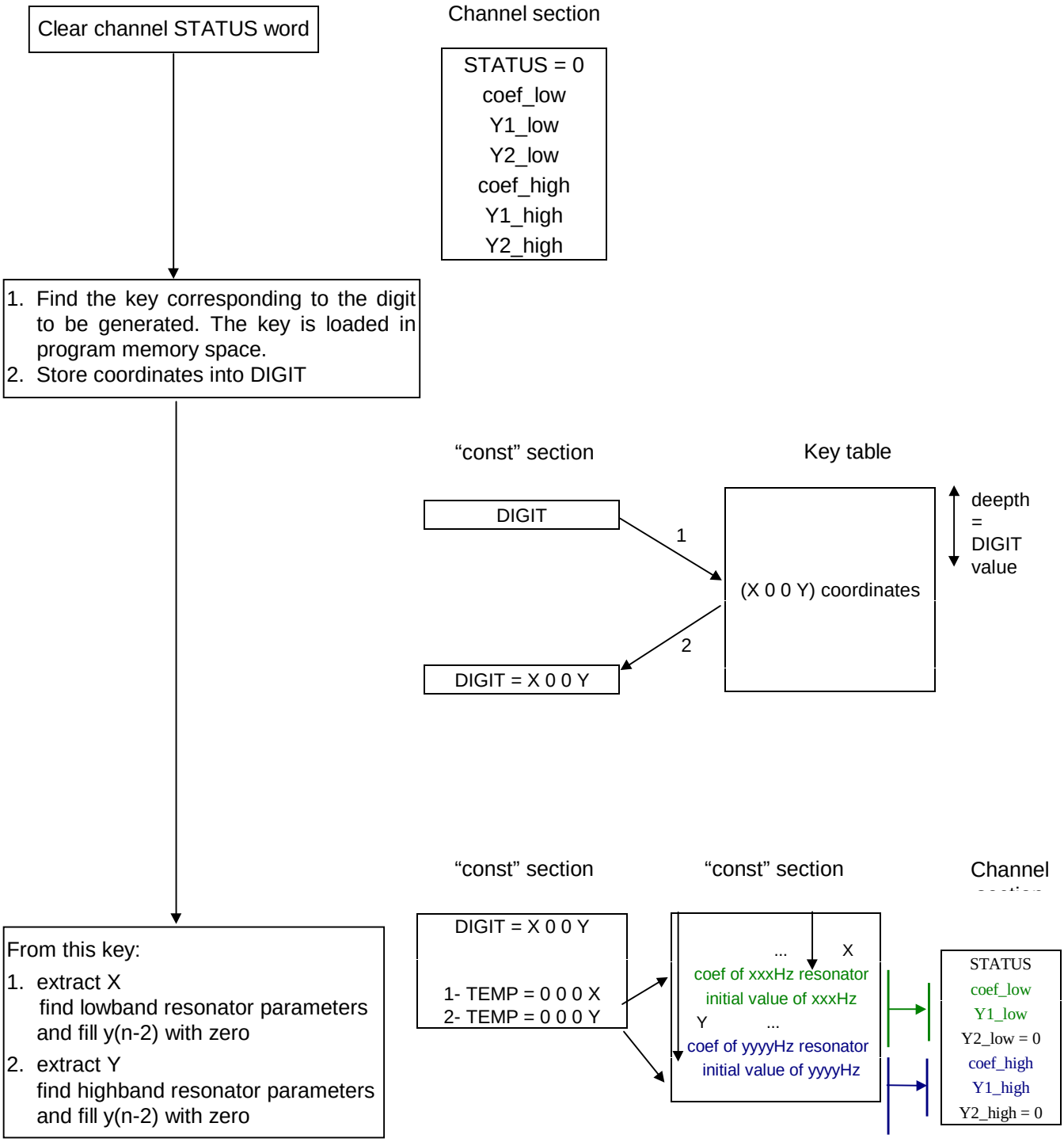
I would use ‘_INIT_CHAN’ like this:

```
.ref _INIT_CHAN
.
.
.
LAR  AR2,#0100h
LDP  #DIGIT
LACL #1        :to transmit digit 1 from the keypad
CALL _INIT_CHAN
LAR  AR2,#087dh
LDP  #DIGIT
LACL #2
CALL _INIT_CHAN
.
.
RETE
```

You can see that memory space reserved for channel 2 crosses page 18. As ‘Data memory’ organization explains, this is not forbidden

You have to call ‘_INIT_CHAN’, before ‘_DTMF_CHAN’, every time a new digit is sent.

_INIT_CHAN synoptic



6.2 '_DTMF_GEN'

This is the DTMF generator routine. Its outline is given in previous chapter. It has a single input parameter: the channel to evaluate.

You have to:

- save the processor state (ST0, ST1) and save the contents of AR2.
- load AR2 with the lowest address of channel space location you want to work with.

Then you can call it.

For example*:*
if 0x0100 to 0x0106 of data memory space is reserved for the channel we want to evaluate we'll do:

```
; save AR2
; save ST0,ST1
.
.
LAR AR2,#0100h
CALL_DTMF_GEN
```

How does one read the result of '_DTMF_GEN'?
Results are stored into accumulator and _VALUE from 'const' section.
If Acc==0, the new value has been loaded into _VALUE word.
If Acc!=0, the whole of necessary samples have been generated and _VALUE contains the last DIGIT sample to transmit.
If Acc!=0, and if the channel is not immediately RE-INITIALIZED, the process will generate a pause (0 into _VALUE) and still return (via accumulator) a figure not equal to zero, until the process re-initialization.

Example:

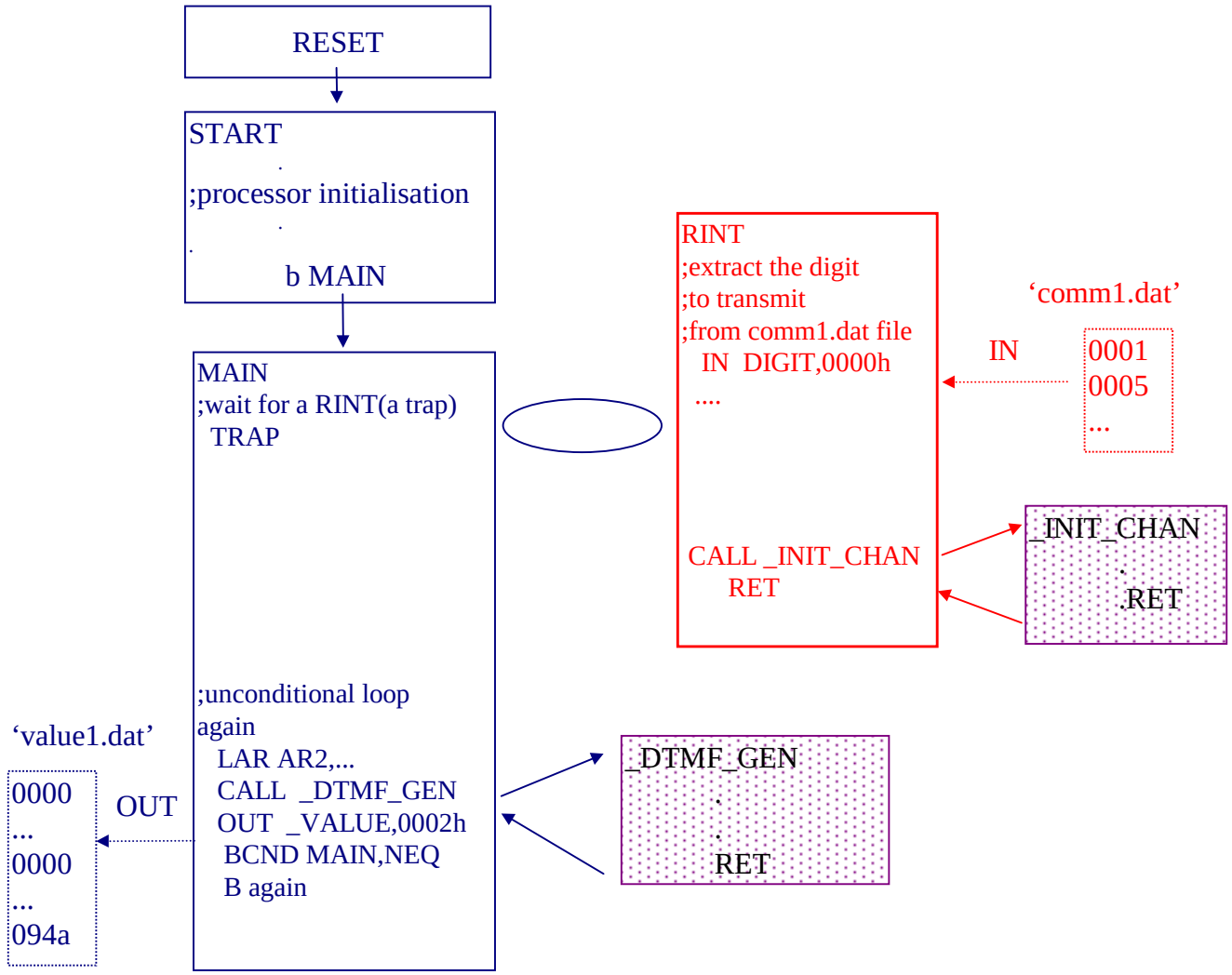
```
   .global _DTMF_GEN

   LAR  AR2, #channel
   CALL _DTMF_GEN

   ;Transmit _VALUE
   ;send the computed sample to serial port
   ;DP is still equal to #_VALUE
   LAR  AR2,_VALUE
   LDP  #DXR
   NOP
   SAR  AR2,DXR
   BCND other_digit,NEQ
   ...
other_digit
   ;this section depends on your system management.
```

6.3 About amainC2xx and amainC5x files ...

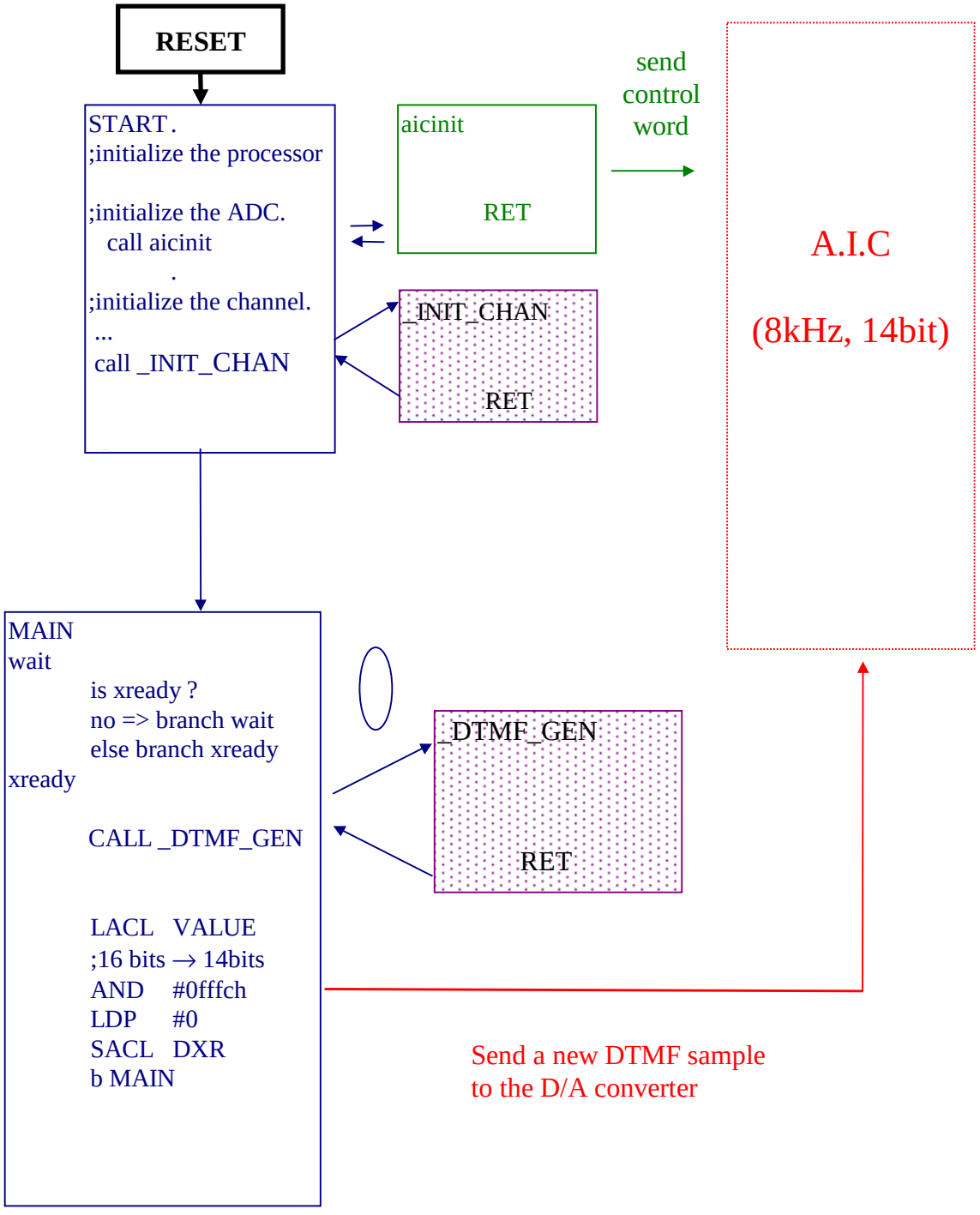
'_DTMF_GEN.asm' can be linked with amainC2xx.asm via amainC2xx.cmd, or with amainC5x.asm via amainC5x.cmd.

By using amainC2xx(.asm .cmd), '_DTMF_GEN.asm' code can be implemented in the *'C2xx Simulator* while amainC5x(.asm .cmd) allows users to use and test '_DTMF_GEN.asm' code with the *'C50 EVM*.

6.3.1 amainC2xx



6.3.2 mainC5x -transmission of a single tone



7. Memory space and MIPS required

7.1 Memory requirements

Section name	Type	Length (in words)
'init_chan'	routine	52
'dtmf_gen'	routine	71
'const'	variables & constants	20
key table	variables	16
for channels	variables	7 per channel
		TOTAL = 159 + 7.n

7.2 Cycle & Mips Requirements

Routine name	cycles	averaged call number	MIPS
_INIT_CHAN	52	1/1040	0.0004
_DTMF_CHAN Pause Algorithm: 1→519th 520th Tone Algorithm: 521→1039th 1040th Hurry up : 1041→+∞ +∞ = until you start transmitting a new digit	41 42 56 60 26	519/1040 1/1040 519/1040 1/1040 xxxx	[(41+56).519+(42+60)]* 8000 1040 = 0.388
			TOTAL = 0.39 MIPS

8. Conclusion

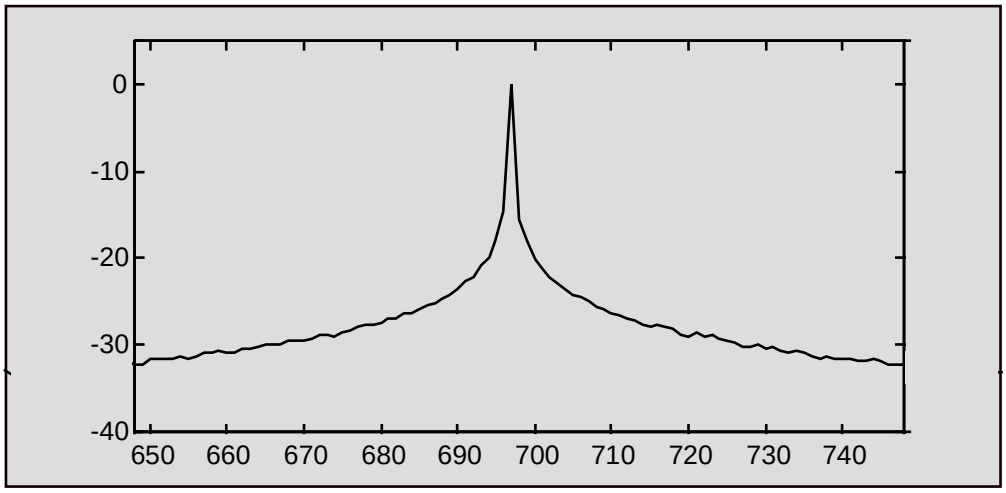
Powers of accepted frequency components were theoretically set as if they were a pure sine function. Actually, the generated signals contain noise: as a result, DTMF components would be less powerful.

Added noise seems to meet the Q.23 recommendation: its relative level is -20dB less than the accepted components.

The following figure is the fft of the signal generated by the 697Hz cell (notice: this cell contains more noise than the others, due to MATLAB).

number of computed samples: 8000
x: frequency
y: relative power
computation performed with MATLAB:

- % signal = generated by a cell (with Q.15 format)
- power = abs(fft(signal)); % | fft(signal)|
- relative_power = power/max(power)



Of course for each cell, the most powerful component is the theoretical one (697Hz component for the 697Hz cell ...).

Appendix A: Calling the routines from C environment

The routine names have been underscored in order to be easily identified from the C environment.
In the same way, VALUE has also been underscored, becoming_VALUE.

A.1 How activate the C compatibility of these routines?

This is done by setting the assembly constant “C_compatibility” low or high. This constant is declared before variable and routine declarations, at the top of detect.asm file.

- activate C environment => C_compatibility .set YES
- disable C environment => C_compatibility .set NO

Some sections are conditionally assembled: this condition is a function of “C_compatibility” setting.

A.2 Routine C prototypes

A.2.1 _DTMF_GEN

prototype: int DTMF_GEN (int)
input: this must contain the channel memory space base address.
output: if 0, it means the digit is not totaly transmitted..
otherwise it means another digit can be generated.
The generated sample is always stored into _VALUE word.
declaration into C files:

```
extern int DTMF_GEN(int);      /* (WITHOUT UNDERSCORE) */
extern int VALUE;
```

A.2.2 _INIT_CHAN

prototype: void INIT_CHAN (int,int)
input: int from the left side: this must contain the channel memory space base address.
int from the right: this must contain the digit to transmit.
output: none.
declaration into C files:

```
extern void INIT_CHAN(int,int);      /* (WITHOUT UNDERSCORE) */
```

A.3 Conventions performed by the called routines

_DTMF_GEN and _INIT_CHAN respect conventions of C called functions;

- do not modify AR0,AR1,AR6,AR7
- return value in the accumulator.

For information: PMST is modified

AR2,AR3 are used as local pointers, so their contents are modified **but not restored**.

A.4 MIPS modification

_INIT_CHAN needs 6 cycles more.

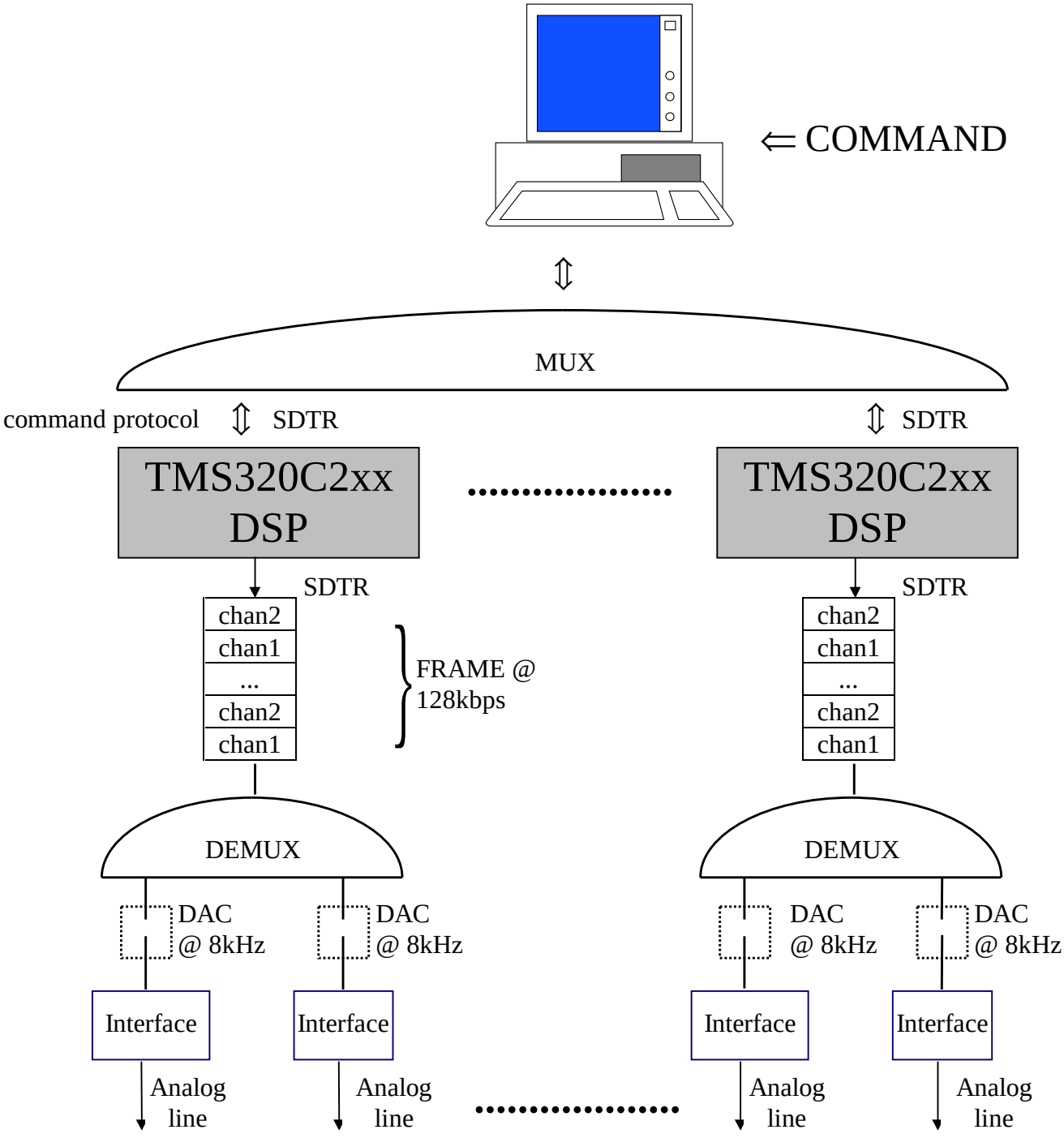
_DTMF_GEN needs 3 cycles more.

As a consequence, required MIPS remain the same.

Appendix B: A multichannel management example

B.1 Context

The DSP is used only to perform generation on 2 channels (for example): another system would provide channel management. The DSP and the 'manager' would communicate via the serial port.



B.2 DSP code example for simulator in C language

Two memory space locations are first defined: 0x0200h for channel 1
0x0200h+7 for channel 2

Reset routine: 1- Initialization of C environment.

main routine: 1- Initialization of channels(via RINT routine).
2- Then the process, indefinitely:
calls , DTMF_GEN.
analyzes the result (if result $\neq 0$, another digit can be transmitted, call RINT).

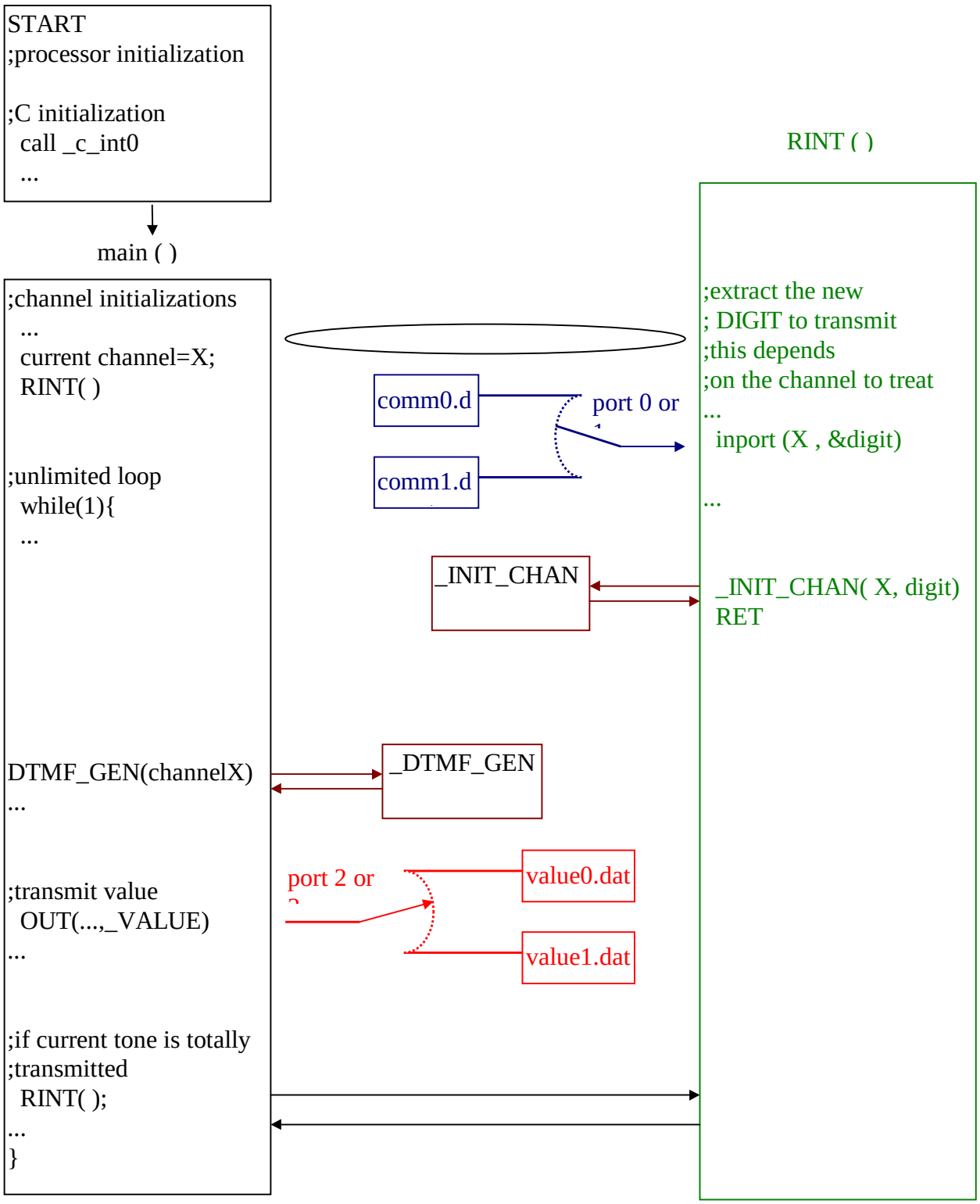
RINT routine: 1- Extraction of a digit from comm0.dat or comm1.dat file.
2- Call of _INIT_CHAN for channel 0 or 1.

Here is cmain2xx.c:

```
#include "ioports.h"

extern      void INIT_CHAN (int,int);
extern      int VALUE;
extern      int DTMF_GEN (int);
#define chan_nb 2
int channel[chan_nb]={0x200,0x200+7};
int current=0;  /*it can also be called current channel number */

main(){
      int out;
      int i;
      /* Initialize channels for digit transmissions */
      for(i=0;i<chan_nb;i++){
            RINT();
            current++;
      }
      current=0;
      /* Now the main process can start */
      while(1){   /* unconditional loop */
            out=DTMF_GEN(channel[current]);
            outport(current+chan_nb,VALUE); /* transmit the computed value */
            if (out != 0)
                 RINT(); /* the current channel must be reinitialized */
            current++;
            if (current == chan_nb)
                  current=0;  /* current={0,1,...,chan_nb-1}
      }
}


RINT(){
      unsigned int digit;
      inport(current,&digit);
      INIT_CHAN(channel[current],digit);
```

B.3 Block diagram

2 channels are managed.



References

1. Alan V. Oppenheim, Ronald W. Schafer, *Discrete-Time Signal Processing,* Published by Prentice-All, Inc. A Division of Simon & Schuster, Englewood Cliffs, New Jersey, 1989
2. Tim Massey and Ramesh Iyer, *DSP Solutions for Telephony and Data/Facsimile Modems*, Application Book, Texas Instruments Inc., 1997
3. Pascal DORSTER (TI), *Sine, Cosine on the TMS320C2xx,* Application Report, Texas Instruments Inc., 1996