Unitext V2.0 Rev B - Port Level Programmer Specification - DRAFT ONLY

Introduction:

The Unitext teletext card can be configured to your choice of 4 port base addresses. The card utilises the next 8 addresses above this base address. The port ranges are:

318h-31Fh
328h-32Fh
338h-33Fh
348h-34Fh

The 8 ports starting with the base port control all of the features of the Unitext card and are mapped to 8 registers on the card. Version 2.x backwardly port compatible with V1.x. The tuner support is provided via extra read and write bits in the Colour Burst Blanking Width register. VBI line selection are multiplexed over the V1.x registers 4 & 5 and selected by setting one bit in the reset register port (details to follow).

The card will run in two modes. Single magazine mode and all magazine mode. In Single magazine mode, the card will only allow pages from a preset magazine through to the buffer. This is extremely useful when the card is being used in a slower machine (386sx20 or slower) as the CPU overhead needed to run the card is far lower. In All magazine mode the card passes all information that is read through to the buffer including all packet and teletext data. This requires more CPU time but allows a software driver to be constructed that can handle throughput from all teletext magazines as well as packet 31 at the same time.

Although the card has an 8Kb buffer installed, it is only able to address the bottom 2Kb in this version. This can cause some problems when developing software to service the card. The 2K FIFO requires servicing every 20ms to ensure that data overflows do not occur. This rate is normally too fast for the IBM PC's (in all magazine mode) 8253/8254 programmable interval timer when running at the standard DOS timer interrupt rate of 54.95ms (18.2/s). Although this timer can be sped up and have the driver software emulate the standard rate to all chained software (As TTSR.COM does in timer mode), it is necessary to use the IRQ generation features for boot time installed drivers.

BBC1	551250	~~[illegible]~~
BBC2	519250	
ITV	495250	
CH4	471250	
CH5		

495
519
551

[illegible]
CS4237

Unitext V2.0 Rev B - Port Level Programmer Specification - DRAFT ONLY

The Unitext card will generate an interrupt after every VBI whether there is data available or not. The card has jumpers to adjust the IRQ to either IRQ-2,5,7,10,11,12 or 15 (10,11,12 & 15 only available if installed in 16 bit slot). Individual bits in a status register (covered below) indicate whether the card has data available and in fact whether that card actually generated the interrupt. Because of this last feature it is possible to slave hardware onto the same hardware interrupt (provided the driver software is loaded last) as if the interrupt bit is not set then the previously installed IRQ ISR can be called to service the slaved hardware.

The Unitext card has many adjustable features including everything from frame timing through to magazine selection. The major adjustable features are listed below:

* Operation mode (All/Single Magazine)
* Programmable VBI line selection (line 6-21)
* Teletext Magazine (Single Mag Mode Only)
* Hardware level Packet 31 Channel Selection
* Positive / Negative edge triggering.
* Packet length   (teletext row or P31 length)
* Framing Code Value
* IRQ operation ON/OFF
* Max synchronisation delay (for run in & framing)
* Maximum colour burst blanking signal
* Software tuner control via I2C bus
* Software tuner feedback via I2C bus

Using these features together in software can allow a software driver to be written that will run the card under various conditions and for various operations without modifying or configuring hardware other than through the software interface.

Unitext V2.0 Rev B - Port Level Programmer Specification - DRAFT ONLY

The read and write registers are arranged so that some are single purpose and some have operations controlled by individual bits of the port.

A quick guide to the write port uses follows:

			Port if base=318h
R0	-	Mode select.	318h
R1	-	Teletext Magazine Selection (Single Mag Mode Only).	319h
R2	-	Unused (Must Be 07h).	31Ah
R3	-	Teletext/Packet Width.	31Bh
R4	-	Framing Code Value.	31Ch
	*	VBI line select (lines 6/319 - 13/326)	
R5	-	Synchronisation Delay.	31Dh
	-	Negative / Positive edge triggering	
	*	VBI line select (lines 14/327 - 21/334)	
R6	-	I2C, Miscellaneous & Reset	31Eh
	-	CIDAC Reset.	
	-	I2C Bus clock	
	-	I2C Bus data	
	-	Port bank switch selection	
	-	IRQ On/Off.	
R7	-	Colour Burst Blanking Width.	31Fh

* denotes selection based on the setting of bit 6 of R6.

Unitext V2.0 Rev B - Port Level Programmer Specification - DRAFT ONLY

A quick guide to the read port uses follows:

			Port if base=318h
R0	-	Status.	318h
R1	-	Teletext Data.	319h
R6	-	Interrupt generation bit, I2C Bus data & clock	31Eh

Unitext V2.0 Rev B - Port Level Programmer Specification - DRAFT ONLY

Unitext detects teletext by blanking the colour burst and waiting for a valid clock run-in then searching for a framing code to signify the start of data. All aspects of this operation is adjustable by software on the fly however some defaults are recommended for general compatibility. These defaults are listed beside each register discussion (where applicable).

Write Registers

R0 - Mode select. (Def 06h/6dec)

Bits 0 - 2 Mode select.

The mode select bits of R0 allows you to select between All magazine and single magazine modes. The default is all magazine (06h). In this mode all data including packet31 data and teletext rows from all magazines are passed through to the buffer. The values for the two modes are:

06h	=	All magazine mode
04h	=	Single Magazine Mode

If you are programming the card to use single magazine mode you must also set R3 and R1 correctly or no data will be stored.

Bits 3 -7 unused (must be 0)

R1 - Magazine Select (Single Magazine Mode) (Def 4?h)

Bits 0-3 Magazine select

The magazine select register will only have an effect if you are using the single magazine mode. It is used to select the magazine that you wish to capture data in (0-7). For example:
magazine 1=page 100-1FF
magazine 2=page 200-2FF
magazine 3=page 300-3FF
magazine 0 is possible and usually has closed caption information.

Bits 4-7 Reserved (must be set to 0100b or 4h)

R2 - unused (must be set to 07h/7dec)

R3 - Teletext/Packet31 width (Def 2Ah/42dec)

Bits 0-5 Teletext/Packet31 width

This register sets the width of the packets to be stored in the FIFO. In all magazine mode, the default is 2Ah/42dec signalling 42 bytes wide. This is because Unitext will deliver 40 bytes of teletext plus the magazine and row address group (2 bytes) ahead of each row. In all single, magazine mode the value should be set to 28h/40h. Although this would suggest that 40 bytes will be delivered for each row, actually 41 bytes are delivered for each. These are the row number + the teletext data.

Bits 6-7 unused (Should be zero)

R4 - Framing Code Value or Line Select Port 2

Bits 0-7 Framing code value (If Bit 6 of reset R6 is clear) (Def 27h/39dec)

This register sets the framing code which will be saught by the card immediately after the clock run-in. The default value is 27h (00100111b). If the framing code being telecast is different to the code shown then change this value to suit. This code is telecast LSBit first so it may appear in specifications as a bit stream and is shown backwards (11100100b).

Bits 0-7 VBI line select (if Bit 6 of R6 is set) Lines 6-13 (319-326 interlaced)

This register sets the VBI line selections for lines 6-13. When you write a value to this port, the lines corresponding to each bit will be enabled or disabled according to the bit state. Line 6=bit 0 , line 7=bit 1 etc. If a VBI Line is disabled, data on the corresponding line will be ignored and not submitted to the FIFO.

R5 - Sync Delay,Pos/Neg Triggering or Line Select Port 3

Bits 0 - 6 Synchronisation delay (when Bit 6 of reset R6 is clear) (Def D2h/210dec)

The Synchronisation delay governs the maximum time for the card to wait until run in is found. It is measured in us and can be adjusted by calculating the register value **'VALUE'** using the formula below. Given the required delay **'DELAY'** value in us.

PAL applications:

VALUE=(DELAY*6.9375)+1

*NTSC applications will require a different value.

The default delay is just under 12us (Bit 0-6 or of R5=52h/82dec)

Bit 7 Positive/Negative Edge Triggering (when Bit 6 of reset R6 is clear)

The Msbit of R5 toggles Positive or negative edge triggering. (1=Positive, 0=Negative). The recommended setting is positive (set).

Bits 0-7 VBI line select (if Bit 6 of R6 is set) Lines 14-21 (327-334 interlaced)
This register sets the VBI line selections for lines 14-21. When you write a value to this port, the lines corresponding to each bit will be enabled or disabled according to the bit state. Line 14=bit 0 , line 15=bit 1 etc. If a VBI Line is disabled, data on the corresponding line will be ignored and not submitted to the FIFO.

R6 - VBI/Reset/IRQ enable (Def 80h/128dec)

Any Write Reset CIDAC
This register is slightly different to all other registers on the card. Any value written to this register will automatically reset the CIDAC, clear the FIFO and reset the framing system ready for new rows/packets.
Bits 0 - 3 unused

Bit 4 I2C Data Line
***NOTE: I2C Bus signals are inverted on this bit (setting this bit pulls the I2C line low).**
This bit is mapped to the I2C tuner bus data line and is used to transmit data to the tuner via the I2C bus. This bit must be used in conjunction with the **I2C Clock Line** bit when communicating with the tuner. Setting this bit to 1 will pull the data line low

Bit 5 I2C Clock Line
***NOTE: I2C Bus signals are inverted on this bit (setting this bit pulls the I2C line low).**
This bit is mapped to the I2C tuner bus clock line and is used to synchronise data transmission between the software and the tuner hardware. The programmer must drive this bit low (set this bit) prior to writing the bit value to the **I2C Data Line** and then bring it high again (clear this bit) to inform the hardware to take the **I2C Data Line** state and submit it to the bus.

Bit 6 Select R4 & R5 functions
This bit allows selection of either the VBI line select ports and I2C lines or the Framing code and Sync Delay &+-triggering are to be addressed via ports R4 and R5. When the bit is clear output to R4 will be mapped to the internal framing code register, output to R5 will be mapped to the Sync delay and triggering edge select internal registers. If this bit is set, output to R4 will be mapped to the VBI select register for lines 6-13, the output to R5 will be mapped to the VBI select register for lines 14-21.

Bit 7 IRQ enable
When set this bit will cause the card to generate a retrace interrupt every time the signal exits Vertical Blanking. The interrupt flag will remain set and no further interrupts will be generated until a status read (Read R0) is detected.

R7 - Colour Burst Blanking (Def 3Bh/59dec)

Bits 0 - 5 Colour burst blanking width (when Bit 6 of reset R6 is clear).
The colour burst blanking width can be adjusted to blank the colour burst out prior to starting clock run-in. If you adjust this value too far downward you run the risk of the colour burst being interpreted as being part of the run-in causing sync errors. If you adjust it upward too far you can run the risk of blanking too much of the clock run-in, again causing sync errors.

The recommended default for this value is 8.36us (R7=3Bh) for PAL. The value can be adjusted to a delay value using the following formula and inserting the required delay '**DELAY**' into it to solve for the register value **'VALUE'**.

PAL applications:

VALUE=(DELAY*6.9375)+1

*NTSC applications will require a different value.

Read Registers

R0 - Read FIFO & IRQ Status

Bit 0 FIFO overflow if set

Bit 1 signals that a FIFO overflow has occurred. This happens when more than 2047 bytes have been received since your software last serviced the card. The data may no longer be framed correctly and a reset (write to register 6) should be performed.

Bit 1 Data available in read register if unset

This bit is cleared once data is ready for reading via the data register (R1).

Bit 2 Memory empty if set

This bit is set if the memory is empty. Caution because data may still entering FIFO and is not ready for reading. Use bit 1 to check if data is available at the read register rather than this bit.

Bit 3-6 unused

Bit 7 IRQ was generated by the card if set

If the MSBit of R0 was set then this signifies that the IRQ (if turned on in R6) was generated by the Unitext card. If you are operating using IRQs, read the status register and if this bit is clear then chain the original ISR for the IRQ. If the bit is set then this actual read would reset the internal interrupt flag telling Unitext to resume generation of VBI-end interrupts. If the bit is found on you must read your data from the FIFO immediately before it is overwritten by incoming data.

R1 - Read 1 Byte Of FIFO Contents

This register is the FIFO data interface register. Once data is found when reading the status, at least 1 byte is ready to be read. Your routine should read a byte then re-check the status and if data is still available then read the next byte. Data will always be framed (except after an overflow) so that the first byte of teletext in the row (or MRAG if all magazines) will be read from this register with the first read. If mid-row you find that a reasonable time has expired without the rest of the row becoming available, the CIDAC should be reset as a reception glitch has occurred causing a time-out.

R6 - Read FIFO & IRQ Status

Bit 4 I2C Data Line (when Bit 6 of reset R6 is clear)

This bit is mapped to the I2C tuner bus data line and is used to read data from the tuner via the I2C bus. This bit must be used in conjunction with the **I2C Clock Line bit** (R6 bit 5 Write) when communicating with the tuner. If this bit is set then the data line is low.

Discussion

The data should be in rows or packets of anything from 0 - 16 rows in each VBI. Your routine should be capable of handling variable numbers of rows each VBI as data is not always transmitted on each line.

Remember that if IRQ generation was being used that you must acknowledge the interrupt in the 8259 programmable interrupt controller and if a high IRQ was used, the slaved 8259 must be answered. The card does not need acknowledgment as reading the status register (R0) is seen as acknowledgment and the internal interrupt flag is cleared.

If using Microsoft Windows then a timer set to 55ms intervals will be adequate if using a timer driven polling method in single magazine capture mode.

A packet driver should process the data the same way as discussed above and if an incomplete packet is found (< 42 bytes before timeout) then reset the CIDAC and dump the errored packet.

Pseudo Code for data collection IRQ - ISR:

```
Start
Read Status Register
If Interrupt Bit Set Then
    Acknowledge 8259 PIC
    If Overflow Bit Set Then
        Reset CIDAC
    Else
        Set Bytecount=0
        Timeout_Count=0
        Do While Data Ready Bit Set Or Timeout_Count<100
            If  Data Ready Bit Set Then
                Read And Store Byte In Row/Packet Buffer
                Increment Bytecount
                If  Bytecount=42 Then  {Note Bytecount=41 if single mag mode}
                    Store Row In Page Buffer        { Store Packet}
                    Bytecount=0
                Endif
                Timeout_Count=0
            Else
                Increment Timeout_Count
            End if
            Read Status
        While-End
        If Timeout_Count>=100 Then
            Reset CIDAC
        Endif
    Endif
Else
    Call Original ISR
Endif
Return From ISR
```

Unitext V2.0 Rev B - Port Level Programmer Specification - DRAFT ONLY

FI 12xx Tuner Addendum

The Unitext Version 2.0 system uses the Philips FI1200 series Multimedia tuner. This tuner uses an I2C bus to allow software to serially read and write commands and data to the tuner. For reasons of forward compatibility and ease of maintenance the I2C bus clock and data lines have been directly mapped to 2 bits of the Unitext's IO ports. The tuner requires that '*Telegrams*' of commands be sent to specify the channel frequency. A *telegram* can also be read to report back to software the state of the tuners ADC register to determine whether the tuner is on station and/or in band.

To program the tuner correctly you will need to obtain a FI1200 series tuner specification from your Philips distributor. The following source code may be sufficient to perform basic tuning functions however.

The I2C Bus is mapped for reading and writing to register R6 at bits 4 (data) and 5 (clock). Writing a 1 to the corresponding bit will drive the I2C line **LOW**. Combining the toggling of the clock and data lines in accordance to the I2C Bus spec allows the serial transmission of the data telegrams used to set the tuner's channel frequency.

Beware: Any write to register R6 will also signal the Unitext card to reset its CIDAC therefore losing any data in it's FIFO, therefore adjustment of I2C tuner using R6 will interrupt data flow momentarily.

Sample programming code in IBM 8086 assembler to set a tuner frequency.

***WARNING: This document is private and subject to non-disclosure.**
This example uses the cmacros.inc include found in the Windows SDK to easily show parameter passing to exported routines.

These routines are copyright 1996/7 Pelican Electronic Developments and may be included into support software for products of Pelican Electronic Developments **ONLY**. You may not duplicate or distribute this document nor may you use the intellectual knowledge contained within for any other purpose.

MACROS USED:

```
The Macros shown here assume dx=read or write port and
al=current state of read or write register.

@RESETI2C    MACRO
        mov      al,00h         ;reset i2c bus
        out      dx,al
        ENDM
@CLKLOW      MACRO
        or       al,00100000b ;pull clock line low
        out      dx,al
        ENDM
@CLKHI       MACRO
        and       al,11010000b;send clock line high
        out       dx,al
        ENDM
@DATLOW      MACRO
        or       al,00010000b ;pull data line low
        out      dx,al
        ENDM
@DATHI       MACRO
        and      al,11100000b ;send data line hi
        out      dx,al
        ENDM
@STOP        MACRO
        mov      al,0           ;send stop
        out      dx,al
        ENDM
```

Unitext V2.0 Rev B - Port Level Programmer Specification - DRAFT ONLY

DATA AND VALUES USED:

```
VHF_THRS      EQU       172
UHF_THRS      EQU       451
COLOURBURST   EQU       3Bh       ;Default R7 colourburst setting
I2CWRITEREG   EQU       6
I2CREADREG    EQU       6
I2CACKNOWBIT  EQU       00010000b
I2CLOCKONLY   EQU       00010000b
I2CDATAONLY   EQU       00100000b
```

freq divide — control = slow tune, test = off, Ratio select = 31.25 KHZ

BAND = high band.

```
Wtelegram  label  byte
db      0C2h,00h,00h,8Eh,30h,00h,00h    ;pre set to address C2
Rtelegram  label  byte
db      0C3h,00h,00h,00h,00h,00h,00h    ;pre set to address C3
```

address

```
;******************************************************************
;   FUNCTION:     TuneCard()    -   Exported
;
;   PURPOSE: Tune the FI12XX tuner on the card to the values
;            passed
;
;   INPUT  : Mhertz = Mhz part of frequency
;                     (ie 182.25Mhz, Mhertz = 182)
;            Khertz = Khz part of frequency
;                     (ie 182.25Mhz, Khertz = 250)
;   OUTPUT : Returns the result (ah==1=bad, ah==0=good)
;                   (if good al=tune quality from ADC)
;                   (if ah=1 the tuner failed to acknowledge)
;
;******************************************************************
cProc         TuneCard, <FAR,PUBLIC>, <es,di>
parmW Mhertz;
parmW KHertz;
cBegin        TuneCard

        cCall    calcfreq, <Mhertz,Khertz>
        call     sendfreq     ;send multiple times as the
        call     sendfreq     ;tuner will take >200ms
        call     sendfreq     ;to tune to a frequency
        call     sendfreq
        call     sendfreq
        call     sendfreq
        mov      ah,1         ;assume no tuner ah=1=bad
        cmp      al,0         ;if al=0 then no tuner ack
        je       badsend
           call      gettelegram  ;get the tuner status in al
           mov       ah,0     ; ah=0=tuner acknowledged
badsend:
cEnd    TuneCard
```

```
;*****************************************************************
;   FUNCTION:     SetVBILines()     -  Exported
;
;   PURPOSE: Set Which VBI Lines The System Will Capture Data
;            From.
;
;   INPUT   : lineflag - long value with a bit per line
;
;*****************************************************************
cProc         SetVBILines, <FAR,PUBLIC>, <es,di>
parmW lineflag;
cBegin   SetVBILines

         mov      bx,lineflag
         mov      dx,wIOPort    ;ie 318h
         add      dx,6          ;ie 31Eh
         mov      al,40h        ;01000000b =VBI set bit
         out      dx,al
         jmp      $+2           ;let bus settle
         sub      dx,2          ;ie 31Ch
         mov      al,bl         ;line mask from caller
                                ;         (hi 8 bits)
         out      dx,al
         jmp      $+2           ;let bus settle
         inc      dx            ;ie 31Dh
         mov      al,bh         ;line mask from caller
                                ;         (lo 8 bits)

         out      dx,al
         jmp      $+2           ;let bus settle
         inc      dx            ;ie 31Eh

         mov      al,R6         ;reset
         out      dx,al
         jmp      $+2           ;let bus settle

cEnd     SetVBILines
```

```
;*****************************************************************
;   FUNCTION:     calcfreq()    - Not Exported
;
;   PURPOSE: Calculate the frequency values for the FI12XX ;
             tuner
;        The calc is (Mhz+38.9)*16.
;   INPUT  :
;        Mhtz = Mhz part of frequency
;               (ie 182.25Mhz, Mhtz = 182)
;        Khtz = Khz part of frequency
;               (ie 182.25Mhz, Khtz = 250)
;
;   OUTPUT :
;        Writes tuner telegraph string to blank spaces in
;        Wtelegram
;
;*****************************************************************
cProc calcfreq, <FAR,PUBLIC>, <es,di>
Parmw Mhtz;
Parmw Khtz;
cBegin          calcfreq

        push    ax
        push    cx
        push    di
        push    bx

        mov     bx,Mhtz
        mov     cx,Khtz

        and     bx,07FFh ;mask off invalid bits
        and     cx,03FFh ;mask off invalid bits

        mov     dl,0A0h  ;assume low band
        cmp     bx,VHF_THRS
        jl      gotband

        mov     dl,090h  ;assume mid band
        cmp     bx,UHF_THRS
        jl      gotband

        mov     dl,030h  ;must be high band
gotband:
        push  dx
        add      bx,38     ; add 38.9 Mhz to value
        add      cx,900

        shl      bx,1 ;*16
        shl      bx,1
        shl      bx,1
        shl      bx,1

        shl      cx,1 ;*16
        shl      cx,1
        shl      cx,1
        shl      cx,1

        mov      ax,cx
        mov      cx,1000
        xor      dx,dx
        div      cx         ;/1000 to get Khz part of value
        add      ax,bx      ;tuner value in ax=Mhz+(Khz/1000)
        pop      dx

        mov      di,offset Wtelegram
        inc      di
        mov      ds:[di],ah
        inc      di
        mov      ds:[di],al
        add      di,2
        mov      ds:[di],dl          ;write band in di

        pop      bx
        pop      di
        pop      cx
        pop      ax

cEnd    calcfreq
```

```
;*******************************************************************
;   FUNCTION:     longpause() - Not Exported
;
;   PURPOSE: Act as a long temporary bus pause to slow
;            writing to the tuner
;
;   INPUT  : None
;   OUTPUT : None
;*******************************************************************
longpause    proc     near

        pushf
        push     cx
        mov      cx,8000h
loopmain:
        loop     loopmain     ;wait for bus to settle
        pop      cx
        popf
        ret

longpause    endp

;*******************************************************************
;   FUNCTION:     shortpause() - Not Exported
;
;   PURPOSE: Act as a short temporary bus pause to slow
;            writing to the tuner
;
;   INPUT  : None
;   OUTPUT : None
;*******************************************************************
shortpause   proc     near

        pushf
        push     cx
        mov      cx,400h
loop1:
        loop     loop1
        pop      cx
        popf
        ret

shortpause   endp
```

```
;*****************************************************************
;   FUNCTION:      sendfreq() - Not Exported
;
;   PURPOSE: Sends the contents of Wtelegram to the FI12xx
;            tuner on the card so that it tunes to the value
;            set by calling calcfreq
;
;   ASSUMES: DS=Data Segment
;
;   INPUT   : Wtelegram
;
;   OUTPUT  : AL=0=NoTuner,AL=1=GoodSend
;
;*****************************************************************
sendfreq proc      near

         push      si

         mov       dx,wIOPort        ;ie 318h

         add       dx,I2CWRITEREG    ;ie 31Eh

         @RESETI2C

         call      longpause

         mov       al,40h            ;select
         out       dx,al

         mov       si,offset Wtelegram ;addr of str is ds:[si]

         mov       cx,5              ;5 bytes in the string

         call      sendi2cstr        ;send telegram to tuner

         pop       si
         ret

sendfreq endp
```

```
;*************************************************************
;   FUNCTION:     gettelegram() - Not Exported
;
;   PURPOSE:      Gets the tuner's registers into Rtelegram to
;                 check fine tuning etc
;
;   INPUT  :      From Port
;
;   OUTPUT :      RTelegram
;
;*************************************************************
gettelegram  proc    near

        push     si

        mov      dx,wIOPort        ;ie 318h
        add      dx,I2CWRITEREG    ;ie 31Eh

        @RESETI2C

        call     longpause

        mov      al,40h       ;select
        out      dx,al

        mov      si,offset Rtelegram ;addr of str ds:[si]

        mov      cx,5         ;5 bytes max in string

        call     ReadI2Cstr   ;read telegram from tuner

        mov      si,offset Rtelegram ;addr of str ds:[si]
        add      si,1
        mov      al,ds:[si]
        xor      ah,ah

        pop      si

        ret

gettelegram  endp
```

```
;******************************************************************
;   FUNCTION:       sendi2cstr() - Not Exported
;
;   PURPOSE:        Sends a telegram string to the tuner
;
;   INPUT   :       cx = telegram length
;                   dx = I2C port address (set in sendfreq)
;                   ds = segment of telegram
;                   si = offset of telegram
;
;   OUTPUT  :       al = 1 if sent ok
;                   al = 0 if it fails
;
;******************************************************************
sendi2cstr   proc    near
        ;send start sequence
        xor     al,al
        @DATLOW

        push    bx
        ;loop through sending bytes
Wbyteloop:
        xchg    al,ah           ;swap state to ah
        lodsb                   ;load next byte
        xchg    al,ah           ;swap state & byte back
        call    sendi2cbyte     ;send byte in al to I2C bus
        mov     bl,ah
        and     bl,I2CACKNOWBIT
        jnz     badack          ;was it acknowledged ok ???
        loop    Wbyteloop

        pop     bx

        ;acknowledge & stop sequence
        call    shortpause

        @CLKLOW

        call    shortpause

        @DATLOW

        call    shortpause

        @CLKHI
        call     shortpause
        call     shortpause  ;need one extra wait

        @CLKLOW

        call     shortpause
        call     shortpause

        @CLKHI

        call     shortpause

        @STOP

        mov      al,1         ;set al=1 means tuned ok
        ret
badack:
        pop      bx
        mov      al,0         ;set al=0 means tuner error

        ret


sendi2cstr   endp
```

```
;*****************************************************************
;   FUNCTION:     sendi2cbyte() - Not Exported
;
;   PURPOSE:      ends a byte to the FI12xx tuner
;
;   INPUT   :     dx = I2C port address (set in sendfreq)
;
;   OUTPUT  :     al = port state
;                 ah is destroyed
;
;*****************************************************************
sendi2cbyte  proc    near

        push     cx
        mov      cx,8
bitloop:
        call     shortpause   ;wait a bit

        @CLKLOW      0x20 set

        call     shortpause   ;wait a bit more

                                    0x10
        or       al,I2CACKNOWBIT    ;speculate that its a zero
        shl      ah,1         ;get first bit of address
        jnc      high0        ;was bit 0 high, jump if it was
        and      al,I2CDATAONLY 0x20 ;drive data high
        or       al,COLOURBURST 0x30
                                      WHY?
high0:
        out      dx,al        ;send bit

        call     shortpause
        @CLKHI       0x20 clear.
        call     shortpause
        loop     bitloop

        call     getack       ;see if acknowledged

        pop      cx
        ret

sendi2cbyte      endp
```

```
;*****************************************************************
;   FUNCTION:      readi2cstr() - Not Exported
;
;   PURPOSE:       Reads a telegram from the FI12xx tuner
;
;   INPUT   :      cx = max telegram len
;                  dx = I2C read port (set in gettelegram)
;                  ds = telegram segment
;                  si = telegram offset
;
;   OUTPUT  :      None
;
;*****************************************************************
readi2cstr          proc     near

        ;send start sequence
        mov         al,COLOURBURST
        or          al,I2CLOCKONLY ;data low = start condition
        out         dx,al

        xchg        al,ah          ;swap state to ah
        lodsb                      ;load next byte
        xchg        al,ah          ;swap state & byte back
        call        SendI2Cbyte    ;send byte in al to I2C bus

        push        bx
        ;loop through getting bytes
Rbyteloop:
        call        readi2cbyte    ;get byte in al from I2C bus
        mov         ds:[si],al
        inc         si
        mov         bl,ah
        and         bl,I2CACKNOWBIT
        jnz         endtelegram
        loop        Rbyteloop

endtelegram:
        pop         bx
        ret

readi2cstr          endp
```

```
;*******************************************************************
;   FUNCTION:     readi2cbyte() - Not Exported
;
;   PURPOSE: Reads a byte from the FI12xx tuner
;
;   INPUT  : dx = I2C read port
;
;   OUTPUT : al = port state
;        ah = byte received from the FI12xx
;
;*******************************************************************
readi2cbyte  proc    near

        push     cx
        mov      cx,8
        mov      bl,0
Rbitloop:
        shl      bl,1           ;1st time wont affect anything
                                ;after that will roll the byte

        call     shortpause     ;wait a bit

        @CLKLOW

        call     shortpause     ;wait a bit more

        xchg     al,ah

        sub      dx,I2CREADREG
        in       al,dx          ;read bit    | WHY.
        add      dx,I2CREADREG

        xchg     al,ah
        ;ah has data bits

        or       bl,1           ;speculate that the bit was set
        test     ah,I2CACKNOWBIT  ;00010000b
        jnz      bitwasset
        and      bl,11111110b ;wasn't set so unset the bit

bitwasset:
        call     shortpause

        @CLKHI
        call      shortpause
        loop      Rbitloop

        call      getack                  ;see if acknowledged
                                  ;insert code here if not acked

        mov       al,bl                   ;return byte in al
        pop       cx
        ret

readi2cbyte  endp
```

```
;****************************************************************
;    FUNCTION:      getack() - Not Exported
;
;    PURPOSE:       gets an acknowledge signal from the tuner
;
;    INPUT   :      dx = I2C Port
;
;    OUTPUT  :      ah = acknowledgement
;
;****************************************************************
getack   proc     near

         call     shortpause

         @CLKLOW

         call     shortpause

         @DATHI

         call     shortpause

         sub      dx,I2CREADREG      ;ie 318h
         in       al,dx          ;bit 4 should be low if ack
         mov      ah,al          ;store in ah for calling
routine
         add      dx,I2CREADREG      ;ie 318h

         @CLKHI

         call     shortpause

         ret

getack   endp
```

WHY? should be same as write.