

[illegible]

```

''currently only operative for servo PWM synthesis. Support for high speed UART
''communication will be added later once serial ring buffers are supported by Bascom.
''!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
''!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
''!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
' =====> SET UP THE BOARD AND CRYSTAL HERE <=====
' =====> THIS IS THE ONLY THING YOU NEED TO DO <=====
'Const Ver_board = 0                                'Crumb 2560
'$regfile = "m2560def.dat"

'Const Ver_board = 1                                'AR7212_A Ralf Kull (2560)
'$regfile = "m2560def.dat"

Const Ver_board = 2                                'AR7212_B Ralf Kull (1280)
$regfile = "m1280def.dat"

'Const Ver_board = 3                                'Xmega-A1-USB
'$regfile = "kixm128A1def.dat"

'Const Ver_board = 4                                'AR7212_C Ralf Kull (1280)
'$regfile = "m1280def.dat"

'Const Ver_cryst = 0                                '14.745600 MHz
Const Ver_cryst = 1                                '11.059200 MHz
'Const Ver_cryst = 2                                '16.000000 MHz
'Const Ver_cryst = 3                                '32.000000 MHz

'Const Ver_uart0 = 0                                'USB and Telit GM862 as 2nd
link
Const Ver_uart0 = 1                                'USB and HAC UM96 as 2nd link

'Const Ver_press = 0                                'MPX4115
Const Ver_press = 1                                'SPC1000-P01 (SPI)

'Const Batadc = 0                                    'not used right now
'=====
'=====
'!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
'!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
'!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
'-----[Software Extensions in preparation]-----
'Extension 1: GPS, pressure sensor (altitude) and 2 Co-Pilot sensor modules
'from FMA direct for flight stabilization and radio controlled waypoint setting.
'"Stay where you are"/airparking option. "Return home" option. "Learn to fly"
'option. PC only required at home.
'Extension 2: 2 x Hac UM96 modules or other 2nd links and PC interface.
'All options of extension kit 1 plus PC control with telemetry, course re/setting
'and GPS tracking on anything that is covered by Google Maps. 500 m range limitation.
'Autothermalizing and autonomous lift finding option. PC required at field.
'Extension 3: 2 x Telit GSM/GPRS modules and PC interface. Range limitation
'only by cell phone coverage. OBEYING THE VISUAL FLIGHT RULES IS A MUST.
'Note that the above extensions were tested on a MPX Easystar using a pre-AR7212
'board earlier (actually already in 2006-2007), but need more testing here and
'thus will be published AFTER the test-phase IN THE AIR.Note that the definition
'of constants and global variables als well as the declaration of subs and
'functions are already included here.
',
'-----[Mode of operation]-----
'The basic idea is about an intelligent RX that can be programmed via USB from
'a PC. At the PC side there is the AR7212 programmer, viz. a little program
'allowing to read, edit and write 2 tables that are stored in the EEPROM of
'the board and independantly on PC disk. All tables have been encoded as 1d
'word or integer arrays, due to the lack of 2d arrays in Bascom.
',
'First comes the servo data table (Table 1)
'The rows are:

```

```

''(#01) for servo 1
''(#02) for servo 2
''(#03) ..
''(#04) ..
''(#05) ..
''(#06) ..
''(#07) ..
''(#08) ..
''(#09) ..
''(#10)..
''(#11)..
''(#12)..

''The columns are:
''(a) servo channel name (1-12), eg. MOTO, RUDD, ELEV, AILR, AILL, ...
''(b) servo center. Pressing on < or > will affect the SELECTED servo accordingly.
''(c) servo normal high: This sets the upper value of the normal travel range.
''(d) servo normal low: The opposite position.
''(e) servo maximal high: Maximal travel limiter for one direction.
''(f) servo maximal low: The opposite direction.
,,

''Next comes the mixer data table (Table 2).
''The rows are:
''(#01) for servo 1
''(#02) for servo 2
''(#03) ..
''(#04) ..
''(#05) ..
''(#06) ..
''(#07) ..
''(#08) ..
''(#09) ..
''(#10)..
''(#11)..
''(#12)..
,,

''The columns are as found on the DX7 plus offset and delay.
''(a) THRO
''(b) AILE
''(c) ELEV
''(d) RUDD
''(e) GEAR
''(f) FLAP
''(g) offset
''(h) delay
,,

''Under these settings three DX7-switches act as controls: GEAR, FLAP, and AUX2
''AUX2 is kept as a general control channel - allowing to switch between manual
''and autonomous flight. From GEAR (2 position switch) and FLAP (3 position switch)
''six flight modes can be encoded. Table 2 thus needs to be defined for all 6 fltmodes.
''The AR7212 programmer supports an easy way to copy and paste or edit the tables.
''Flightmodes may be clones, slightly changed copies of other flightmodes, or completely
''independent. Examples:
''(1) Flight Mode 1: e.g. Start (GEAR 0, FLAP N): Throttle stick controls motor
''(2) Flight Mode 2: e.g. Normal flight (GEAR 0, FLAP 1): Throttle controls motor
''(3) Flight Mode 3: e.g. Camera pan/tilt (GEAR 0, FLAP 2): Throttle and rudder stick
''    control camera for FPV while elevator and aileron the airplane
''(4) Flight Mode 4: e.g. Landing (GEAR 1, FLAP N): Motor off,
''    Throttle controls Butterfly plus airbrakes
''(5) Flight Mode 5: e.g. Landing (GEAR 1, FLAP 1): Motor off,
''    Throttle controls just airbrakes
''(6) Flight Mode 6: e.g. Thermal/speed flight: (GEAR 1, FLAP 2) Motor off,
''    Throttle controls camber
''One is completely flexible here. A beginner will appreciate to have just one
''flight mode viz. fltmode 1. He will program the other flight modes as clones of
''flight mode 1. An experienced pilot may even control two airplanes simultaneously,
''plane 1 with 3 fltmodes and servo channels 1-6, plane 2 with 3 fltmodes and servo

```

```

''channels 7-12 using a second AR7212. Let's call this air juggling. Never do this
''with hotliners!!! Only for Zagi's, Easystars, etc.
''
''Computing the servo signals for each slave channel j is a straightforward task:
''
''(a) determine the flight mode and read the offsets.
''(b) add to the offsets the sum of products of master(i) * mixer(i,j)
''(c) If delay (Table 1) is zero remember the value for channel j. If not compute
''    the difference between to be and is-value, divide it by the number of
''    increments, add the servo increment, and repeat addition in next cycles
''    until to be is established.
''(d) Take the result for channel j and correct if outside range (servo maxlimiters in
''    table 1). Update the 12 servos directly after the last operation is finished.
''
''Breaking it down to Bascom with 1d table limitation: Define Master(i) array
''(=DX7) and Slave(j) (=Servo) array as integer. Mixdat(k) array has the Dimension
''of 12x6=72 for each fltmode. The resolution of Master channels from DX7 is a
''little less than 10bit setting an integer range of -384 to +384. Servo PWM
''resolution depends on the crystal frequency and is 2000/ms at 16 MHz (not
''recommended), while 1842/ms and 1382/ms at 14.75 and 11.06 MHz clock speed,
''respectively. The latter crystals are baud rate crystals recommended to achieve
''0% baud rate errors at 115200 bps needed for the Spektrum satellites.
''
''
''-----[Mixer types]-----
''The AR7212 currently distinguishes the following mixer types stored in byte
''arrays MixTyp() and MixTypm1()..MixTypm6(). Values are:
''
'' 0 for normal linear mixers (positive and negative slope)
'' 1 for J-mixers
'' 2 for F-mixers
'' 3 for L-mixers
'' 4 for T-mixers
''
'' JFLT-mixers are bimodal linear mixers whose outputs are neutral before or after
'' the respective stick-center position, while proportional to the stick move in
'' the other direction.
''
'' The boxes below represent corresponding input-output relations:
''
''
''      +---+      +---+      +---+      +---+      +---+      +---+
''      | / |      | \ |      | / |      | | |      | \ |      | | |
'' out | 0 | out | 0 | out |-0 | out | 0- | out | 0- | out |-0 |
''      | / |      | \ |      | | |      | / |      | | |      | \ |
''      +---+      +---+      +---+      +---+      +---+      +---+
''      in         in         in         in         in         in
''
''      -----normal-----   ---J---   ---F---   ---L---   ---T---
''
'' 0 represents the center of the coordinate system.
'' / represents a positive slope, viz. mixer values > 0
'' \ represents a negative slope, viz. mixer values < 0
'' - represents a zero slope      , viz. mixer values = 0
''
'' Mixer (slope) values are stored in byte arrays MixDat() and MixDatm1()..MixDatm6().
'' Offset (intercept) values are stored in byte arrays offset() and offsetm1()
'' ..offsetm6().
'' The slope-direction of normal mixers is determined just by the sign,
'' while in the case of JFLT mixers slope direction is determined by characters,
'' J and F for positive slope, L and T for negative.
'' Exponential response (for the above stick inputs) stays programmable on the DX7.
'' The same holds for dual-rate switches. Any other "mixing" is to be shifted into
'' the AR7212.
''
''
''-----[The Spektrum satellite receiver serial frame format]-----
''Thanks to Rainer Walther who found out the following:

```

```

''
''Connection Spektrum Receiver:
''  Orange: Vcc = 3V (Ki: 3.3V is also OK)
''  Black: GND
''  Grey: TX signal from receiver
''
''DX7/DX6i: One data-frame at 115200 baud every 22ms.
''DX7se: One data-frame at 115200 baud every 11ms.
''  byte1: unknown
''  byte2: unknown
''  byte3: and byte4: channel data (FLT-Mode) = FLAP
''  byte5: and byte6: channel data (Roll) = AILE
''  byte7: and byte8: channel data (Nick) = ELEV
''  byte9: and byte10: channel data (Gier) = RUDD
''  byte11: and byte12: channel data (Gear Switch) GEAR
''  byte13: and byte14: channel data (Gas) = THRO
''  byte15: and byte16: channel data (AUX2) = AUX2
''
''DS9 (9 Channel): One data-frame at 115200 baud every 11ms,
''alternating frame 1/2 for CH1-7 / CH8-9
'' 1st Frame:
''  byte1: unknown
''  byte2: unknown
''  byte3: and byte4: channel data
''  byte5: and byte6: channel data
''  byte7: and byte8: channel data
''  byte9: and byte10: channel data
''  byte11: and byte12: channel data
''  byte13: and byte14: channel data
''  byte15: and byte16: channel data
'' 2nd Frame:
''  byte1: unknown
''  byte2: unknown
''  byte3: and byte4: channel data
''  byte5: and byte6: channel data
''  byte7: and byte8: 0xffff
''  byte9: and byte10: 0xffff
''  byte11: and byte12: 0xffff
''  byte13: and byte14: 0xffff
''  byte15: and byte16: 0xffff
''
''Each channel data (16 bit= 2byte, first msb, second lsb) is arranged as:
''
''Bits: F 0 C3 C2 C1 C0 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0
''
''0 means a '0' bit
''F: 1 = indicates beginning of 2nd frame for CH8-9 (DS9 only)
''C3 to C0 is the channel number. 0 to 9 (4 bit, as assigned in the transmitter)
''D9 to D0 is the channel data (10 bit) 0xaa..0x200..0x356 for 100% transmitter-travel
''
''Unknown bytes interpretation (Kiedro).....
''  Monitoring the unknown bytes as a function of distance and RX shielding
''  (one of the two satellites in the test was put into a metal box for this test)
''  revealed the following outcomes: 0x0301, 0x0201, 0x0101, 0x1201, 0x2101.
''  Optimal conditions gave 0x0301, worst conditions 0x2101. My interpretation
''  is as follows:
''
''  byte1: [0 0 W1 W0 0 0 S1 S0] is a status byte
''  byte2: [0 0 0 0 0 0 0 1] is a sync byte
''
''W1: Warning bit set on repetitive frame losses.
''W0: Warning bit set on newly occurring frame loss..
''S1 to S0 define signal quality in three steps.
''
''Note that the AR7212 takes a conservative approach towards this interpretation:
''From the two satellites it takes the signal with higher or equal quality and of
''lower or equal warning status.

```

```

''
''
''
''-----[ Special port assignments for AR7212-boards A & B by Ralf Kull ]-----
''-----[ Board B is described in the Application Note ]-----
''-----

'Port A: n.c.
'

'Port B: 7-4: Output compare for Timer 1 (PWM).
'      3-0: MISO MOSI SCK SS (useable for pressure sensor SCP-1000 later)
'          7: Output Compare OC1C (used for servo 5)
'          6: Output Compare OC1B (used for servo 4)
'          5: Output Compare OC1A (used for servo 3)
'          4: SD_SW (SD-Card)
'          3: MISO
'          2: MOSI
'          1: SCK
'          0: CS (SCP-1000)
'

'Port C: LED and configuration port.
'      7: CS-SD (connect to SW1, also used)
'      6: SW1 (Switch to GND)
'      5: LED2 (output 1 to activate)
'      4: LED3 (output 1 to activate)
'      3: LED4 (output 1 to activate)
'      2: n.c.
'      1: JMP1 (pin jumper to GND)
'      0: JMP2 (solder bridge to GND)
'

'Port D: Configuration, 2nd hardware UART (UART1), two wire serial interface
'      7: JMP4 (solder bridge to GND)
'      6: JMP3 (solder bridge to GND)
'      5: n.c.
'      4: n.c.
'      3: TXD UART1
'      2: RXD UART1 (used for GPS)
'      1: TWI serial data (SDA for compass module)
'      0: TWI serial clock (SCL for compass module)
'

'Port E: Output compare for Timer3 (PWM), 1st hardware UART (UART0)
'      7: n.c.
'      6: n.c.
'      5: Output Compare OC3C (used for servo 6)
'      4: Output Compare OC3B (used for servo 2)
'      3: Output Compare OC3A (used for servo 1)
'      2: n.c.
'      1: TXD UART0 (used for USB/GSM/radiomodem)
'      0: RXD UART0 (used for USB/GSM/radiomodem)
'

'Port F: Analog inputs
'      7-4: Formerly used for analog inputs from ADX or FMA Copilot.
'      3-0: Formerly for MPX fine, Vref, BatV, BatI.
'          7: ADC7 to JP2-10
'          6: ADC6 to JP2-09
'          5: ADC5 to JP2-08
'          4: ADC4 to JP2-07
'          3: ADC3 to JP2-06
'          2: ADC2 to JP2-05
'          1: ADC1 to JP2-04
'          0: ADC0 to JP2-03
'

'Port G: n.c.
'

'Port H: Output compare for Timer4 (PWM), 3rd hardware UART (UART2)
'      7: n.c.
'      6: n.c.

```

```

'      5: Output Compare OC4C (used for servo 9)
'      4: Output Compare OC4B (used for servo 8)
'      3: Output Compare OC4A (used for servo 7)
'      2: n.c.
'      1: TXD UART2
'      0: RXD UART2 (used for Spektrum Remote Receiver A)
'

```

```

'Port J: 4th hardware UART (UART3)
'      7: n.c.
'      6: connected to large-ISP header, pin 2 (SS)
'      5: n.c.
'      4: n.c.
'      3: n.c.
'      2: n.c.
'      1: TXD UART3.
'      0: RXD UART3 (used for Spektrum Remote Receiver B)
'

```

```

'Port K: Analog inputs ADC8-15.
'      7: n.c.
'      6: n.c.
'      5: n.c.
'      4: n.c.
'      3: ADC11 to OPC-ZA3
'      2: ADC10 to OPC_ZA2 (Copilot Z)
'      1: ADC09 to OPC_XY3 (Copilot Y)
'      0: ADC08 to OPC_XY2 (Copilot X)
'

```

```

'Port L: Output compare for Timer 5 (PWM)
'      5: Output Compare OC5C (used for servo 10)
'      4: Output Compare OC5B (used for servo 11)
'      3: Output Compare OC5A (used for servo 12)
'

```

```

'.....
'
' SERVO  01  02  03  04  05  06  07  08  09  10  11  12
'   OC   3A  3B  1A  1B  1C  3C  4A  4B  4C  5C  5B  5A
'  BEST  AR  AL  EL  RU  MO  nc  FR  FL  nc  BR  BL  nc
'-----

```

```

'-----[ Special port assignments for AR7212-board C by Ralf Kull ]-----
'

```

```

'Port A: n.c.
'

```

```

'Port B: 7-4: Output compare for Timer 1 (PWM).
'      3-0: MISO MOSI SCK SS (useable for pressure sensor SCP-1000)
'      7: Output Compare OC1C (used for servo 5)
'      6: Output Compare OC1B (used for servo 4)
'      5: Output Compare OC1A (used for servo 3)
'      4: SD_SW (to SW1/SW2 = switch to GND in boxed connector, superfluous?)
'      3: MISO (to SPI_3 and ISP_3)
'      2: MOSI (to SPI_4 and ISP_4)
'      1: SCK (to SPI_2 and ISP_2)
'      0: SS_CSPS ( to ISP_6)
'

```

```

'Port C: Configuration port & Main-LED
'      7: CS-SD (pin_2 boxed connector, superfluous?)
'      6: SW1 (Switch to GND)
'      5: JMP2 (solder bridge to GND, tbd)
'      4: JMP3 (solder bridge to GND, tbd)
'      3: JMP4 (solder bridge to GND, tbd)
'      2: CS_SPI (to ISP_7)
'      1: JMP1 (pin jumper to GND)
'      0: LED2 (output 1 to activate)
'

```

```

'Port D: LED port, 2nd hardware UART (UART1), two wire serial interface
'      7: LED3 (output 1 to activate)
'      6: LED4 (output 1 to activate)
'

```

```

'      5: n.c.
'      4: n.c.
'      3: TXD UART1
'      2: RXD UART1 (used for GPS)
'      1: TWI serial data (SDA for compass module)
'      0: TWI serial clock (SCL for compass module)
'
'Port E: Output compare for Timer3 (PWM), 1st hardware UART (UART0)
'      7: n.c.
'      6: n.c.
'      5: Output Compare OC3C (used for servo 6)
'      4: Output Compare OC3B (used for servo 2)
'      3: Output Compare OC3A (used for servo 1)
'      2: SPI_CS (to SPI_5)
'      1: TXD UART0 (used for USB/GSM/radiomodem)
'      0: RXD UART0 (used for USB/GSM/radiomodem)
'
'Port F: Analog inputs (Pre assignments)
'      7: ADC7 n.c.
'      6: ADC6 n.c.
'      5: ADC5 to ADC_1 (Flight bat voltage by voltage divider)
'      4: ADC4 to ADC_2 (Flight bat current from voltage drop at mOhm shunt, amplified
by Opamp)
'      3: ADC3 to ADC_3 (MPX 4115 pressure sensor output conditioned by Opamp)
'      2: ADC2 to ADC_4 (MPX 4115 reference voltage)
'      1: ADC1 to ADC_5
'      0: ADC0 to ADC_6
'
'Port G: n.c.
'
'Port H: Output compare for Timer4 (PWM), 3rd hardware UART (UART2)
'      7: n.c.
'      6: n.c.
'      5: Output Compare OC4C (used for servo 9)
'      4: Output Compare OC4B (used for servo 8)
'      3: Output Compare OC4A (used for servo 7)
'      2: n.c.
'      1: TXD UART2
'      0: RXD UART2 (used for Spektrum Remote Receiver A)
'
'Port J: 4th hardware UART (UART3)
'      7: n.c.
'      6: n.c.
'      5: n.c.
'      4: n.c.
'      3: n.c.
'      2: n.c.
'      1: TXD UART3.
'      0: RXD UART3 (used for Spektrum Remote Receiver B)
'
'Port K: Analog inputs ADC8-15.
'      7: n.c.
'      6: n.c.
'      5: n.c.
'      4: n.c.
'      3: ADC11 to OPC_ZA3
'      2: ADC10 to OPC_ZA2 (Copilot Z)
'      1: ADC09 to OPC_XY3 (Copilot Y)
'      0: ADC08 to OPC_XY2 (Copilot X)
'
'Port L: Output compare for Timer 5 (PWM)
'      7: n.c.
'      6: n.c.
'      5: Output Compare OC5C (used for servo 10)
'      4: Output Compare OC5B (used for servo 11)
'      3: Output Compare OC5A (used for servo 12)
'      2: n.c.

```



```

1: n.c.
0: n.c.

'RESET:      to ISP_5
.....

SERVO  01  02  03  04  05  06  07  08  09  10  11  12
OC     3A  3B  1A  1B  1C  3C  4A  4B  4C  5C  5B  5A
BEST   AR  AL  EL  RU  MO  nc  FR  FL  nc  BR  BL  nc
-----

[ Special port assignments CRUMB2560-based AR7212 ]-----
[ from www.chip45.com. Still in my Alpina 4001   ]

```

'Hardware-Requirements: What you need to solder:

(1) A Servo Connector board. This is a simple DIY circuit as follows:

```

S01 S02 S03 S04 S05 S06 S07 S08 S09 S10 S11 S12
*   *   *   *   *   *   *   *   *   *   *   *
*-----*-----*-----*-----*-----*-----* +5V
*-----*-----*-----*-----*-----*-----* GND

```

Each * stands for a standard 0.1" (2.54 mm) grid male header pin.
Recommended horizontal distance: 3 mm instead of 2.54 because
some servo connectors (female) have casings that do not fit into
the 0.1" grid. Vertical distance between pins must be 2.54 mm.

Alternatives, e.g. 2x6 on a breadboard with double 0.1" (5.08 mm) spacing
are absolutely fine - just know your fuselage dimensions yourself.
You may etch your own board at your requirements or just use a properly
sized breadboard.

(2) A 3.3V voltage regulator and two capacitors mounted to a small breadboard.
VIN (+5V) is connected to +5V at the Crumb module/Servo Connector.
GND is connected to GND of the Crumb module/Servo Connector, as well as to
the BLACK lines of the Spektrum satellites.
VOUT is going ONLY to the ORANGE lines of the Spektrum satellites.

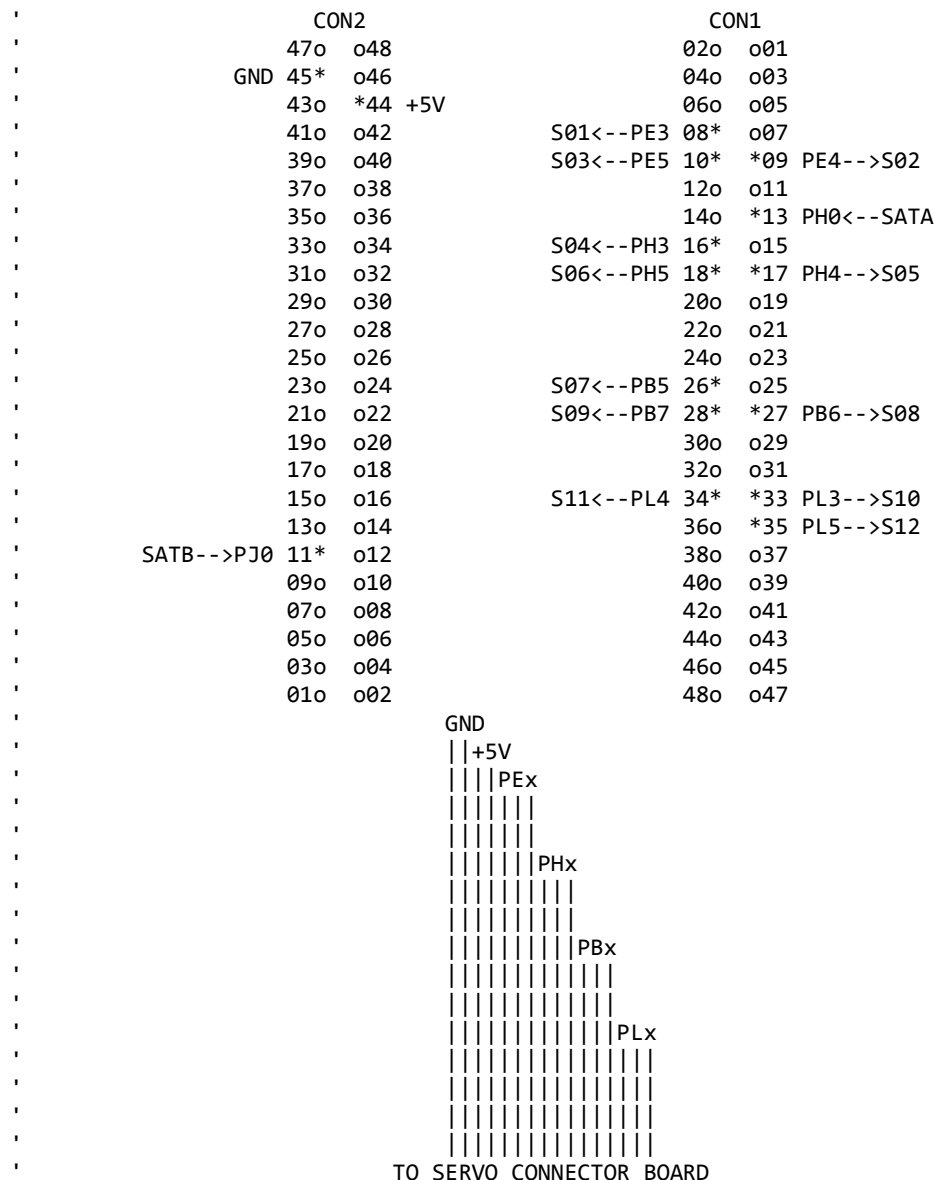
```

VIN (+5V)  --*-----*   *-----*-- VOUT (3.3V)   C1 = 10µ solid tantalum
              |         |         |         |         (care for proper polarity)
              C1    ##IC##    C2         IC = LT1127-3.3 (or similar)
              |         |         |         low dropout regulator
GND  --*-----*-----*-- GND         C2 = 3.3µ solid tantalum (pol.!)

```

(3) A Crumb2560 module from www.chip45.com. Order with 14.7 MHz crystal. Don't
forget a mini-USB cable (A <--> mini B, 5 Pin).
Connect the Crumb2560 to the servo board via a 15-20 cm 16-line flat
ribbon cable. It is recommended to use 2 lines for +5V and GND for safety
reasons, e.g. line 1&2 for +5V and line 3,4 for GND. Note that SATA and SATB are
the GREY lines to the Spektrum Satellites. TAKE SPECIAL CARE FOR THE PROPER
SETTING OF JUMPERS TO DISABLE THE MAX3221 ONBOARD. IF YOU EXPERIENCE PROBLEMS
HERE, CONNECT SATA TO USART1 RX (PD2 = Pin 40o @ CON1), AND CHANGE SW ACCOR-
DINGLY (USE COM2 instead of COM3).

BACKSIDE VIEW FOR SOLDERING THE CONNECTIONS



Cut the flat ribbon cable as shown - the vertical distance from GND to PLx is about $11 \times 0.4 = 4.4$ cm. That's it basically. I did the flatband to boards soldering directly just to save weight and size. Of course, one may design a 3x6cm sandwich-board for the crumb module carrying the servo connections, as well as the 3.3V needed for the satellites. It all depends on the space you have in your fuselage.

Port A: Standard: Address low byte of external RAM. Not used here.

Port B: 7-4: Output compare and PWM for Timer 2-0.

3-0: MISO MOSI SCK SS (may be used for pressure sensor SCP-1000 later)

7: Output Compare OC1C (used for servo 9)

6: Output Compare OC1B (used for servo 8)

5: Output Compare OC1A (used for servo 7)

Port C: Standard: Address high byte of external RAM. Not used here.

Port D: Timer clock input, 2nd hardware UART (UART1), two wire serial interface OR external hardware interrupts

3: TXD UART1

2: RXD UART1 (useable for GPS)

```

'      1: TWI serial data (SDA useable for compass module)
'      0: TWI serial clock (SCL useable for compass module)
'
'Port E: Timer clock inputs, output compar., analog comparator
' 1st hardware UART (UART0)
'      5: Output Compare OC3C (used for servo 3)
'      4: Output Compare OC3B (used for servo 2)
'      3: OutputCompare OC3A (used for servo 1)
'      1: TXD UART0 (@CP2101, useable for GSM/radiomodem)
'      0: RXD UART0 (@CP2101, useable for GSM/radiomodem)
'
'Port F: JTAG and analog inputs
' 3-0: Analog inputs: (useable for MPX fine, Vref, BatV, BatI)
'
'Port G: WR/RD/ALE ext. memory, TOSC2/TOSC1 for timer2, OC0B timer 0. Not used.
'
'Port H: RXD2/TXD2/XCK2 for UART2, OC4A-C for timer4, OC2B timer2, input timer4
'      5: Output Compare OC4C (used for servo 6)
'      4: Output Compare OC4B (used for servo 5)
'      3: Output Compare OC4A (used for servo 4)
'      1: TXD UART2 (@MAX3221)
'      0: RXD UART2 (@MAX3221, used for Spektrum Remote Receiver A)
'
'Port J: RXD3/TXD3/XCK3 for UART3, or PCINT9-15
'      7: LED
'      1: TXD UART3
'      0: RXD UART3 (used for Spektrum Remote Receiver B)
'
'Port K: Analog inputs ADC8-15 (not used here).
'
'Port L: Input capture trigger Timer 4&5, Timer 5 clock input, OCA5-OCC5
'      5: Output Compare OC5C (used for servo 12)
'      4: Output Compare OC5B (used for servo 11)
'      3: Output Compare OC5A (used for servo 10)
'-----
'
'-----[Special port assignments for the Xmega-A1-USB module]-----
'-----[from www.avr-praxis.de. Currently in the dev. stage ] -----
'
'Port A: Analog-digital Port A
'
'Port B: Analog-digital Port B
'
'Port C: Digital I/O Port C
'      0: SDA (I2C_C0: e.g. compass module)
'      1: SCL (I2C_C0: e.g. compass module)
'      2: RXD0 (USART_C0: COM1 - Sat A/)
'      3: TXD0 (USART_C0: COM1 - Sat A/n.c.)
'      4: SS (SPI_C1: SCP-1000)
'      5: MOSI (SPI_C1: SCP-1000)
'      6: MISO (SPI_C1: SCP-1000)
'      7: SCK (SPI_C1: SCP-1000)
'
'Port D: Digital I/O Port D
'      0: OC0A (PWM_D0: Servo 1)
'      1: OC0B (PWM_D0: Servo 2)
'      2: OC0C (PWM_D0: Servo 3)
'      3: OC0D (PWM_D0: Servo 4)
'      4: OC1A (PWM_D1: Servo 5)
'      5: OC1B (PWM_D1: Servo 6)
'      6: RXD1 (USART_D1: COM4 - Sat B)
'      7: TXD1 (USART_D1: COM4 - Sat B/n.c.)
'
'Port E: Digital I/O Port E
'      0: OC0A (PWM_E0: Servo 7)

```

```

'      1: OC0B (PWM_E0: Servo 8)
'      2: OC0C (PWM_E0: Servo 9)
'      3: OC0D (PWM_E0: Servo 10)
'      4: OC1A (PWM_E1: Servo 11)
'      5: OC1B (PWM_E1: Servo 12)
'      6: RXD1 (USART_E1: COM6 - GPS)
'      7: TXD1 (USART_E1: COM6 - GPS/n.c.)
'
'Port F: Digital I/O Port F
'      0: n.c.
'      1: n.c.
'      2: RXD0 (USART_F0: COM7 - USB or 2nd link)
'      3: TXD0 (USART_F0: COM7 - USB or 2nd link)
'      4: OC1A (PWM_F1: Servo 13 or n.c. when using 8 bit Timer for pause recognition)
'      5: OC1B (PWM_F1: Servo 14 or n.c. when using 8 bit Timer for pause recognition)
'      6: RXD1 (USART_F1: COM4 - CHR9d)
'      7: TXD1 (USART_F1: COM4 - CHR9d)
'
'Port H: Extended Bus Interface Port H
'
'Port J: Extended Bus Interface Port J
'
'Port K: Extended Bus Interface Port K
'
'Port R: Special Port R
'
'Port Q: Special Port Q
'      0:
'      1:
'      2: Button
'      3: Status LED
'-----
'
'-----
'----- Declaration of Constants -----
'-----
' TimerOverflowSeconds = 2^Nbit * prescale / crystalfreq, where
'                       = 2^16 * 8 / 16000000
'                       = 0.032768 s = 33ms
'AR7212 (14.7456 MHz):
'--Timer 1/3/4/5 MAX TCNT Overflow for 14.7456MHz is at 35.555555555 ms
'TOP for 22 ms is: 40550
'      1 ms is: 1843
'--Timer 0/2 overflow after 2^8 * 1024/14745600 = 17.778 ms
'1 Byte (10 bit with parity and 1 stopbit) at 115200 bps takes 1/11520 sec =
'0.0868 ms, 16 Byte gives 1.3889 ms. 7ms = 256 * 7/17.778 = 100.798 apprx. 100
'
'AR7212 (11.0592 MHz):
'--Timer 1/3/4/5 MAX TCNT Overflow for 11.0592MHz is at 47.407407407 ms
'TOP for 22 ms is: 30413 (30412.8)
'      1 ms is: 1382 (1382.4)
'--Timer 0/2 overflow after 2^8 * 1024/11059200 = 23.704 ms
'1 Byte (10 bit with parity and 1 stopbit) at 115200 bps takes 1/11520 sec =
'0.0868 ms, 16 Byte gives 1.3889 ms. 7ms = 256 * 7/23.704 = 75.6 apprx. 76
'
'AR7212 (16.0000 MHz):
'--Timer 1/3/4/5 MAX TCNT Overflow for 16MHz is at 32.768 ms
'TOP for 22 ms is: 44000
'      1 ms is: 2000
'--Timer 0/2 overflow after 2^8 * 1024/16000000 = 32.768 ms
'1 Byte (10 bit with parity and 1 stopbit) at 115200 bps takes 1/11520 sec =

```

'0.0868 ms, 16 Byte gives 1.3889 ms. 7ms = 256 * 7/32.768 = apprx. 109

'ARX7212 (32.0000 MHz):

'--Timer TCD0, TCD1, TCE0, TCE1 after $1/32000000 \cdot 2^{16} \cdot 8 = 16.384$ ms

#if Ver_cryst = 0
\$crystal = 14745600

Const Top = 40550

Const T7 = 100

given prescaler and quartz. Is 100 for 14.7 MHz

#endif

'defines 22 ms cycle time

'defines 7 ms with timer0/2 at

#if Ver_cryst = 1
\$crystal = 11059200

Const Top = 30413

Const T7 = 76

given prescaler and quartz. Is 76 for 11.059 MHz

#endif

'defines 22 ms cycle time

'defines 7 ms with timer0/2 at

#if Ver_cryst = 2
\$crystal = 16000000

Const Top = 44000

Const T7 = 109

given prescaler and quartz. Is 109 for 16 MHz

#endif

'defines 22 ms cycle time

'defines 7 ms with timer0/2 at

#if Ver_cryst = 3
\$crystal = 32000000

Const Top = 65535

Const T7 = 218

set to 8 bit.

#endif

'defines 16.38 ms cycle time

'defines 7 ms with timer TCF1

#if Ver_board = 0

'-----OC3A,B,C

Const Ser_oc3a = 1

Const Ser_oc3b = 2

Const Ser_oc3c = 3

'-----OC4A,B;C

Const Ser_oc4a = 4

Const Ser_oc4b = 5

Const Ser_oc4c = 6

'-----OC1A,B,C

Const Ser_oc1a = 7

Const Ser_oc1b = 8

Const Ser_oc1c = 9

'-----OC5A,B,C

Const Ser_oc5a = 10

Const Ser_oc5b = 11

Const Ser_oc5c = 12

'-----

#endif

'Crumb 2560

'Throttle

'Rudder

'Elevator

'Rightaileron

'Leftaileron

'Rightflapmid

'Leftflapmid

'Rightflapinn

'Leftflapinn

'Rightairbrak

'Leftairbrak

'Special

#if Ver_board = 1 Or Ver_board = 2 Or Ver_board = 4
Ralf Kull

'-----OC3A,B,C

Const Ser_oc3a = 1

Const Ser_oc3b = 2

Const Ser_oc3c = 6

'-----OC4A,B;C

Const Ser_oc4a = 3

Const Ser_oc4b = 4

Const Ser_oc4c = 5

'-----OC1A,B,C

Const Ser_oc1a = 7

Const Ser_oc1b = 8

Const Ser_oc1c = 9

'Dedicated AR7212 boards by

```

'-----OC5A,B,C
Const Ser_oc5a = 12
Const Ser_oc5b = 11
Const Ser_oc5c = 10
'-----
#endif
#if Ver_board = 3                                'Xmega-A1-USB
'-----TCD0A,B,C,D
Const Ser_tcd0a = 1
Const Ser_tcd0b = 2
Const Ser_tcd0c = 3
Const Ser_tcd0d = 4
'-----TCD1A,B
Const Ser_tcd1a = 5
Const Ser_tcd1b = 6
'-----TCE0A,B,C,D
Const Ser_tce0a = 7
Const Ser_tce0b = 8
Const Ser_tce0c = 9
Const Ser_tce0d = 10
'-----TCD1A,B
Const Ser_tce1a = 11
Const Ser_tce1b = 12
#endif

'-----keeping integrity during compilation of old version (April 09 changes)
'Const Aileron = 4
Const Dx7flap = 6
Const Dx7aileron = 2
Const Dx7elevator = 3
Const Dx7rudder = 4
Const Dx7gear = 5
Const Dx7throttle = 1
Const Dx7aux2 = 7

'-----APRIL 09 TO BE CHANGED
'Const Ppmcellrcswitch = 7                        ''formerly 8: will be the
channel number for AUX2 On DX7
'Const Ppmeasymodeswitch = 7                      ''Easy Mode = taking roll and
pitch from the sticks via Copilot-sensors
''to be reassigned in the

AR7212X
Const Ppmredbutton = 7                            ''Formerly 6: Is the knob for
autocalibration and trimming
''to be reassigned in the

AR7212X

Const Tccra = 170                                 '' Timer counter register A for
fast pwm, mode 14, (see later in the annotations)
Const Tccrb = 26                                  '' Timer counter register B for
fast PWM and prescaler 8 (see later)

Const Mpxout = 7
Const Mpxvref = 6
reference (potentiometer for 4V at 5V Vref)
Const Batamp = 5
Const Batvolt = 4
Const Fmacox = 8
left eyes
Const Fmacoy = 9
right eyes
Const Fmacoz = 10
'Const Xaccel = 9
'Const Xacce0 = 2
'Const Yaccel = 13

'Port F channel assignments:
'ADC: MPX pressure sensor value
'MPX pressure sensor voltage

'Battery amperage from shunt
'Battery voltage
'CoPilot's front right - rear
'CoPilot's front left - rear
'Copilots up/down eyes

```

```

'Const Yacce0 = 0

Const Cmaxchar = 82
$-command
Const Compass = 192

Const Revid = &H00
Const Datawr = &H01
(read/write)
Const Addptr = &H02
(read/write)
Const Oper = &H03
(read/write)
Const Opstat = &H04
Const Rstr = &H06
Const Status = &H07
Const Datard8 = &H1F
bit) or

EEPROM (read)
Const Datard16 = &H20
bit) or

indirect reg (read)
Const Tempout = &H21
data (read)

Const Cfg = &H00
(read/write)
Const Cfg2 = &H09

Const Modtst2 = &H2D
(WRITE)
Const Usrdata1 = &H29
(read/write)
Const Usrdata2 = &H2A
(read/write)
Const Usrdata3 = &H2B
(read/write)
Const Usrdata4 = &H2C
(read/write)
'

'-----
'----- Declaration of Subs and Functions -----
'-----

Declare Sub Readsatellites()
received from Spektrum satellite receivers
Declare Sub Readusart0()
loopistparser upon receipt of CR
Declare Sub Switchboard()
operations
Declare Sub Updateslaves()
Declare Sub Loopistparser()
programmer
Declare Sub Servoadjust()
the AR7212 programmer
Declare Sub Mixeradjust()
the AR7212 programmer
Declare Sub Printteeram()
programmer

'number of characters per GPS
'I2C make address of compass

' RevID register (read)
' Indirect reg access data
' Indirect reg access pointer
' Operation register
' Operation status (read/write)
' ASIC software reset (write)
' ASIC top-level status (read)
' Pressure output data (MSB 8-
' 8-bit data read from
' Pressure output data (LSB 16-
' 8-bit data read from
' 14-bit temperature output
' ATTENTION - is 0x22??????
' Config register-indirect
' MISO config (read/write)
' only for SCP1000-D01
' Noise level config-indirect
' User Data EEPROM indirect
' User Data EEPROM indirect
' User Data EEPROM indirect
' User Data EEPROM indirect

```

```

'.....'From here on extended
functionality
Declare Sub Loopistparser_ext()           'parses commands from AR7212
programmer and ground station, extension required
Declare Sub Setcommands()                 'parses set commands from the
ground station
Declare Sub Interpretedx7()               'NEEDS MAJOR OVERWORK FOR
AR7212x FUNCTIONALITY

DIFFERENT AUTONOMOUS FLIGHT MODES FROM DX7

'DECLARATION OF GLOBAL VARIABLES
'AND NOT FROM GROUND STATION
Declare Sub Readadc()                     'reads analog sensors
Declare Sub Readgps()                     'reads NMEA-strings from GPS
Declare Sub Autonomousservocontrol()      'computes
Auto_Rudder/Elevator/Aileron/Throttle from sensor data and settings
Declare Sub Ggaparse()                     'parses NMEA GGA string and
computes long integers for latitude and altitude

'in "decimal" format
Declare Sub Rmcparse()                     'parses NMEA RMC string for
heading and speed over ground
Declare Sub Homeset()                     'calculates navigation
parameter lone2m and late2m from home waypoint
Declare Sub Wayparse()                     'parses latitude, longitude,
altitude received in long integer format from GPS
Declare Sub Teledata()                     'processes analog sensor
information, computes altitude and attitude, and

'synthesizes telemetry strings
$MPX (4115), $BAT, $COP with GPS (xor) checksum.
Declare Sub Gsminit()                     'Initialization of Telit
modules GM862,... for CSD calls
Declare Sub Pressure_sensor_init()         'Initialization of Pressure
sensor SCP-1000: tbd
Declare Sub Getspi()                       'Test for reading SCP-1000 via
SPI: tbd
Declare Sub Direct_write()                 'Direct write of SCP-1000
registers
Declare Sub Direct_read()                  'Direct read of SCP-1000
registers
Declare Sub Indirect_write()               'Indirect write of SCP-1000
registers
Declare Sub Indirect_read()                'Indirect read of SCP-1000
registers
Declare Sub Read_status()                  'Read status register of
SCP_1000.

'was before LED output: shows
Declare Sub Errorbyservo(byval Kby As Byte)
GSM errors by servo moves
Declare Sub Isok(byval Atst As String , Byval Kst As String , Byval Llo As Long , Byval Kby
As Byte)

'analysis of GSM strings
Declare Function Xoradd(byval Xyzline As String) As String 'checksum routine
Declare Function I2cread8_w(byval I2cnode As Byte , Byval Location As Byte) As Word
'I2C connectivity for cmp03

compass (can be deactivated, not anymore called)
Declare Function Throttleoff(byval Pulse As Byte) As Byte 'throttle optimisation (can be
deactivated, motor control integrated into

'autonomousservocontrol
Declare Sub Restart()                       'Reinitialization of main loop
after loosing GSM connection

```

```

'-----
'----- Declaration of global variables -----
'-----

```

'General programming style follows PicNicks recommendations to achieve small and


```

'fast compiled code.
'general overwritables for SUBS/GOSUBS (I, J, K, L in several variable formats)
'take special care here!!!!
'Loop variables need to be not overwritten: LoopIby, Gpsiby,

Dim Hby As Byte
Dim Loopiby As Byte
Dim Iby As Byte
Dim Jby As Byte
Dim Kby As Byte
Dim Lby As Byte
Dim Gpsiby As Byte
Dim Loopkby As Byte
average of ADC registers
',
SRAM = 7

Dim Iwo As Word
Dim Jwo As Word
Dim Kwo As Word
Dim Lwo As Word
',
SRAM = 8

Dim Iin As Integer
Dim Jin As Integer
Dim Kin As Integer
Dim Lin As Integer
',
SRAM = 8

Dim Ilo As Long
Dim Jlo As Long
Dim Klo As Long
Dim Llo As Long
',
SRAM = 8

Dim Isi As Single
Dim Jsi As Single
Dim Ksi As Single
Dim Lsi As Single
',

Dim Msi As Single
Dim Nsi As Single
Dim Osi As Single
Dim Psi As Single
Dim Qsi As Single

',
SRAM = 18

Dim Loopist As String * Cmaxchar
from usart0!!!
Dim Ist As String * Cmaxchar
Dim Jst As String * Cmaxchar
Dim Kst As String * Cmaxchar
Dim Lst As String * Cmaxchar

Dim Gpsst As String * Cmaxchar
Dim Readline As String * Cmaxchar
GPS!!!!
Dim Rmcline As String * Cmaxchar
Dim Ggaline As String * Cmaxchar
Dim Mpxline As String * 40
USART0 output to ground
Dim Cmpline As String * 20
USART0 output to ground
Dim Batline As String * 20
string for USART0 output to ground
Dim Copline As String * 20
string for USART0 output to ground
'Dim Adxline As String * Cmaxchar
Dim Errline As String * Cmaxchar
',
SRAM = 920

'Dim D As Byte
Dim Readchar As String * 1

```

'counts chars in GPS string
'stores index for moving

' in main loop for instring

'Attention: Used in Telit PLUS
'NEW NAV : GPS \$RMC string
'NEW NAV : GPS \$GGA string
'Holds MPX pressure string for
'Holds CMP compass string for
'Holds BAT voltage/amperege
'Holds FMA Copilot sensor

```
Dim Readcharasc As Byte
Dim Gpsparseist As String * 12
temporary items separated by commas
```

```
'ATTENTION: KIJUNE09: FORMRLY 20 STRINGS IN ISTARR
```

```
Dim Istarr(10) As String * 30
string array holds items separated by blanks
', SRAM = 314
```

```
Dim Ggacnt0 As Long
position averaging
Dim Ggati0 As Long
GPS position averaging
Dim Ggati0d As Long
fix during GPS position averaging
Dim Ggati0e As Long
after midnite
Dim Ggalate As Long
decimal format multiplied by 10^6
Dim Ggalone As Long
Dim Ggafixe As Integer
Dim Ggasate As Integer
Dim Ggahode As Single
Dim Ggaalte As Single
',
```

```
SRAM = 36
```

```
Dim Waytime As Long
so far)
Dim Waylate As Long
format
Dim Waylone As Long
format
Dim Wayalte As Single
sea level
Dim Hometime As Long
Dim Homelate As Long
format
Dim Homelone As Long
format
Dim Homealte As Single
sea level
',
```

```
SRAM = 32
```

```
Dim Ggans As String * 1
north/south
Dim Ggaew As String * 1
east/west
Dim Gpsreadyflag As Byte
Dim Gsmreadyflag As Byte
Dim Restartflag As Byte
connection is lost or bad - somewhat superfluous
',
```

```
Dim Mpxoutp As Single
MPX sensor
Dim Mpxrefp As Single
calculated from voltage at spindle potentiometer
', SRAM = 9
```

```
Dim Mpxo(10) As Integer
sensor for moving average calculation
Dim Mpxr(10) As Integer
voltage for moving average calculation
Dim Batv(10) As Integer
for moving average calculation
Dim Bati(10) As Integer
amperage for moving average calculation
Dim Copx(10) As Integer
sensor for moving average calculation
Dim Copy(10) As Integer
sensor for moving average calculation
```

```
'GPS parser string holds
```

```
'Main loop (USART0) parser
```

```
'holds counter/seconds for GPS
```

```
'stores time of 1st fix during
```

```
'holds time passed since 1st
```

```
'time from NMEA GGA in seconds
```

```
'latitude from NMEA Late in
```

```
'longitude
```

```
'fix status
```

```
'number of satellites found
```

```
'horizontal error
```

```
'altitude in M above sea level
```

```
'Waypoint time (not really used)
```

```
'Waypoint latitude special
```

```
'Waypoint longitude special
```

```
'Waypoint altitude im m above
```

```
'Homepoint time (?)
```

```
'Homepoint latitude in special
```

```
'Homepoint longitude in special
```

```
'Homepoint altitude in m above
```

```
'GGA direction indicator flag
```

```
'GGA direction indicator flag
```

```
'was Bit: Set when GPS is ready
```

```
'was Bit: Set when GSM is ready
```

```
'was Bit: Set when GSM
```

```
'holds calculated pressure from
```

```
'holds reference pressure
```

```
'holds ADC from MPX pressure
```

```
'holds ADC from MPX reference
```

```
'holds ADC from battery voltage
```

```
'holds ADC from battery
```

```
'holds ADC from Copilot x
```

```
'holds ADC from Copilot y
```

Dim Copz(10) As Integer		'holds ADC from Copilot z
sensor for moving average calculation		
'Dim Xacc(10) As Integer		
'Dim Yacc(10) As Integer		
,		
	SRAM = 140	
Dim Mpxsumo As Integer		'holds corresponding sum of 10
measurements		
Dim Mpxsumr As Integer		'holds corresponding sum of 10
measurements		
Dim Batsumv As Integer		'holds corresponding sum of 10
measurements		
Dim Batsumi As Integer		'holds corresponding sum of 10
measurements		
Dim Copsumx As Integer		'holds corresponding sum of 10
measurements		
Dim Copsumy As Integer		'holds corresponding sum of 10
measurements		
Dim Copsumz As Integer		'holds corresponding sum of 10
measurements		
'Dim Adxxac0 As Integer		
'Dim Adxxacc As Integer		
'Dim Adxyacc As Integer		
'Dim Adxyac0 As Integer		
,		
	SRAM = 12	
Dim Mpxtransa As Single		'Transfer function: Mpx_x_p
=(Mpxsumx / (10*1024) + Mpxtransb) / Mpxtransa		
Dim Mpxtransb As Single		
Dim Mpxgain As Single		'Gain during pressure calc
(=22)		
Dim Mpxdiv As Single		'somewhat superfluous: = 10 *
1024		
Dim Batvdiv As Single		'theoretically 1024/ 5 * 3 V
(or 2 if jumper closed)		
Dim Batidiv As Single		'theoretically 1024 / 5 V (from
50 mV at a 1 mOhm Shunt with		
		'50 A - amplified with a Gain
= 100)		
Dim Adc7v As Single		'ADC read of Vcc
Dim Bat_v As Single		'Battery voltage
Dim Bat_i As Single		'Battery amperage
Dim Vcc As Single		'= 5 (V)
Dim Altp0 As Single		'pressure at sea level (ca. 110
kPa)		
Dim Alta As Single		'temperature gradient (ca.
0.006 K/m)		
Dim Alth As Single		'altitude in m
Dim Altt As Single		'normal temperature (ca. 288K)
Dim Altn As Single		'reference factor for standard
model (ca. 5)		
Dim Alth0 As Single		'Altitude at ground
,		
	SRAM = 64	
'Dim Heading As Word		'variable to hold compass value
Dim Readable_heading As Single		'earlier taken from compass,
now from GPS heading		
Dim Atst As String * 20		'modem string in ISOK routine
'Dim Servonumber As Byte		'Autonomous control now
simulated "as if" from DX7		
		'ATTENTION: Auto_ (INTEGER)
was formerly Pulse_ (WORD)		
Dim Auto_min As Integer		'minimal stick reading (-512)
Dim Auto_max As Integer		'maximal stick reading (-512)
Dim Auto_rudder As Integer		'simulated rudder stick
Dim Auto_aileron As Integer		'simulated aileron stick
Dim Auto_elevator As Integer		'simulated elevator stick

```

Dim Auto_throttle As Integer           'simulated throttle stick
Dim Auto_ruddercenter As Integer       'Rudder center as if from DX7
Dim Auto_elevatorcenter As Integer     'Elevator center as if from DX7
Dim Auto_throttlecenter As Integer     'Throttle center (???) as if
from DX7
Dim Auto_aileroncenter As Integer      'Aileron center as if from DX7
Dim Auto_throttleopt As Integer        'Throttle optimum for keeking
altitude as if from DX7
Dim Auto_dx7halfresolution As Integer  'Half range resolution of the
DX7: 128 + 256 = 384
Dim Mixer_max As Integer               'Maximal mixer value to keep
everything within integer range: 32768/384 = 2^15/Auto_max
'Dim Actualservoword As Word
'Dim Xaccoffin As Integer
'Dim Yaccoffin As Integer
'Dim Xaccneutr in As Integer
'Dim Yaccneutr in As Integer

'Dim Joystickservoloopflag As Byte     'was Bit
Dim Autonomousservocontrolflag As Byte 'was Bit: MOST IMPORTANT FLAG:
setting autonomous control via DX7's AUX2 switch
Dim Waysetflag As Byte                'was Bit: Waypoint flag is
either set from GC or from DX7 when in boxflight mode (tbd)
Dim Homesetflag As Byte               'was Bit: Homepoint flag is set
when homepoint becomes waypoint
Dim Speedsetflag As Byte              'was Bit: Speedset flag is set
by GC - somewhat superfluous
Dim Altitudesetflag As Byte           'was Bit: Altitudeset flag is
set by GC or during trim on ground
Dim Rmcspeed As Single                'NEW NAV: Speed over ground
from GPS RMC
Dim Rmcheading As Single              'NEW NAV: Heading from GPS RMC
Dim Knottokmh As Single               'NEW NAV: Factor for km/h from
knots
Dim Normspeed As Single               'is a model specific parameter
defining speed over ground on calm day
Dim Speedfactor As Single             '=actual speed/normspeed
'Dim Knottoms As Single
'Dim Headingfixflag As Byte          'NEW NAV was Bit
Dim Itimer3 As Word                  'NEW NAV: is the counter for
evaluating the red button trim knob (needs 5 x 20 ms)
Dim Jtimer3 As Word                 'is the counter for doing ADC
read (now each cycle)
Dim Ktimer3 As Word                 ' is the Counter for printing
servo signals bitwise to ground (10x 4or5 bytes per second)

Dim Motorofftimer As Word            'TO BE CHECKED!!!:
Motorofftimer counts the receival of a GPS GGA string and is set zero after
                                     'third
receival!!!!????
'Dim Storetimer3 As Word
'Dim Adxflag As Integer
'Dim Gpsflag As Integer
Dim Error As Byte                   'only needed for Stack overflow
checking!!!!!!!
',
                                     SRAM = 70
Dim Navsetyaw As Single              'the direction to go: CHECK
CALCULATION (BETTER ONLY TAKEN FROM RMCHEADING, WHY FROM GGA?!!!!)
Dim Navsetspeed As Single            'Speed is set only by GC
Dim Navsetaltitude As Single         'altitude only by GC - CHECK
WHY NOT IN BOX FLIGHT MODE?!!!!!!
Dim Navsetroll As Single             'roll angle to keep
Dim Navsetpitch As Single            'pitch angle to keep
Dim Maxroll As Single               'maximal roll angle - model
specific parameter!!! Currently 40
Dim Maxyaw As Single                'maximal yaw angle. Currently
30. TO BE CHECKED!!!

```

Dim Maxpitch As Single	'maximal pitch angle: Currently
5. TO BE CHECKED!!!	
Dim Maxdist As Single	'distance from homepoint above
homepoint is set automatically (currently 400 m)	
'Dim Calibrationflag As Byte	'was bit: formerly compass
calibration. Removed on 30.08.09	
'Deleted !!!!Ki 30.08.09	'no direct steering from GC
joystick anylonger	
'Dim Rudder_traf As Single	'Travel-Factors
'Dim Elevator_traf As Single	'to be multiplied
'Dim Throttle_traf As Single	'by 128 to give
	'Half-Travels
Dim Late2m As Single	'conversion factor: latitude °
to m, computed from homepoint	
Dim Lone2m As Single	'conversion factor: longitude °
to m, computed from homepoint	
'!!!!Ki 30.08.09 deleted	
'Dim Maxspeed As Single	
'Dim Maxaltitude As Single	
'	
SRAM = 56	
Dim Waypointradius As Single	'Radius of Waypoint in Meter
deciding whether waypoint was met	
Dim Gpsallflag As Byte	'was Bit: Decides if all GPS
information is sent to GC	
'Dim Justdirectflag As Byte	
'!!!!Ki 30.08.09 deleted	
'Dim Servoprint As String * 10	
Dim GsmSignal As String * 10	'holds gsm signal quality
'Dim Iwaypoint As Byte	
Dim Motoroffflag As Byte	'is set if current is too high
or if battery voltage is too low	
Dim Rof As Single	'roll factor (formerly 20):
roll angle * rof gave servo setting	
200counts = 100µs.	'e.g. 10° x 20(counts/°) =
7.68 ca. 8	'New ROF is 20/2000*6*128 =
Dim Pif As Single	'pitch factor (see above)
Dim Sqr2 As Single	'= SQR(2)
Dim Roz As Single	'Roll zero is the roll offset
gained from trimming Copilot sensors	
Dim Piz As Single	'Pitch zero is the pitch offset
gained from trimming Copilot sensors	
Dim Autotrimflag As Byte	'Flag to set Autotrim: Either
by GC or by pressing the trim knob (need to be done	
Dim Integrationflag As Byte	'differently in AR7212x
and averaging of copilot sensors. Right now only by GC.	'Flag to set ADC integration
Dim Waypointreachedflag As Byte	
waypoint. Currently only set and cleared.	'Flag set when reaching
notification of GC. Somewhat SUPERFLUOUS.	'Only action is the
'Dim Time0 As Long	
Dim Alivetime As Long	'Latency time is evaluated by
GC, currently not by the AR7212. Somewhat SUPERFLUOUS	
'Dim Nogsmcounter As Word	
'Dim Servostringcounter As Word	

'Dim Actualdistance As Single	
'Dim Windspeed As Single	
'Dim Winddirection As Single	
Dim Alt_downfromnavset As Single	'Altitude window border below
waypoint altitude	
Dim Alt_upfromnavset As Single	'Altitude window border above
waypoint altitude	
Dim Regulatormode As Byte	'Altitude regulator mode by
which altitude is kept constant via throttle	
'Dim Ppmin(10) As Word	'May 1 2007: PPMIN read in
from RX	
'Dim Automaster(7) As Integer	'End august 2009: NEW simulated
DX7 format of channels (-512)	
'Dim Ppminwo As Word	'
'Dim Int6_isr_delay As Word	'
'Dim Ppmokflag As Byte	' Set if PPM signal is OK
(within time window and without timer overflow:	
AR7212X	' NEEDS ADAPTATION FOR THE
'Dim Ipulse As Byte	'
'Dim Npulse' As Byte	'
Dim Maxnpulse As Byte	'Should be better coined as
MaxDX7channel (= 7)	
'Dim Timer3_ipulse As Byte	
'	
	SRAM = 54
Dim Failcount As Word	'Counts the number of
occurences of bad frames	
SATELLITES AND REMOVED	' SHOULD BE SHIFTED INTO READ
Dim Rmcvalidflag As Byte	' FROM INTERPRETE PPM
valid, used in navigation	'Set if GPS RMC message is
Dim Easymodeflag As Byte	'EASY MODE (=take pitch and
roll angles directly from the sticks)	
switch:	'was formerly set by an FX18-
no setting via Interpret PPM!	'NEEDS TO BE ZERO IN AR7212X -
Dim Flightstatusflag As Byte	
just started, 3 airborne above a certain altitude	'3 state Flag: 1 on Ground., 2
STABILISATION DURING START	'COULD BE USEFUL FOR FLIGHT
Dim Redbuttonflag As Byte	
button on FX18 - NEEDS REWORK FOR DX7	'Flag set by pressing the Trim
Dim Ail2lonefactor As Single	'computed (like lone2m,
late2m), from homepoint coordinates.	
RECHECK	'A bit confusing: NEEDS A
Dim Ele2latefactor As Single	
Dim Thr2altefactor As Single	'see above
 	'still a bit confusing
Dim Ail2rollfactor As Single	
proportionality between aileron stick setting and roll angle deviation	'Factor determining the
Dim Ele2pitchfactor As Single	'Factor determining the
proportionality between elevator stick setting and pitch angle deviation	
'Dim Sumpulse As Word	
'	
	SRAM = 30
Dim Rxahiby As Byte	
satellite A	'High byte of word received by

Dim Rxbhiby As Byte	'High byte of word received by
satellite B	
Dim Rxaloopby As Byte	'Loop counter (1..8) within
frame received by satellite A	
Dim Rxbloopby As Byte	'Loop counter (1..8) within
frame received by satellite B	
Dim Rxaloopflag As Byte	'Set by 1st byte from satellite
A, toggled by the following bytes	
Dim Rxbloopflag As Byte	'Set by 1st byte from satellite
B, toggled by the following bytes	
Dim Rxastartwo As Word	'Startword encoding signal
quality and receiver status of satellite A	
Dim Rxbstartwo As Word	'Startword encoding signal
quality and receiver status of satellite B	
Dim Rxainwo(8) As Word	'Array of received words from
satellite A	
Dim Rxbinwo(8) As Word	'Array of received words from
satellite B	
Dim Rxamaster(16) As Word	'Received words from satellite
A decoded into DX7 channels	
Dim Rxbmaster(16) As Word	'Received words from satellite
B decoded into DX7 channels	
Dim Master(7) As Integer	'The best set of DX7 channel
data taken from satellites A or B (decided by startword analysis)	
Dim Slave(12) As Integer	'The set of servo signals
'Dim Dummy As Eram Word	
'	
	SRAM = 150
#if Ver_board = 3	
Config Eeprom = Mapped	
#endif	
Dim Pilotnames As String * 20	'Pilot name string in SRAM
Dim Pilotnamee As Eram String * 20	'Pilot name string in EERAM
Dim Modelnames As String * 20	'Model name string in SRAM
Dim Modelnamee As Eram String * 20	'Model name string in EERAM
Dim Servonames(12) As String * 5	'Servo name strings in SRAM
Dim Servonamee(12) As Eram String * 5	'Servo name strings in EERAM
Dim Fltmodenames(6) As String * 30	'Flighmode name strings in SRAM
Dim Fltmodenamee(6) As Eram String * 30	'Flighmode name strings in
EERAM	
'	
'	
	SRAM = 280
	ERAM = 280
Dim Mixdat(72) As Byte	'Actual mixer adjust data in
SRAM	
Dim Mixdatm1(72) As Eram Byte	'EERAM mixer adjust data for
Flightmode 1	
Dim Mixdatm2(72) As Eram Byte	'EERAM mixer adjust data for
Flightmode 2	
Dim Mixdatm3(72) As Eram Byte	'EERAM mixer adjust data for
Flightmode 3	
Dim Mixdatm4(72) As Eram Byte	'EERAM mixer adjust data for
Flightmode 4	
Dim Mixdatm5(72) As Eram Byte	'EERAM mixer adjust data for
Flightmode 5	
Dim Mixdatm6(72) As Eram Byte	'EERAM mixer adjust data for
Flightmode 6	
Dim Mixtyp(72) As Byte	'Actual mixer type data in SRAM
Dim Mixtypm1(72) As Eram Byte	'EERAM mixer type data for
Flightmode 1	
Dim Mixtypm2(72) As Eram Byte	'EERAM mixer type data for
Flightmode 2	

Dim Mixtypm3(72) As Eram Byte Flightmode 3	'EERAM mixer type data for
Dim Mixtypm4(72) As Eram Byte Flightmode 4	'EERAM mixer type data for
Dim Mixtypm5(72) As Eram Byte Flightmode 5	'EERAM mixer type data for
Dim Mixtypm6(72) As Eram Byte Flightmode 6	'EERAM mixer type data for
Dim Offsets(12) As Integer in SRAM	'Actual servo channel offsets
Dim Offsetm1(12) As Eram Integer for Flightmode 1	'EERAM servo channel offsets
Dim Offsetm2(12) As Eram Integer for Flightmode 2	'EERAM servo channel offsets
Dim Offsetm3(12) As Eram Integer for Flightmode 3	'EERAM servo channel offsets
Dim Offsetm4(12) As Eram Integer for Flightmode 4	'EERAM servo channel offsets
Dim Offsetm5(12) As Eram Integer for Flightmode 5	'EERAM servo channel offsets
Dim Offsetm6(12) As Eram Integer for Flightmode 6	'EERAM servo channel offsets
Dim Delays(12) As Integer SRAM	'Actual servo channel delays in
Dim Delaym1(12) As Eram Integer Flightmode 1	'EERAM servo channel delays for
Dim Delaym2(12) As Eram Integer Flightmode 2	'EERAM servo channel delays for
Dim Delaym3(12) As Eram Integer Flightmode 3	'EERAM servo channel delays for
Dim Delaym4(12) As Eram Integer Flightmode 4	'EERAM servo channel delays for
Dim Delaym5(12) As Eram Integer Flightmode 5	'EERAM servo channel delays for
Dim Delaym6(12) As Eram Integer Flightmode 6	'EERAM servo channel delays for
Dim Centers(12) As Integer	'Servo center settings in SRAM
Dim Centere(12) As Eram Integer	'Servo center settings in EERAM
Dim Utravels(12) As Integer in SRAM	'Servo normal uptravel settings
Dim Dtravels(12) As Integer settings in SRAM	'Servo normal downtravel
Dim Utravele(12) As Eram Integer in EERAM	'Servo normal uptravel settings
Dim Dtravele(12) As Eram Integer settings in EERAM	'Servo normal downtravel
Dim Utramaxs(12) As Integer settings in SRAM	'Servo maximal uptravel
Dim Dtramaxs(12) As Integer settings in SRAM	'Servo maximal downtravel
Dim Utramaxe(12) As Eram Integer settings in EERAM	'Servo maximal uptravel
Dim Dtramaxe(12) As Eram Integer settings in EERAM	'Servo maximal downtravel
'	
SRAM = 312	
'	
ERAM = 2136	
'	
Sum SRAM (01.09.09) = 2528 (of 8192 in AT2560, of 4096 in AT1280) OK	
'	
SUM ERAM (01.09.09) = 2416 (of 4096 in AT2560, of 4096 in AT1280) OK	
Dim Rxabflag As Byte	'holds which satellite (A=0,
B=1) gave the best signal	


```

Dim Rxaquality As Byte 'holds signal quality of
satellite A
Dim Rxbquality As Byte 'holds signal quality of
satellite B
Dim Rxastatus As Byte 'holds receiver status of
satellite A
Dim Rxbstatus As Byte 'holds receiver status of
satellite B
Dim Rxdoneflag As Byte 'RXdoneflag is set after the
receival of a data frame fom RXA or RXB
Dim Fltmode As Byte 'holds the actual flight mode
as selected by DX7 switches
Dim Oldfltmode As Byte 'holds the previous flight mode
'Dim Ringcounter As Word
Dim Rxstarttwo As Word 'holds the best recent
startword from either sat A or B
'Dim Rxasync As Byte
Dim Dx7printflag As Byte 'indicates whether the AR7212
should print a received frame of HEX words to the AR7212 programmer
Dim Mixtype As Byte 'overwritable (holds the mixer
type (0..5) in the communication of AR7212 with AR7212 programmer)
Dim Nin As Integer 'additional general
overwritable
Dim Nwo As Word 'additional general
overwritable

Dim Bat_imax As Single 'Limit of maximal battery
current above which motor is switched off.
Dim Bat_vmin As Single 'Limit of minimal battery
voltage below which motor is switched off
Dim Regaddr As Byte
Dim Iobyte As Byte
Dim Numbits As Byte
Dim SpiBy(10) As Byte 'Byte array for communication
with SCP-1000 D01
Dim Spinword As Word
Dim Eeprom As Byte
Dim Indregaddr As Byte
Dim Inddata As Byte
'Dim Baudjmp As Byte

'----- This needs to be checked occasionally (STCHECK or DBG)-----

$hwstack = 512 ' 1024 default - originally 32
for the hardware stack

' hwstack tested: problems

start at 128.
$swstack = 128 ' 128 ... 64 default -
originally 10 for the SW stack
$framesize = 512 ' 256 .. 128 default -
originally 40 for the frame space

' could be of cause smaller...

'-----
'----- Input and output pins -----
'----- CONFIGS -----
'-----
,

#if Ver_board = 0
Config Pinj.7 = Output 'ATTENTION CONFIG PORTJ!!!!
' Config Pinh.0 = Output 'port not defect: output capabilities checked 090525
Mainled Alias Portj.7
#endif

```

```

#if Ver_board = 1 Or Ver_board = 2
'Port C: LED and configuration port.
'
'       7: CS-SD (connect to SW1)
'       6: SW1   (Switch to GND)
'       5: LED2  (to GND)
'       4: LED3  (to GND)
'       3: LED4  (to GND)
'       2: n.c.
'       1: JMP1  (pin jumper to GND)
'       0: JMP2  (solder bridge to GND)
'
'Port D: Configuration, 2nd hardware UART (UART1), two wire serial interface
'
'       7: JMP4  (solder bridge to GND)
'       6: JMP3  (solder bridge to GND)
'semantic definitions
'
'   LED1 : power ON
' LED2 (C5) : RX: Toggle in nonautonomous mode, ON in autonomous mode: Mainled
' LED3 (C4) : GPS: Toggle: Gpsled
' LED4 (C3) : Set if ready for take off: Readyled
'   JMP1 : Set to GND for Baudrate = 115200: Reset needs the Baud directive

'DDRs
Config Portc = &B10111000
Config Portd = Input

Mainled Alias Portc.5
Gpsled Alias Portc.4
Readyled Alias Portc.3
Baudjmp Alias Portc.1
#endif

#if Ver_board = 4
'Port C: Configuration port & Main-LED
'
'       7: CS-SD (pin_2 boxed connector, superfluous?)
'       6: SW1   (Switch to GND)
'       5: JMP2  (solder bridge to GND, tbd)
'       4: JMP3  (solder bridge to GND, tbd)
'       3: JMP4  (solder bridge to GND, tbd)
'       2: CS_SPI (to ISP_7)
'       1: JMP1  (pin jumper to GND)
'       0: LED2  (output 0 to activate)
'
'Port D: LED port, 2nd hardware UART (UART1), two wire serial interface
'
'       7: LED3  (output 0 to activate)
'       6: LED4  (output 0 to activate)
'       5: n.c.
'       4: n.c.
'       3: TXD UART1
'       2: RXD UART1 (used for GPS)
'       1: TWI serial data (SDA for compass module)
'       0: TWI serial clock (SCL for compass module)
'DDRs
Config Portc = &B10000101
Config Portd = Output

Mainled Alias Portc.0
Gpsled Alias Portd.7
Readyled Alias Portd.6
Baudjmp Alias Portc.1
#endif

#if Ver_board <> 3
'----- Hardware UARTS and serial communication buffers -----
'1. Usart (UART0): PE0 = RXD, PE1 = TXD
'2. Usart (UART1): PD2 = RXD, PD3 = TXD, Com2
'3. Usart (UART2): PH0 = RXD, PH1 = TXD, Com3
'4. Usart (UART3): PJ0 = RXD, PJ1 = TXD, Com4 (shielded in AR7212.log test, Rxabflag = 1)

```

```
'The 2560 has an extended UART.
'when CONFIG COMx is not used, the default N,8,1 will be used
Config Com1 = 115200 , Synchron = 0 , Parity = None , Stopbits = 1 , Databits = 8 ,
Clockpol = 0      'USB
Config Com2 = 4800 , Synchron = 0 , Parity = None , Stopbits = 1 , Databits = 8 , Clockpol
= 0      'GPS
'Config Com2 = 115200 , Synchron = 0 , Parity = None , Stopbits = 1 , Databits = 8 ,
Clockpol = 0      'was used for SatB at Crumb2560 in the interim
Config Com3 = 115200 , Synchron = 0 , Parity = None , Stopbits = 1 , Databits = 8 ,
Clockpol = 0      'Spektrum Remote Receiver A
Config Com4 = 115200 , Synchron = 0 , Parity = None , Stopbits = 1 , Databits = 8 ,
Clockpol = 0      'Spektrum Remote Receiver B
```

Enable Interrupts

```
Config Serialin = Buffered , Size = 128      'was 40 before
Config Serialout = Buffered , Size = 128     'was 255 before
Config Serialin1 = Buffered , Size = 128     'was 255 before
Config Serialin2 = Buffered , Size = 32
Config Serialin3 = Buffered , Size = 32
'Set Ucsr2b.7      'did not solve unavailability
of port (Crumb)
'Open "Com1:" For Binary As #1      'is always open
Open "Com2:" For Binary As #2
Open "Com3:" For Binary As #3
Open "Com4:" For Binary As #4
```

Enable Serial

```
'DECOMMENT FROM HERE....
'----- AD conversion at Port F -----
'Config Adc = Single , Prescaler = Auto , Reference = Avcc , Reference = Off
'formerly AVCC then OFF
'Analog digital conversion (Fort F)
,
'Start Adc
'!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

'----- TWI at Port D (also AR7212)-----
'Config Sda = Portd.1
'Config Scl = Portd.0
'Config I2cdelay = 10      'CHECK FOR AR7212
'NEW NAV
'TO HERE FOR SOFTWARE EXTENSIONS

'----- Config Watchdog for 2 seconds -----
Config Watchdog = 2048
Config Timer2 = Timer , Prescale = 1024
Config Timer0 = Timer , Prescale = 1024
```

#else

```
$lib "xmega.lib"
$external _xmegafix_clear
$external _xmegafix_rol_r1014
```

```
Config Osc = Enabled , 32mhzosc = Enabled      'remarkably accurate given the
fact that no xtal is employed
'Config Osc = Enabled , Extosc = Enabled , Range = 12mhz_16mhz , Startup = Xtal_16kclk (so
far doesn't work)
'configure the systemclock
Config Sysclock = 32mhz , Prescalea = 1 , Prescalebc = 1_1
Config Com7 = 38400 , Mode = Asynchronous , Parity = None , Stopbits = 1 , Databits = 8
Config Input7 = Cr , Echo = CrLf      ' CR is used for input, we echo
back CR and LF
Config Com1 = 115200 , Mode = Asynchronous , Parity = None , Stopbits = 1 , Databits = 8
```

```

Config Input1 = Cr , Echo = CrLf          ' CR is used for input, we echo
back CR and LF
Config Com4 = 115200 , Mode = Asynchronous , Parity = None , Stopbits = 1 , Databits = 8
Config Input4 = Cr , Echo = CrLf          ' CR is used for input, we echo
back CR and LF
Open "COM7:" For Binary As #1
Open "COM1:" For Binary As #3
Open "COM4:" For Binary As #4

'Config Portf = Output
'Config Portf.3 = Output                  ' TX pin must be output
'Config Portf.2 = Input
Config Portq = Output
Mainled Alias Portq.3

'Config Eeprom = Mapped                  ' shifted before EEPROM Dims:
when using EEPROM , add this config command
'configure the priority
'config priority=static|roundrobin,vector=application|boot,HI=enabled|disabled,
LO=enabled|disabled,ME=enabled|disabled
Config Priority = Static , Vector = Application , Lo = Enabled

'test an interrupts
On Usartf0_rxc Rxf0_isr
Enable Usartf0_rxc , Lo
Enable Interrupts
'Iby = Inkey(#1)
Goto After_isr

Rxf0_isr:
  Toggle Mainled
Return

After_isr:
#endif

#if Ver_board = 1 Or Ver_board = 2 Or Ver_board = 4          '2nd link during airtime only
supported for boards by Ralf Kull
#if Ver_uart0 = 0
If Baudjmp = 1 Then
  Baud = 38400          'Telit
Else
  Baud = 115200
End If
#endif
#if Ver_uart0 = 1
If Baudjmp = 1 Then
  Baud = 9600          'Telit GM..
Else
  Baud = 115200
End If
#endif
#endif

'-----
'----- PRESSURE SENSOR INITIALIZATION -----
'----- DECOMMENT FOR SOFTWARE EXTENSION -----
'-----
' Pressure_sensor_init

'-----
'----- READ DATA FROM EERAM -----
'-----
Pilotnames = Pilotnamee

```

```

Modelnames = Modelnamee
For Iby = 1 To 6
    Fltmodenames(iby) = Fltmodenamee(iby)
Next Iby
For Iby = 1 To 12
    Servonames(iby) = Servonamee(iby)
Next Iby
For Iby = 1 To 72
    Mixdat(iby) = Mixdatm1(iby)
Next Iby
For Iby = 1 To 12
    Offsets(iby) = Offsetm1(iby)
    Delays(iby) = Delaym1(iby)
Next Iby
For Iby = 1 To 12
    Centers(iby) = Centere(iby)
    Utravels(iby) = Utravele(iby)
    Dtravels(iby) = Dtravele(iby)
    Utramaxes(iby) = Utramaxe(iby)
    Dtramaxes(iby) = Dtramaxe(iby)
Next Iby

```

'=====TOTAL SRAM GLOBALS: SRAM = 1738 of 4096

```

Auto_dx7halfresolution = 384
Auto_min = 0 - Auto_dx7halfresolution
Auto_max = 0 + Auto_dx7halfresolution
Mixer_max = 32767 / Auto_max
everything within integer range:

```

'Maximal mixer value to keep

'= 32768/384 = 2^15/Auto_max =

```

85
Auto_rudder = 0
Auto_elevator = 0
Auto_aileron = 0
Auto_throttle = Auto_min
Auto_ruddercenter = 0
Auto_elevatorcenter = 0
Auto_throttlecenter = 0
Auto_throttleopt = 0
Auto_aileroncenter = 0

```

```

Isi = Auto_max
Jsi = Auto_throttleopt
Isi = Isi - Jsi
Alt_downfromnavset = Sqr(isi)
Isi = Auto_throttleopt
Jsi = Auto_min
Isi = Isi - Jsi
Alt_upfromnavset = Sqr(isi)

```

'----- Floating point "constants" -----

```

'Rudder_traf = 6.2                                'for direct steering
'Elevator_traf = 6.2
'Throttle_traf = 6.2

```

```

Vcc = 5
Mpxtransa = 0.009
Mpxtransb = 0.095
Mpxgain = 22
Mpxdiv = 10240
'Batvdiv = 682.67
jumper closed)

```

'1024 * 10 / 5 / 3 V (or 2 if

'Batvdiv = 614.4
2 if jumper closed)
resistances of the

3.78k

divideer viz.

jumper open.

R2=0, R3 = 3.78

452.4 as batvdiv.

precision measurements!!!

SOLDERING!!!

'Batvdiv = 699.45
7.33 - displayed is 7.81
'Batvdiv = 745.253
correct values

Batvdiv = 743.23
measured (VC10: 7.35, displayed 7.33) so another refinement.

is recommended for 5.5 Volt -

in groud unit every 10 second

Batidiv = 204.8
mOhm Shunt with
100)

"calibration":

mx + b)

negligible since

0.2% while

max. 10 mv.

VC 120.

Altp0 = 101.315
Alta = 0.00651
Altt = 288
Altn = 5.255

Knottokmh = 1.853
Normspeed = 36

Late2m = .111
Lone2m = .070
'Maxspeed = 35
'Maxaltitude = 200
Maxdist = 400
Waypointradius = 10

'Corrected: 1024/ 5 * 3 V (or

'still wrong - the MEASURED

'voltage divider are:

'from + to -:

'R1 = 4.57k, R2 = 4.56k, R3 =

'giving 3.78/12.91=1/3.4153 as

'1024 / 5 * 3.4153 = 699.45 for

'Jumper closed: R1 = 4.57,

'giving 1/2.209 as divider and

'NEVER TAKE 5% Resistors for

'ALWAYS MEASURE BEFORE

'STILL WRONG!!! measured is

'is needed to arrive at

'was still not correct -

'cut down of engine for 2 Lipos

'will do it at 5.8 V and warn

'1024 / 5 V (from 50 mV at a 1

'50 A - amplified with a Gain =

'new transfer function after

'ADC = 205.2 * Volt - 1.51 (Y =

'viz. Volt = (ADC + 1.51)/205.2

'The difference is almost

'the slope deviation is approx

'the offset is 1.5 bit, viz.

'Below precision of a Voltcraft

'kPa

'K/m

'K

'NEW NAV

'km/h (=10m/s)

```

(Cell-RC on Easystar):
Maxroll = 40
Maxyaw = 30
Maxpitch = 5
Rof = 8
formerly -7 , then +10 (15 was OK on 1stAF, then +20 but lateral phygoids)
Pif = 8
formerly 7 , then +10 (15 was OK on 1stAF, then +20)
Sqr2 = Sqr(2)
Roz = 5
Piz = -17

'Boxy = 300
'Boxz = 150

Bat_imax = 50
Bat_vmin = 18

'Joystickservoloopflag = 0
'Maxservos = 4
Regulatoremode = 1
Maxnpulse = 7
'Sumpulse = 8000

Ail2rollfactor = 30 / Auto_dx7halfresolution
Ail2rollfactor = 0 - Ail2rollfactor
Ele2pitchfactor = 30 / Auto_dx7halfresolution
Ele2pitchfactor = 0 - Ele2pitchfactor

'Start in failsafe mode (= Flight Mode 1 with Motor off)

Master(dx7throttle) = Auto_min
Master(dx7aileron) = 0
Master(dx7elevator) = 0
Master(dx7rudder) = 0
Master(dx7gear) = Auto_max
Master(dx7flap) = Auto_max
Master(dx7aux2) = Auto_max

Disable Interrupts

#if Ver_board <> 3
'-----
'----- SERVO PULSE GENERATION BY FAST PWM (MODE 14) -----
'-----
,
''(1) Set OCRxy pins as output
''(2) Set TOP (Counter at 20ms) into ICRx (Input Capture Register)
''(3) Set OCRxA, OCRxB, OCRxC to Servox1, Servox2, Servox3
''(4) Set FAST PWM,Mode 14 (Bits WGMx3..1 = 1, WGMx0 = 0) and start timers
,
'The resolution is 2000 for 1ms at 16 MHz, Prescale = 8.
'The resolution is 1843 for 1ms at 14.7456 MHz, Prescale = 8.

'ad 1) Timer1: OC1A, OC1B, OC1C pins are DDRB = &B11100000 (=224)
'      Timer3: OC3A, OC3B, OC3C pins are DDRE = &B00111000 (=56)
'      Timer4: OC4A, OC4B, OC4C pins are DDRH = &B00111000 (=56)
'      Timer5: OC5A, OC5B, OC5C pins are DDRL = &B00111000 (=56)
Ddre = 56
Ddrh = 56
Ddrb = 224
Ddr1 = 56

```

'From 9th autonomous flight

'formerly 30

'formerly 15

'need to be tested

'In PPM times is was 20,

'In PPM times is was 20,

'In PPM times is was 20,

'Roll zero Copilot sensor

'Pitch zero Copilot sensor

'Model specific parameter

'Model specific parameter

'formerly ...30/820

```

'ad 2) ICR1H: 0x87 ICR3H: 0x97 ICR4H: 0xA7 ICR5H: 0x127
'      ICR1L: 0x86 ICR3L: 0x96 ICR4L: 0xA6 ICR5L: 0x126
'      as predefined in M2560.def - checked:Ki
'
'      MAX TCNT Overflow for 14.7456MHz is at 35.55555555 ms
'      TOP for 22 ms is: 40550
'      TOP for 11 ms is: 20275
'      1 ms is: 1843
'
'      In BASCOM:
'      TOP = 40550: ICRxH = High (TOP): ICRxL = Low (TOP)
Icr1h = High(top)
Icr1l = Low(top)
Icr3h = High(top)
Icr3l = Low(top)
Icr4h = High(top)
Icr4l = Low(top)
Icr5h = High(top)
Icr5l = Low(top)

'ad 3) Write output compare start values
Switchboard

'ad 4) TCCRxA: Bit      7      6      5      4      3      2      1      0
'      Name    COMxA1 COMxA0 COMxB1 COMxB0 COMxC1 COMxC0 WGMx1 WGMx0
'      value    1      0      1      0      1      0      1      0
'      means    clear at compare match                    fast PWM
'                  set at Bottom                          mode 14
'      address (for) 0x80 (Timer1)
'                   0x90 (Timer3)
'                   0xA0 (Timer4)
'                   0x120 (Timer5)
'
'      TCCRxB: Bit      7      6      5      4      3      2      1      0
'      Name    ICNCx ICESx  -   WGMx3 WGMx2 CSx2  CSx1  CSx0
'      value    0      0      0      1      1      0      1      0
'      means noise c edge  -   fast PWM  -- prescale 8--
'      address (for) 0x81 (Timer1)
'                   0x91 (Timer3)
'                   0xA1 (Timer4)
'                   0x121 (Timer5)
'
'      In BASCOM:
'      TCCRA = &B10101010 (=170)
'      TCCRB = &B00011010 (=26)
'      TCCRxA = TCCRA (TCCRxA = TCCRxA OR 170)
'      TCCRxB = TCCRB (TCCRxB = TCCRxB OR 26)

'Start PWM
Tccr1a = Tccra
Tccr1b = Tccrb
'waitms 2 may be introduced here, if one ever encounters peak current problems
Tccr3a = Tccra
Tccr3b = Tccrb
'waitms 2 may be introduced here, if one ever encounters peak current problems
Tccr4a = Tccra
Tccr4b = Tccrb
'waitms 2 may be introduced here, if one ever encounters peak current problems
Tccr5a = Tccra
Tccr5b = Tccrb

#else
'-----
'----- SINGLE SLOPE PWM ON XMEGA128A1 -----
'-----

```



```

'pwm initialisation here
'RC cycle period should be ideally at 20ms, viz. 50 Hz.
'With fpwm = fosc/per/clckdiv = 32MHz/(2^16)/8 = 1MHz/2^14 = 61.0352 Hz we have a
'cycle period of 16.384 ms most servos should be able to live with.

'From the Xmega AN on Timer/Counters:
'6.4 Using a Timer/Counter for PWM Generation
'    Task: Configure TCC0 for pulse width modulation output with varying duty cycle on
channel A.
'    1. Configure PC0 for output by setting bit 0 in PORTC.DIR.
'    2. Select the timer period by setting the PER[H:L] register.
'    3. Select a waveform generation mode by setting the WGMODE[2:0] bits in CTRLB
'    4. Enable Compare Channel A by setting the CCAEN bit in CTRLB.
'    5. Start the TC by selecting a clock source (CLKSEL[3:0] in CTRLA).
'    6. Calculate the desired compare value.
'    7. Write the new compare value to CCA[H:L].
'    8. Wait for the TC Overflow Flag to be set. (OVFIF in INTFLAGS).
'    9. Clear the TC Overflow flag.
'    10. Go to step 6.
'Using this sequence, the compare value will be updated once every PWM period.

'The following procedure is a slight modification which writes to the compare
'buffers instead:
'    1. Configure PD0, PD1, PE0, PE1 for output --> PORTD.DIR, PORTE.DIR
Portd_dir = &HFF
Porte_dir = &HFF

'    2. Select the timer period by setting the PER[H:L] register.(16.384 ms at 32 MHz)
Tcd0_perl = &HFF
Tcd0_perh = &HFF
Tcd1_perl = &HFF
Tcd1_perh = &HFF
Tce0_perl = &HFF
Tce0_perh = &HFF
Tce1_perl = &HFF
Tce1_perh = &HFF

'    3. Define WGMODE 011 = single slope by setting the WGMODE[2:0] bits in CTRLB
'    and set OCA-D active by setting the CCAEN bits in CTRLB.
Tcd0_ctrlb = &HF3
Tcd1_ctrlb = &HC3
Tce0_ctrlb = &HF3
Tce1_ctrlb = &HC3

'    4. Write the new compare value to CCxBUF[H:L]
Switchboard

'    5. Start the TC by selecting a clock source (CLKSEL[3:0] in CTRLA.
Tcd0_ctrla = &H04
Timers (Bits 3..0 define clckdiv)
Tcd1_ctrla = &H04
Tce0_ctrla = &H04
Timers (Bits 3..0 define clckdiv)
Tce1_ctrla = &H04

'PWM runs
#endif

Enable Interrupts

Gsmreadyflag = 1

```

```

'##### main loop init #####
'It may or may not be advisable to break down the code into smaller segments
'in subs or function. One big loop and overwritables is however better in terms
'of speed considerations and memory usage.
'#####
Mainloop:
Hby = 0
Loopiby = 0
Jby = 0
Loopkby = 0
Lby = 0
Iwo = 0
Jwo = 0
Kwo = 0
Lwo = 0
Iin = 0
Jin = 0
Kin = 0
Lin = 0
Ilo = 0
Jlo = 0
Klo = 0
Llo = 0
Isi = 0
Jsi = 0
Ksi = 0
Lsi = 0
Loopist = ""
Jst = ""
Kst = ""
Lst = ""
'Disturbdoneflag = 0
'Disturbflag = 0
'Motoroffflag = 0
'#if Ver_board = 1
'Portc.5 = 0
'Portc.4 = 0
'Portc.3 = 1
'#endif

Modelnames = Modelnamee
Print Modelnames
Print "#!Condition is red!"

#if Ver_board <> 3
Start Watchdog
Timer2 = 0
Timer0 = 0
#endif

Rxahiby = 0
Rxbhiby = 0
Rxaloopby = 1
Rxbloopby = 1
Rxaloopflag = 0
Rxbloopflag = 0
'Portj.7 = 1
Nwo = 0

```

```

'=====
'=====
'=====
'===== CORE MAIN LOOP START =====
'=====

```

```

'=====
'=====

'Rxasync = 0

Do
  #if Ver_board <> 3
    Reset Watchdog
  #endif
  Readsatellites
  and servo signal making
  updateslaves
  Readusart0
  calls

  'Watchdog timer is 2s
  'calls switchboard for mixing
  'switchboard calls
  'calls loopistparser which
  'servoadjust (parser)
  'mixeradjust (parser)
  'Printeeram (printer to

AR7212 programmer)

'=====
'==== NOTE THAT EXTENSIONS NEED THE LOOPISTPARSER_EXT TO BE CALLED FROM =====
'==== READUSART0 (INSTEAD OF LOOPISTPARSER) =====
'==== Decoment the following lines for the AR7212 extensions =====
'=====
'If Autonomousservocontrolflag = 0 Then Goto Skipend

'If Calibrationflag = 1 Then Goto Skipend
'If Gsmreadyflag = 0 Then Goto Skipend
'If Justdirectflag = 1 Then Goto Skipend

  If Restartflag = 1 Then Exit Do
GSM connection
  ' Interpretedx7
  ' Readadc
  ' Readgps
Teledata

  Skipend:

Loop Until Restartflag = 1

'=====
'=====
'=====
'===== CORE MAIN LOOP END =====
'=====
'=====

'=====
'===== Decoment the following line for the AR7212 extensions =====
'=====
'Restart
Goto Mainloop
End

'end program

*****
***** Read Satellites *****
*****
Sub Readsatellites()
  #if Ver_board <> 3

```

```

'Goto Notimer2
'2nd method is based on the detection of a 7ms delay
If Timer2 >= 254 Then Timer2 = T7
Do
    Readcharasc = Ischarwaiting(#3)
    If Readcharasc = 1 Then
        Iby = Inkey(#3)
        If Timer2 >= T7 Then
read, start sync
            'Toggle Portj.7
            Rxaloopby = 1
packet
            Rxaloopflag = 0
high/low
            Rxahiby = Iby
            Rxdoneflag = 0
an update unless not set by RXA
        Else
            Rxaloopflag = 1 - Rxaloopflag
byte
            If Rxaloopflag = 1 Then
received
                If Rxaloopby < 8 Then
lowbyte
                    Iwo = Makeint(iby , Rxahiby)
                    Rxainwo(rxaloopby) = Iwo
                    Incr Rxaloopby
                Else
process received
                    'Process receival
                    Rxaloopby = 8
arrays in desync state
                    'should also block reading

2'nd frame
                    Iwo = Makeint(iby , Rxahiby)
                    Rxainwo(rxaloopby) = Iwo
                    Rxastartwo = Rxainwo(1)
info
                    For Jby = 2 To 8
                        Iwo = Rxainwo(jby)
                        Jwo = Iwo And &B0000001111111111
                        Kwo = Iwo And &B0011110000000000
                        Kby = High(kwo)
                        Rotate Kby , Right , 2
                        Incr Kby
array index
                        Rxamaster(kby) = Jwo
dimensioned at 16
                    Next Jby
                    ' final exam

                    If Kby = 4 Then
                        Iby = High(rxastartwo)
1201, 2101 (was nonshielded)
                        Rxquality = Iby And &B00000011
higher is better
                        Rxastatus = Iby And &B00110000
lower is better

                    If Rxquality >= Rxbquality Then
                        If Rxastatus <= Rxbstatus Then
15h in the air by now
                            If Rxdoneflag = 0 Then
                                Rxabflag = 0
                                Rxstarttwo = Rxastarttwo
                                For Iby = 1 To 7
                                    Master(iby) = Rxamaster(iby) - 512

```

```

'stay at 7ms
'intermediately #2 on Crumb2560
'If character in buffer
'intermediately #2 on Crumb2560
'If more than 7 ms since last

'reset loopcounter for new

'reset Loopflag: even/odd means

'store MSB for 1st entry
'clear Rxdoneflag. This allows

'change loopflag for high/low

'if low byte received
'--if less than 8 words

'make word from hbyte and

'store word at index

'--when 8 words received,

'necessary for not bouncing

'should also block reading

'make last word
'store word at index
'take first word as special

'Process data in channels 2-8

'last 10 bits give channel data
'these bits give channel number

'after shifting to the LSB side
'zero not allowed as Bascom

'Rxamaster is sufficiently

'

'rxastarttwo: 0301, 0201, 0101,

'last 2 bits: either 3, 2, 1, 0

'last 2 bits: either 3, 2, 1, 0

'to be improved
'to be improved - but works for

'NEW NEEDS AIR TEST
'im Kasten

```

	Next Iby	
	Switchboard	'calls DX7 switch reading and
mixing operation		
	Rxdoneflag = 1	
	End If	
	End If	
	End If	
	Rxaloopby = 1	'Loop reset for new stream of
data		
	End If	'of Abschlussprüfung
	End If	'of analysis of received frame
	Else	'else high byte was received
	Rxahiby = Iby	'store high byte
	End If	'of low byte received
	End If	'of successive bytes in frame
received		
	Timer2 = 0	'reset timer
	End If	'of any byte received
	Loop Until Readcharasc = 0	'leave loop for other tasks
	Notimer2:	
	'Goto Notimer0	
	If Timer0 >= 254 Then Timer0 = T7	
	Do	
	Readcharasc = Ischarwaiting(#4)	
	If Readcharasc = 1 Then	'If character in buffer
	'If Ischarwaiting(#4) = 1 Then	
	Iby = Inkey(#4)	
	If Timer0 >= T7 Then	'If more than 7 ms since last
read, start sync		
	Rxbloopby = 1	'reset loopcounter for new
packet		
	Rxbloopflag = 0	'reset Loopflag: even/odd means
high/low		
	Rxbhiby = Iby	'store MSB for 1st entry
	Rxdoneflag = 0	'This allows an update by RXB
if RXA		
	Else	
	Rxbloopflag = 1 - Rxbloopflag	'change loopflag for high/low
byte		
	If Rxbloopflag = 1 Then	'if low byte received
	If Rxbloopby < 8 Then	'--if less than 8 words
received		
	Iwo = Makeint(iby , Rxbhiby)	'make word from hibernate and
lowbyte		
	Rxbino(rxbloopby) = Iwo	'store word at index
	Incr Rxbloopby	
	Else	'--when 8 words received,
process received		
	'Process receive	
	Rxbloopby = 8	'necessary for not bouncing
arrays in desync state		
		'should also block reading
2nd frame		
	Iwo = Makeint(iby , Rxbhiby)	'make last word
	Rxbino(rxbloopby) = Iwo	'store word at index
	Rxbstartwo = Rxbino(1)	'take first word as special
info		
	For Jby = 2 To 8	'Process data in channels 2-8
	Iwo = Rxbino(jby)	
	Jwo = Iwo And &B0000001111111111	'last 10 bits give channel data
	Kwo = Iwo And &B0011110000000000	'these bits give channel number
	Kby = High(kwo)	
	Rotate Kby , Right , 2	'after shifting to the LSB side
	Incr Kby	'zero not allowed as Bascom
array index		

```

                                Rxbmaster(kby) = Jwo                                'Rxbmaster is sufficiently
dimensioned at 16
                                Next Jby
                                'Final exam
                                If Kby = 4 Then
                                    Iby = High(rxbstartwo)
                                    Rxbquality = Iby And &B00000011                                'last 2 bits: either 3, 2, 1, 0
                                    Rxbstatus = Iby And &B00110000
                                    If Rxbquality => Rxaquality Then
                                        If Rxbstatus <= Rxastatus Then
                                            If Rxdoneflag = 0 Then
                                                Rxabflag = 1                                'frei
                                                Rxstartwo = Rxbstartwo
                                                For Iby = 1 To 7
                                                    Master(iby) = Rxbmaster(iby) - 512
                                                Next Iby
                                                Switchboard                                'calls DX7 switch reading and
mixing operations
                                            Rxdoneflag = 1
                                            End If
                                        End If
                                    End If
                                    Rxbloopby = 1                                'Loop reset for new stream of
data
                                End If
                                End If
                                Else
                                    Rxbhiby = Iby                                'high byte is received
                                End If                                'store MSB
                                End If
                                Timer0 = 0                                'reset timer
                                End If
                                Loop Until Readcharasc = 0                                'leave loop for other tasks

                                Notimer0:
                                #endif

End Sub

```

```

'*****
'***** Read USART0 *****
'*****

Sub Readusart0()
    #if Ver_board <> 3
        Do
            control from AR7212-programmer/ground station
            'get a char from the UART
            Readcharasc = Ischarwaiting()
            If Readcharasc <> 0 Then
                Readcharasc = Inkey()
                chr(0) is possible
                Readchar = Chr(readcharasc)
                If Readcharasc = 13 Then
                    If Loopist <> "" Then
                        Loopistparser
                    FOR LOOPISTPARSER_EXT FOR EXTENSIONS
                        End If
                    Else
                        ' (viz. not CHR 13)
                        If Loopiby < Cmaxchar Then
                            Incr Loopiby
                            'DONT USE LoopIby, LoopIst FOR
                            ANYTHING ELSE IN THE MAINLOOP AND ASSOC. ROUTINES!
                            Loopist = Loopist + Readchar
                            'LoopIst holds string to be
                            analyzed
                        Else

```

```

        Print "Error: String too long!"
    End If
End If
End If
Loop Until Readcharasc = 0 Or Restartflag = 1
#else
Do
control from AR7212-programmer/ground station
'get a char from the UART
'Readcharasc = Inkey (#1)
'If Readcharasc <> 0 Then
    Readchar = Chr(readcharasc)
    If Readcharasc = 13 Then
        If Loopist <> "" Then
            Loopistparser
FOR LOOPISTPARSER_EXT FOR EXTENSIONS
            End If
        Else
            If Loopiby < Cmaxchar Then
                Incr Loopiby
ANYTHING ELSE IN THE MAINLOOP AND ASSOC. ROUTINES!
                Loopist = Loopist + Readchar
analyzed
            Else
                Print "Error: String too long!"
            End If
        End If
    End If
'End If
Loop Until Readcharasc = 0 Or Restartflag = 1

#endif
End Sub

```

```

' (for IF Chr(13))
' (for IF Readcharasc <> 0)

'Inner Loop 1: Check for

'was there a char?
'a message has been received
'if not empty it_
'needs processing    REPLACE

' (viz. not CHR 13)
'DONT USE LoopIby, LoopIst FOR
'LoopIst holds string to be

' (for IF Chr(13))
' (for IF Readcharasc <> 0)

```

```

'*****
'***** Switchboard *****
'*****
Sub Switchboard()
    'Timer0 = 0
    'only for testing performance
    'set Satellite2 to Goto

Notimer0
    'Print "OK"
    '=====
    '===== Read DX7 switches and determine flight mode =====
    '=====

    Incr Jtimer3
    22 ms, reset by GPS string receival
    'JTimer3 is incremented every

    If Master(dx7aux2) >= 0 Then
        autonomous flight modes
        'Aux 2 is the swich to enable
        If Rxabflag = 0 Then Toggle Mainled
        If Autonomousservocontrolflag = 1 Then
            Print "auto off"
        End If
        Autonomousservocontrolflag = 0
    Else
        If Rxabflag = 0 Then Reset Mainled
        'Mainled on if auto is on
        If Autonomousservocontrolflag = 0 Then
            Print "auto on"
        End If
        Autonomousservocontrolflag = 1
        Master(dx7throttle) = Auto_throttle
        'These 4 values are going to be
    computed

```

```

Master(dx7aileron) = Auto_aileron           'in autonomous flight modes
(extensions)
Master(dx7elevator) = Auto_elevator
Master(dx7rudder) = Auto_rudder
End If
If Master(dx7flap) >= 172 Then               'Master ranges from -384 to
+384                                         'Master (6) = 3 way switch: up,
down, middle
Elseif Master(dx7flap) < -171 Then
    Fltmode = 3
Else
    Fltmode = 2                             'middle
End If
Fltmode = Fltmode * 2                       'converts Fltmode 1-3 to 2,4,6
If Master(dx7gear) >= 0 Then                'Gear switch off: Flightmodes =
2, 4, 6                                     'Gear switch on: Flightmodes =
1, 3, 5
End If

'=====
'===== set mixer data according to flight mode =====
'=====
If Fltmode <> Oldfltmode Then                'If Fltmode has changed
    Print "flight mode " ; Fltmode
    Select Case Fltmode                    'set flight mode and read mixer
data from eeram
        Case 1
            For Iby = 1 To 72              'read corresponding mixer table
from eeram
                Mixdat(iby) = Mixdatm1(iby)
                Mixtyp(iby) = Mixtypm1(iby)
            Next Iby
            For Iby = 1 To 12
                Offsets(iby) = Offsetm1(iby)    'read corresponding table of
Offsets from eeram
                Delays(iby) = Delaym1(iby)      'read corresponding table of
Delays from eeram
            Next Iby
            'Iby = Memcopy(MixDatm1(1) , MixDat(1) , 72)    'doesn't work so
far, don't know why
            'Iby = Memcopy(offsetm1(1) , Offset(1) , 24)

        Case 2
            For Iby = 1 To 72              'read corresponding mixer table
from eeram
                Mixdat(iby) = Mixdatm2(iby)
                Mixtyp(iby) = Mixtypm2(iby)
            Next Iby
            For Iby = 1 To 12
                Offsets(iby) = Offsetm2(iby)    'read corresponding table of
Offsets from eeram
                Delays(iby) = Delaym2(iby)      'read corresponding table of
Delays from eeram
            Next Iby

        Case 3
            For Iby = 1 To 72              'read corresponding mixer table
from eeram
                Mixdat(iby) = Mixdatm3(iby)
                Mixtyp(iby) = Mixtypm3(iby)
            Next Iby
            For Iby = 1 To 12
                Offsets(iby) = Offsetm3(iby)    'read corresponding table of
Offsets from eeram

```



```

        Delays(iby) = Delaym3(iby)           'read corresponding table of
Delays from eeram
        Next Iby

    Case 4
        For Iby = 1 To 72                   'read corresponding mixer table
from eeram
            Mixdat(iby) = Mixdatm4(iby)
            Mixtyp(iby) = Mixtypm4(iby)
            Next Iby
            For Iby = 1 To 12
                Offsets(iby) = Offsetm4(iby)   'read corresponding table of
Offsets from eeram
                Delays(iby) = Delaym4(iby)     'read corresponding table of
Delays from eeram
            Next Iby

    Case 5
        For Iby = 1 To 72                   'read corresponding mixer table
from eeram
            Mixdat(iby) = Mixdatm5(iby)
            Mixtyp(iby) = Mixtypm5(iby)
            Next Iby
            For Iby = 1 To 12
                Offsets(iby) = Offsetm5(iby)   'read corresponding table of
Offsets from eeram
                Delays(iby) = Delaym5(iby)     'read corresponding table of
Delays from eeram
            Next Iby

    Case 6
        For Iby = 1 To 72                   'read corresponding mixer table
from eeram
            Mixdat(iby) = Mixdatm6(iby)
            Mixtyp(iby) = Mixtypm6(iby)
            Next Iby
            For Iby = 1 To 12
                Offsets(iby) = Offsetm6(iby)   'read corresponding table of
Offsets from eeram
                Delays(iby) = Delaym6(iby)     'read corresponding table of
Delays from eeram
            Next Iby

    Case Else
        End Select
    End If
    Oldfltmode = Fltmode                     'Fltmode is now set

    '=====
    '===== Perform mixing to arrive at simulated stick =====
    '===== settings for each servo (slave) channel =====
    '=====

    'Mixing starts here, results

    (pwm values in Slave array
    For Iby = 1 To 12
        Slave(iby) = 0
    Next Iby
    Jby = 1
    Iby = 0
    For Kby = 1 To 72
        become a table
        Incr Iby
        If Iby = 13 Then
            column counter 1
            Iby = 1
            Incr Jby
        End If

```

```

range from 43 to 213 (to allow bitwise storage of tables)
from 64 to 192
    If Mixdat(kby) = 128 Then
        'nop
    Else
        Iin = Mixdat(kby)
        Iin = Iin - 128
        Jin = Master(jby)
from DX7 radio (-384 to 384)

        Select Case Mixtyp(kby)
            Case 0
                Case 1
                    If Jin < 0 Then Jin = 0
                Case 2
                    If Jin >= 0 Then Jin = 0
                Case 3
                    If Jin < 0 Then Jin = 0
                Case 4
                    If Jin >= 0 Then Jin = 0
                Case Else

            End Select

            Iin = Iin * Jin
384 = -24568 to +24568
            Shift Iin , Right , 6 , Signed
from -384 to +384

            Slave(iby) = Slave(iby) + Iin
Sum of Mastercontribution(J) * Mixer(I,J)
            End If
        Next Kby

        '=====
        '===== Keep simulated stick setting inside valid range, =====
        '===== multiply by up/downtravel factor, add center and =====
        '===== offset values to arrive at servo pwm values. =====
        '===== Finally, keep servo values inside valid range =====
        '===== and update accordingly =====
        '=====

        For Iby = 1 To 12
            Iin = Slave(iby)
simulated stick values for each servo

            If Iin >= Auto_max Then Iin = Auto_max
inside valid range
            If Iin < Auto_min Then Iin = Auto_min
            If Utravels(iby) >= Mixer_max Then Utravels(iby) = Mixer_max
            If Dtravels(iby) >= Mixer_max Then Dtravels(iby) = Mixer_max
to be shifted, redundant here!

            If Iin = 0 Then
                'nop
            ElseIf Iin > 0 Then
                Iin = Iin * Utravels(iby)
                Shift Iin , Right , 5 , Signed
                Iin = Iin * Dtravels(iby)

```

'Dx7 mixer values maximally

'The usual range (=100%) goes

'no mixing, do nothing

'mixer (slope) values
'converted into integer numbers
'gives range from -64 to +64
'holds the input data received

'Analyze for mixer type
'linear mixer

'J mixer

'F mixer

'L mixer

'T mixer

'gives usual range from -64 *

'fast divide by 64 gives range

'mixing is done by Slave(I) =

'slave array so far contains

'keep simulated stick values

' (=85) make

' ****Ki: needs

'multiply by up/downtravel

'fast divide by 32 gives range

```

        Shift Iin , Right , 5 , Signed
    End If
    If Delays(iby) > 0 Then
        'If
        'End
    End If
    Slave(iby) = Centers(iby) + Iin
    Slave(iby) = Slave(iby) + Offsets(iby)
    If Slave(iby) >= Ultramaxs(iby) Then
        'Adding Centers (= 3000)_
        'and offsets_
        'gives then final PWM
        pulselenghts to be corrected_
        Slave(iby) = Ultramaxs(iby)
    ElseIf Slave(iby) < Dtramaxs(iby) Then
        'if outside travel range
        Slave(iby) = Dtramaxs(iby)
    End If
Next Iby

    If Dx7printfld >= 1 Then
        'Option to print out HEX Values
        of frames as originally_
        Decr Dx7printfld
        Print "DX7a:";
        'received from DX7 satellites
        For Iby = 1 To 8
            Print " " ; Hex(rxainwo(iby));
        Next Iby
        Print ""
        Print "DX7b:";
        For Iby = 1 To 8
            Print " " ; Hex(rxbinwo(iby));
        Next Iby
        Print ""
    Else
    End If
    Updateslaves
        'update servo positions

End Sub

```

```

'*****
'***** Updateslaves *****
'*****
Sub Updateslaves()

#If Ver_board <> 3

    Ocr1ah = High(slave(ser_oc1a))
    Ocr1al = Low(slave(ser_oc1a))
    Ocr1bh = High(slave(ser_oc1b))
    Ocr1bl = Low(slave(ser_oc1b))
    Ocr1ch = High(slave(ser_oc1c))
    Ocr1cl = Low(slave(ser_oc1c))

    Ocr3ah = High(slave(ser_oc3a))
    Ocr3al = Low(slave(ser_oc3a))
    Ocr3bh = High(slave(ser_oc3b))
    Ocr3bl = Low(slave(ser_oc3b))
    Ocr3ch = High(slave(ser_oc3c))
    Ocr3cl = Low(slave(ser_oc3c))

    Ocr4ah = High(slave(ser_oc4a))
    Ocr4al = Low(slave(ser_oc4a))
    Ocr4bh = High(slave(ser_oc4b))
    Ocr4bl = Low(slave(ser_oc4b))
    Ocr4ch = High(slave(ser_oc4c))
    Ocr4cl = Low(slave(ser_oc4c))

    Ocr5ah = High(slave(ser_oc5a))

```

```

Ocr5a1 = Low(slave(ser_oc5a))
Ocr5bh = High(slave(ser_oc5b))
Ocr5b1 = Low(slave(ser_oc5b))
Ocr5ch = High(slave(ser_oc5c))
Ocr5c1 = Low(slave(ser_oc5c))

#else
'more elegantly in the future by WIO

Tcd0_ccabuf1 = Low(slave(ser_tcd0a))
Tcd0_ccabufh = High(slave(ser_tcd0a))
Tcd0_ccbbuf1 = Low(slave(ser_tcd0b))
Tcd0_ccbbufh = High(slave(ser_tcd0b))
Tcd0_cccbuf1 = Low(slave(ser_tcd0c))
Tcd0_cccbufh = High(slave(ser_tcd0c))
Tcd0_ccdbuf1 = Low(slave(ser_tcd0d))
Tcd0_ccdbufh = High(slave(ser_tcd0d))

Tcd1_ccabuf1 = Low(slave(ser_tcd1a))
Tcd1_ccabufh = High(slave(ser_tcd1a))
Tcd1_ccbbuf1 = Low(slave(ser_tcd1b))
Tcd1_ccbbufh = High(slave(ser_tcd1b))

Tce0_ccabuf1 = Low(slave(ser_tce0a))
Tce0_ccabufh = High(slave(ser_tce0a))
Tce0_ccbbuf1 = Low(slave(ser_tce0b))
Tce0_ccbbufh = High(slave(ser_tce0b))
Tce0_cccbuf1 = Low(slave(ser_tce0c))
Tce0_cccbufh = High(slave(ser_tce0c))
Tce0_ccdbuf1 = Low(slave(ser_tce0d))
Tce0_ccdbufh = High(slave(ser_tce0d))

Tce1_ccabuf1 = Low(slave(ser_tce1a))
Tce1_ccabufh = High(slave(ser_tce1a))
Tce1_ccbbuf1 = Low(slave(ser_tce1b))
Tce1_ccbbufh = High(slave(ser_tce1b))

#endif
End Sub

```

```

'*****
'***** LOOPIST MESSAGE PARSER *****
'*****

Sub Loopistparser()
    Select Case Loopist
latency
        Case "alive?"
implement a "heartbeat counter"
missing alive?-yes-cycles. Aim:
ground is missing.

        Print "yes"
        Gsmreadyflag = 1
        Alivetime = Ggatetime

        Case "reset_to_upload_firmware"
            Do
                Loop
in 2 seconds

the reset vector

'used for feedback control and
'estimation. Planned to
'based on the number of
'Return HOME if heartbeat from

'watchdog will reset the AR7212
'Better is a direct jump to

```

	Case Else	'prepare for parameter parsing
	For Iby = 1 To 20	'clear the old array
	Istarr(iby) = ""	'should be done in a better way
	Next Iby	
	Iin = 1	
	Lin = Len(loopist)	
	If Lin >= Cmaxchar Then Lin = Cmaxchar	
	Kst = ""	
	Kwo = 0	
	For Jin = 1 To Lin	'go through the string
	Jst = Mid(loopist , Jin , 1)	
	If Jst <> " " Then	'if no separator
	Kst = Kst + Jst	
	Kwo = 1	'add char and set flag that
item is valid		
	Else	
	If Kwo = 1 Then	'if item is valid
	Istarr(iin) = Kst	'keep in list,
to be checked		'*Compiler concern*: Len Istarr
	Incr Iin	'increment list counter,
	Kst = ""	'start new item,
	Kwo = 0	'and reset validity flag after
one blank		
	End If	'this will allow to enter two
blanks between		
	End If	'items without problem
	Next Jin	
		'*Compiler concern*: Len Istarr
to be checked		
	Istarr(iin) = Kst	
	Jst = Istarr(1)	
	If Len(jst) >= 4 Then Jst = Left(jst , 3)	
	Select Case Jst	
	'-----	
	'-----[AR7212 COMMANDS]-----	
	'-----	
	Case "ar7"	'ar7
	Print "OK"	'just For Test
	Case "DX7"	'DX7
frames in HEX	Dx7printfldg = 1	'leads to printing of received
	Case "pil"	'pilot name
	Pilotnames = Istarr(2)	
	Pilotnamee = Pilotnames	
	Print "OK"	
	Case "mod"	'model name
	Modelnames = Istarr(2)	
	Modelnamee = Modelnames	
	Print "OK"	
	Case "flm"	'flight mode
	Iin = Val(istarr(2))	
	Ist = Istarr(3)	
	Fltmodenames(iin) = Ist	
	Fltmodenamee(iin) = Ist	
	Print "OK"	
	Case "ser"	'servo adjust
	Servoadjust	

```

        Case "mix"
            Mixeradjust
        'mixer adjust

        Case "ram"
            Printeeram
        'write eeram

        Case Else
            End Select
        'of Loopist cases
    End Select
    Loopiby = 0
    Loopist = ""

End Sub

'*****
'*****SubMenu SERVO ADJUST *****
'*****
Sub Servoadjust()
    'syntax:
    'ser 3 cen 2745          means set servo 3 center to 2745
    'ser 3 nhi 60            normal uptravel
    'ser 3 nlo 4 40         normal downtravel
    'ser 3 mhi 3650         max uptravel
    'ser 3 mlo 1900         max downtravel
    'mix 2 3 8 -600        offset for fltmode 2, servo 3 at -600
    'mix 2 4 4 52 20       fltmode 2: mix table entry 51 (master 4, slave 3) to
20

    Ist = Istarr(3)
    If Len(ist) >= 4 Then Ist = Left(ist , 3)

    Iin = Val(istarr(2))
    Jin = Val(istarr(4))
    Kst = Istarr(4)
    If Len(kst) >= 5 Then Kst = Left(kst , 4)

    Select Case Ist

        Case "set"
            Slave(iin) = Jin
            Updateslaves

        Case "nam"
            Servonames(iin) = Kst
            Servonamee(iin) = Kst

        Case "cen"
            Centers(iin) = Jin
            Centere(iin) = Jin

        Case "nhi"
            Utravels(iin) = Jin
            Utravele(iin) = Jin

        Case "nlo"
            Dtravels(iin) = Jin
            Dtravele(iin) = Jin

        Case "mhi"
            Utramaxes(iin) = Jin
            Utramaxe(iin) = Jin

        Case "mlo"
            Dtramaxes(iin) = Jin
            Dtramaxe(iin) = Jin

```

```

        Case Else

End Select
Print "OK"

End Sub

'*****
'*****SubMenu MIXER ADJUST *****
'*****
Sub Mixeradjust()

    Iin = Val(istarr(2))           'FltMode
    Jin = Val(istarr(3))           'Servo
    Kin = Val(istarr(4))           'Channel
    Lin = Val(istarr(5))           'Value

    Lst = Istarr(5)
    Lby = Len(lst)
    Lby = Lby - 1
    Kst = Left(lst , 1)           'mixer type

    Select Case Kst
        Case "J"                   'Curve stays at center output
        below center input and increases above center input
            Mixtype = 1
            Lst = Right(lst , Lby)
            Lin = Val(lst)

        Case "F"                   'Curve increases below center
        input and stays at center output above center input
            Mixtype = 2
            Lst = Right(lst , Lby)
            Lin = Val(lst)

        Case "T"                   'Curve stays at center output
        below center input and decreases above center input
            Mixtype = 3
            Lst = Right(lst , Lby)
            Lin = Val(lst)
            Lin = 0 - Lin

        Case "L"                   'Curve decreases below center
        input and stays at center output above center input
            Mixtype = 4
            Lst = Right(lst , Lby)
            Lin = Val(lst)
            Lin = 0 - Lin

        Case Else
            Mixtype = 0
    End Select

    Select Case Iin                'Fltmode dependant offsets,
    delays, mixer value and type of mixer

        Case 1
            If Kin = 7 Then
                Offsetm1(jin) = Lin
            ElseIf Kin = 8 Then
                Delaym1(jin) = Lin
            Else
                Kin = Kin - 1
                Iin = Kin * 12
                Iin = Iin + Jin
                Lin = Lin + 128
                Lby = Lin

```

```

        Mixdatm1(iin) = Lby
        Mixtypm1(iin) = Mixtype
    End If
Case 2
    If Kin = 7 Then
        Offsetm2(jin) = Lin
    ElseIf Kin = 8 Then
        Delaym2(jin) = Lin
    Else
        Kin = Kin - 1
        Iin = Kin * 12
        Iin = Iin + Jin
        Lin = Lin + 128
        Lby = Lin
        Mixdatm2(iin) = Lby
        Mixtypm2(iin) = Mixtype
    End If
Case 3
    If Kin = 7 Then
        Offsetm3(jin) = Lin
    ElseIf Kin = 8 Then
        Delaym3(jin) = Lin
    Else
        Kin = Kin - 1
        Iin = Kin * 12
        Iin = Iin + Jin
        Lin = Lin + 128
        Lby = Lin
        Mixdatm3(iin) = Lby
        Mixtypm3(iin) = Mixtype
    End If
Case 4
    If Kin = 7 Then
        Offsetm4(jin) = Lin
    ElseIf Kin = 8 Then
        Delaym4(jin) = Lin
    Else
        Kin = Kin - 1
        Iin = Kin * 12
        Iin = Iin + Jin
        Lin = Lin + 128
        Lby = Lin
        Mixdatm4(iin) = Lby
        Mixtypm4(iin) = Mixtype
    End If
Case 5
    If Kin = 7 Then
        Offsetm5(jin) = Lin
    ElseIf Kin = 8 Then
        Delaym5(jin) = Lin
    Else
        Kin = Kin - 1
        Iin = Kin * 12
        Iin = Iin + Jin
        Lin = Lin + 128
        Lby = Lin
        Mixdatm5(iin) = Lby
        Mixtypm5(iin) = Mixtype
    End If
Case 6
    If Kin = 7 Then
        Offsetm6(jin) = Lin
    ElseIf Kin = 8 Then
        Delaym6(jin) = Lin
    Else
        Kin = Kin - 1
        Iin = Kin * 12

```



```

        Iin = Iin + Jin
        Lin = Lin + 128
        Lby = Lin
        Mixdatm6(iin) = Lby
        Mixtypm6(iin) = Mixtype
    End If
Case Else
    Print "ERROR: Mixtype not recognized"
End Select
Print "OK"

End Sub

'*****
'*****SubMenu Print EERAM *****
'*****
Sub Printeeram()
    Stop Watchdog
    Ist = Pilotnamee
    Print "pil " ; Ist
    Ist = Modelnamee
    Print "mod " ; Ist

    For Iin = 1 To 6
        Ist = Fltmodenamee(iin)
        Print "flm " ; Str(iin) ; " " ; Ist
    Next Iin

    For Iin = 1 To 12
        Ist = Servonamee(iin)
        Print "ser " ; Str(iin) ; " nam " ; Ist
        Jin = Centere(iin)
        Print "ser " ; Str(iin) ; " cen " ; Str(jin)
        Jin = Utravele(iin)
        Print "ser " ; Str(iin) ; " nhi " ; Str(jin)
        Jin = Dtravele(iin)
        Print "ser " ; Str(iin) ; " nlo " ; Str(jin)
        Jin = Utramaxe(iin)
        Print "ser " ; Str(iin) ; " mhi " ; Str(jin)
        Jin = Dtramaxe(iin)
        Print "ser " ; Str(iin) ; " mlo " ; Str(jin)
    Next Iin

    For Iin = 1 To 6
        For Jin = 1 To 12
            For Kin = 1 To 8
                Lin = Kin * 12
                Lin = Lin - 12
                Lin = Lin + Jin
                Lby = Lin
                Select Case Iin
                    Case 1
                        If Kin = 7 Then
                            Lin = Offsetm1(jin)
                            Print "mix " ; Str(iin) ; " " ; Str(jin) ; " " ; Str(kin) ;
" " ; Str(lin)
                        ElseIf Kin = 8 Then
                            Lin = Delaym1(jin)
                            Print "mix " ; Str(iin) ; " " ; Str(jin) ; " " ; Str(kin) ;
" " ; Str(lin)
                        Else
                            Iby = Mixdatm1(lby)
                            Mixtype = Mixtypm1(lby)
                            Lin = Iby

```

```

        Lin = Lin - 128
        Nin = Abs(lin)
        Ist = Str(nin)
        Ist = Trim(ist)
        Lst = Str(lin)
        Lst = Trim(lst)
        Select Case Mixtype
            Case 0
            Case 1
                Lst = "J" + Ist
            Case 2
                Lst = "F" + Ist
            Case 3
                Lst = "T" + Ist
            Case 4
                Lst = "L" + Ist
            Case Else
        End Select
        Print "mix " ; Str(iin) ; " " ; Str(jin) ; " " ; Str(kin) ;
" " ; Lst
    End If

Case 2
If Kin = 7 Then
    Lin = Offsetm2(jin)
    Print "mix " ; Str(iin) ; " " ; Str(jin) ; " " ; Str(kin) ;
" " ; Str(lin)

Elseif Kin = 8 Then
    Lin = Delaym2(jin)
    Print "mix " ; Str(iin) ; " " ; Str(jin) ; " " ; Str(kin) ;
" " ; Str(lin)

Else
    Iby = Mixdatm2(lby)
    Mixtype = Mixtypm2(lby)
    Lin = Iby
    Lin = Lin - 128
    Nin = Abs(lin)
    Ist = Str(nin)
    Ist = Trim(ist)
    Lst = Str(lin)
    Lst = Trim(lst)
    Select Case Mixtype
        Case 0
        Case 1
            Lst = "J" + Ist
        Case 2
            Lst = "F" + Ist
        Case 3
            Lst = "T" + Ist
        Case 4
            Lst = "L" + Ist
        Case Else
    End Select
    Print "mix " ; Str(iin) ; " " ; Str(jin) ; " " ; Str(kin) ;
" " ; Lst
End If

Case 3
If Kin = 7 Then
    Lin = Offsetm3(jin)
    Print "mix " ; Str(iin) ; " " ; Str(jin) ; " " ; Str(kin) ;
" " ; Str(lin)

Elseif Kin = 8 Then
    Lin = Delaym3(jin)
    Print "mix " ; Str(iin) ; " " ; Str(jin) ; " " ; Str(kin) ;
" " ; Str(lin)

Else

```

```

        Iby = Mixdatm3(lby)
        Mixtype = Mixtypm3(lby)
        Lin = Iby
        Lin = Lin - 128
        Nin = Abs(lin)
        Ist = Str(nin)
        Ist = Trim(ist)
        Lst = Str(lin)
        Lst = Trim(lst)
        Select Case Mixtype
            Case 0
            Case 1
                Lst = "J" + Ist
            Case 2
                Lst = "F" + Ist
            Case 3
                Lst = "T" + Ist
            Case 4
                Lst = "L" + Ist
            Case Else
        End Select
        Print "mix " ; Str(iin) ; " " ; Str(jin) ; " " ; Str(kin) ;
" " ; Lst
    End If

Case 4
If Kin = 7 Then
    Lin = Offsetm4(jin)
    Print "mix " ; Str(iin) ; " " ; Str(jin) ; " " ; Str(kin) ;
" " ; Str(lin)

Elseif Kin = 8 Then
    Lin = Delaym4(jin)
    Print "mix " ; Str(iin) ; " " ; Str(jin) ; " " ; Str(kin) ;
" " ; Str(lin)

Else
    Iby = Mixdatm4(lby)
    Mixtype = Mixtypm4(lby)
    Lin = Iby
    Lin = Lin - 128
    Nin = Abs(lin)
    Ist = Str(nin)
    Ist = Trim(ist)
    Lst = Str(lin)
    Lst = Trim(lst)
    Select Case Mixtype
        Case 0
        Case 1
            Lst = "J" + Ist
        Case 2
            Lst = "F" + Ist
        Case 3
            Lst = "T" + Ist
        Case 4
            Lst = "L" + Ist
        Case Else
    End Select
    Print "mix " ; Str(iin) ; " " ; Str(jin) ; " " ; Str(kin) ;
" " ; Lst
    End If

Case 5
If Kin = 7 Then
    Lin = Offsetm5(jin)
    Print "mix " ; Str(iin) ; " " ; Str(jin) ; " " ; Str(kin) ;
" " ; Str(lin)

Elseif Kin = 8 Then
    Lin = Delaym5(jin)

```

```

        Print "mix " ; Str(iin) ; " " ; Str(jin) ; " " ; Str(kin) ;
" " ; Str(lin)
Else
    Iby = Mixdatm5(lby)
    Mixtype = Mixtypm5(lby)
    Lin = Iby
    Lin = Lin - 128
    Nin = Abs(lin)
    Ist = Str(nin)
    Ist = Trim(ist)
    Lst = Str(lin)
    Lst = Trim(lst)
    Select Case Mixtype
        Case 0
        Case 1
            Lst = "J" + Ist
        Case 2
            Lst = "F" + Ist
        Case 3
            Lst = "T" + Ist
        Case 4
            Lst = "L" + Ist
        Case Else
    End Select
    Print "mix " ; Str(iin) ; " " ; Str(jin) ; " " ; Str(kin) ;
" " ; Lst
End If

Case 6
If Kin = 7 Then
    Lin = Offsetm6(jin)
    Print "mix " ; Str(iin) ; " " ; Str(jin) ; " " ; Str(kin) ;
" " ; Str(lin)
Elseif Kin = 8 Then
    Lin = Delaym6(jin)
    Print "mix " ; Str(iin) ; " " ; Str(jin) ; " " ; Str(kin) ;
" " ; Str(lin)
Else
    Iby = Mixdatm6(lby)
    Mixtype = Mixtypm6(lby)
    Lin = Iby
    Lin = Lin - 128
    Nin = Abs(lin)
    Ist = Str(nin)
    Ist = Trim(ist)
    Lst = Str(lin)
    Lst = Trim(lst)
    Select Case Mixtype
        Case 0
        Case 1
            Lst = "J" + Ist
        Case 2
            Lst = "F" + Ist
        Case 3
            Lst = "T" + Ist
        Case 4
            Lst = "L" + Ist
        Case Else
    End Select
    Print "mix " ; Str(iin) ; " " ; Str(jin) ; " " ; Str(kin) ;
" " ; Lst
End If
Case Else
End Select

Next Kin
Next Jin

```

End Sub

1