C165UTAH, C165H, C161U
SAF C165UTAH-LF Version 1.3
SAF C165H-LF Version 1.3
SAF C161U-LF Version 1.3
ASC Autobaud Detection

Wired
Communications

Never stop thinking.

SAF C165UTAH-LF
SAF C165H-LF
SAF C161U-LF
Revision History: 2001-07-27 DS1

Previous Version:

Page	Subjects (major changes since last revision)

For questions on technology, delivery and prices please contact the Infineon Technologies Offices in Germany or the Infineon Technologies Companies and Representatives worldwide: see our webpage at http://www.infineon.com

Table of Contents Page

Preface

This document is intended to introduce the reader into the usage of the Autobaud Detection capability which has been integrated into the Asynchronous Serial Interface (ASC) of the C165UTAH, C165H, C161U product family.

This Application Note provides an example of how to use the baud rate detection feature as it would be needed e.g. in a modem application. Special care is taken concerning the determination of the detected baud rate and restarting the detection.

1 Basic setup of the Autobaud Detection

The autobaud detection unit in the ASC provides a capability to recognize the mode and the baud rate of an asynchronous input signal at RXD. The device is capable of calculating baudrates in the range of 1200 up to 230.400 baud and the corresponding serial modes 7E1, 7O1, 8N1, 8E1 or 8O1 when receiving the string “at” or “AT”, which is used as ATtention command in the common Hayes modem control set.

Internally the clock which is generated by the ASC fractional divider (FDV) is used by the autobaud detection unit as time base. This implies that the FDV has to be initialized within the software before start of the detection and that the range of detectable baudrates is based on this basic frequency[1)].

[1)] The application note “ASC Serial Interface, Async. Baud rate Calculation with the Fractional Divider“ (AP3227) provides additional information concerning the baud rate calculation.

Basic setup of the Autobaud Detection

The following table describes the register initialization concerning baud rate generation for the recognition of the standard baudrates between 1200 and 230.400 baud at a operating frequency of 36 MHz. For different settings, please consult the datasheet or use the Digital Application virtual Engineer - DAvE - who will assist you in determining the corresponding settings for different applications.

Table 1 Baudrate Register Initialization

Register	Value	Description
S0FDV	0x009D	The frequency divider is initialized with the decimal value 157, which sets the basic ASC sample frequency to a value of 11.0391 MHz.
S0CON	0x0827	FDE = 1b > enable frequency divider (FDV) REN = 1b > enable ASC receiver M = 111b > set all mode bits (will be cleared on successful detection)

After successful detection of baudrate and asynchronous operating mode the bits in the S0CON register and the reload-value S0BG of the baudrate timer are set to the appropriate values.

The handling of all involved command bits and the sequence of initialization are shown in the attached sample code. For hardware implementation details, have a look at the section “Autobaud Detection” within the chapter “Asynchronous/Synchronous Serial Interface” of the corresponding datasheet.

2 Determining the detected Baudrate

In modem applications it is necessary to restart the baudrate detection after each single AT command, since the baudrate is allowed to change between two commands. Nevertheless the modem has to be able to send information to the host in this state, which requires that the last used baud rate has to be restored.

After successful recognition of baud rate and asynchronous operating mode of the RXD data input signal, bits in the S0CON register and the value of the S0BG register in the baud rate timer are set to the appropriate values, and the ASC can start immediately with the reception of serial input data.

However this implies that the baud rate generation timer starts to run immediately and the detected baud rate cannot be determined in software by reading the reload value out of the S0BG register.

This application note shows a way how to detect the baud rate by using the C156UTAH, C165H or C161U external interrupt and PEC capability in combination with the autobaud feature of the ASC module.

The following figure shows the sequence of events before and during the detection process. The single points of interest are described in the subsequent paragraph. This example makes use of the general purpose timer T3 and sets it up for a operating speed of 9 MHz - assuming a operating speed of 36 MHz on the evaluation system (for functional details consult the corresponding datasheet chapter “General Purpose Timer Unit“).

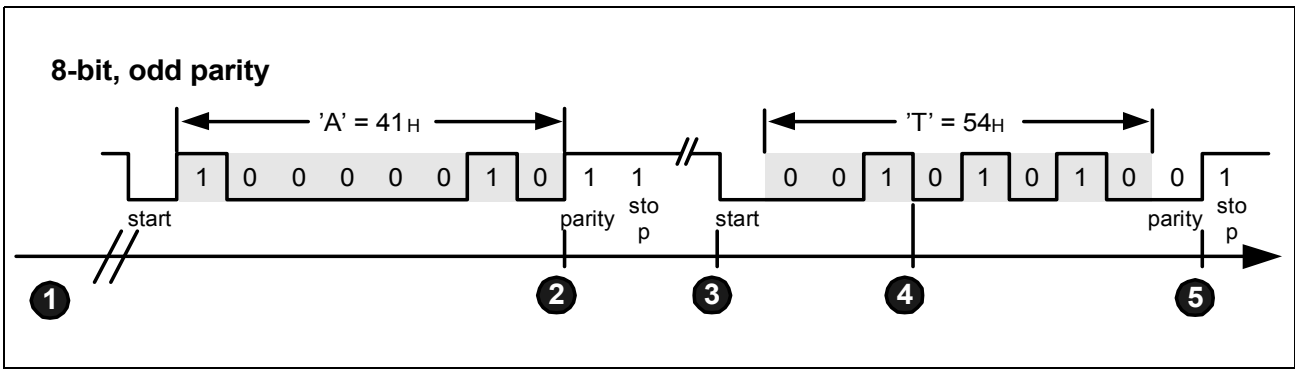

Figure 1 Sequence of events during autobaud detection

Note: The sequence starts with LSB.

1. Initialization of the autobaud module. The module is set up for the basic baudrates and the generation of an interrupt after successful detection of the first character is enabled. The fast external interrupt 2 (EX2INT) is set up for its alternate source ASC RX and is assigned to the falling edge of the incoming signal.
2. The controller initiates an autobaud detected interrupt (ABDETINT) after successful detection of the first character. The DMA mechanism will be used for capturing the

value of T3 on detection of an incoming falling edge. Therefore the PEC source pointer is set to the T3 register and the destination pointer is set to the start of a time buffer. The corresponding interrupt enable bit EX2IE is set and the PEC is configured for incrementation of the destination pointer for two single transfers.

3. The PEC moves the first timer value to the time buffer.
4. The PEC moves the second timer value to the time buffer and the EX2INT interrupt handler is called (see datasheet section “Operation of the PEC channels” within chapter “Interrupt and Trap Functions“ for details). The interrupt is disabled to avoid subsequent interrupts caused by the received data. At this point the time measurement is complete.
5. After detection of the second character the ABDETINT is issued for the second time, indicating the autobauding was successful and the ASC interface is ready for operation. The detected baud rate can now be calculated by means of the two timer values in the time buffer.

The following formula calculates the detected baud rate with the maximum absolute deviation of one timer tick (T_1 and T_2 symbolize the two captured timer values, f_{Timer} is the frequency of the used timer - in our example 9 MHz).

$$BR = 4\frac{f_{Timer}}{(T_1 - T_0)}$$

3 How to restart Autobauding

When it is required to restart autobauding after a successful detection the following sequence of commands is required to guarantee proper operation of the autobaud module under all conditions:

```
S0FDV    =    0;        // load ASC fractional divider register
ABS0CON  =    0xFFFF;   // load ASC autobaud control register
S0FDV    =    0x009D;   // load ASC fractional divider register
ABS0CON  =    0x001A
         =    0x0001;   // enable autobaud detection
```

After this sequence, the next AT command will be detected properly.

4 Sample Code

The sample application coming along with this application note demonstrates autobaud detection and the determination of the detected baud rate as described in the previous sections:

When executed on the controller, the program sets up for the autobaud detection (function ASC_vInit(void), see **Listing 1** below). When receiving "at" or "AT" at the serial interface, the baud rate will be calculated and send back to the interface in text format, together with the mode information (function ASC_viAbEnd(void),
see **Listing 1, File ASC.C**).

To test it with the evaluation system, attach a standard terminal to the serial port and type either “at” or “AT”. Please note that the displayed baud rate will not match exactly with the baud rate selected in the terminal options due to system dependant inaccuracy.

The used serial cable must not connect the serial DTR signal to the C156UTAH, C165H or C161U evaluation board. The DTR signal is used to reset the device and will hold the system in reset since most terminals drive this line!

The project has been set up using the Keil and the Tasking toolchain and can be easily re-compiled by either

opening the “.\Code\Keil\ASCab.uv2” project space within the Keil µVision

or

opening the “.\Code\Tasking\ASCab.pjt” project space for the Tasking EDE.

Listing 1 File ASC.C

```
//***************************************************************************
//  Copyright (c) 2001, Infineon Technologies AG, All rights reserved
//
//  NO WARRANTY
//
//  BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR
//  THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW.  EXCEPT WHEN
//  OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES
//  PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER
//  EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
//  WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE
//  ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH
//  YOU.  SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL
//  NECESSARY SERVICING, REPAIR OR CORRECTION.
//
//  IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING
//  WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR
//  REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR
//  DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL
//  DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM
//  (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED
//  INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF
//  THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR
```

```
//  OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
//------------------------------------------------------------------------------
//
// @Module       ASC
// @Filename     ASC.C
// @Project      ASC Autobauding Sample for the C165UTAH/H and C161U
//               device family
//
//------------------------------------------------------------------------------
//
// @Version      1.0
//
// @Description  ASC module
//
//------------------------------------------------------------------------------
// @Date         01/06/01
//
//******************************************************************************

//******************************************************************************
// @Project Includes
//******************************************************************************

#include <MAIN.H>

//******************************************************************************
// @Macros
//******************************************************************************

//******************************************************************************
// @Defines
//******************************************************************************

//******************************************************************************
// @Typedefs
//******************************************************************************

//******************************************************************************
// @Imported Global Variables
//******************************************************************************

//******************************************************************************
// @Global Variables
//******************************************************************************

uword auwTime[ 2 ];

//******************************************************************************
// @External Prototypes
//******************************************************************************

//******************************************************************************
// @Prototypes Of Local Functions
//******************************************************************************

//******************************************************************************
// @Function      void ASC_vInit(void)
//
//------------------------------------------------------------------------------
```

```
// @Description   This is the initialization function of the ASC function
//                library. It is assumed that the SFRs used by this library
//                are in its reset state.
//
//---------------------------------------------------------------------------
// @Returnvalue   None
//
//---------------------------------------------------------------------------
// @Parameters    None
//
//---------------------------------------------------------------------------
// @Date          01.06.01
//
//***************************************************************************

void ASC_vInit(void)
{

  S0CON        &=  ~0x8000;       // disable baud rate generator

  ///  ----------------------------------------------------------------------
  ///  Configuration of the ASC Autobaud Detection:
  ///  ----------------------------------------------------------------------
  ///  - the ASC module clock is 36 MHz
  ///  - autobaud detection is enabled
  ///  - bir AUREN is set, therefore bit REN is set after a successful
  ///    autobaud detection
  ///  - echo mode is disabled

  ///  - Difinition of the standart baud rates to be detected:
  ///  -   baud rate 1  =  230,400    kBaud
  ///  -   baud rate 2  =  115,200    kBaud
  ///  -   baud rate 3  =  57,600     kBaud
  ///  -   baud rate 4  =  38,400     kBaud
  ///  -   baud rate 5  =  19,200     kBaud
  ///  -   baud rate 6  =  9,600      kBaud
  ///  -   baud rate 7  =  4,800      kBaud
  ///  -   baud rate 8  =  2,400      kBaud
  ///  -   baud rate 9  =  1,200      kBaud

  S0FDV         =  0x009D;       // load ASC fractional divider register

  ///  - autobaud detection interrupt (ABDETIR) becomes active after the
  ///    recognition of the first byte ("a" or "A")
  ///  - with a successful autobaud detection some bits (M, ODD, BR_VALUE)
  ///    are automatically set to a value which corresponds to the mode and
  ///    badrate of the detected serial frame conditions

  ABS0CON       =  0x001A;       // load ASC autobaud control register

  ///  - check overrun error
  ///  - check framing error
  ///  - check parity

  S0CON         =  0x0827;       // load ASC control register

  ///  ----------------------------------------------------------------------
  ///  Configuration of the used ASC Port Pins:
  ///  ----------------------------------------------------------------------
```

```
  ///  - P3.10 is used for ASC Transmit Data Output (TxD0)
  ///  - P3.11 is used for ASC Receive data Input (RxD0)

  P3            |=  0x0400;       // set data register
  DP3           |=  0x0400;       // set direction register

  S0CON         |=  0x8000;       // enable baud rate generator

  ///  ----------------------------------------------------------------------
  ///  Configuration of the used ASC Interrupts:
  ///  ----------------------------------------------------------------------
  ///  - transmit service request node configuration:
  ///  - transmit interrupt priority level (ILVL) = 1
  ///  - transmit interrupt group level (GLVL) = 2

  S0TIC          =  0x003A;

  ///  - error service request node configuration:
  ///  - error interrupt priority level (ILVL) = 1
  ///  - error interrupt group level (GLVL) = 1

  S0EIC          =  0x0045;

  ///  - autobaud end service request node configuration:
  ///  - autobaud end interrupt priority level (ILVL) = 2
  ///  - autobaud end interrupt group level (GLVL) = 1

  ABENDIC        =  0x007E;

  /// Enable ASC Rx path as alternate source for external interrupt 2

  EXISEL    = 0x0010;    // Use alternate source ASC_RxD for external interrupt
  EXICON    = 0x0020;    // Generate interrupt on falling edges

  // Set up and start timer 3

  T3CON      =  0x0800;  // Use timer block in fast mode (fCPU / 4)
  T3         =  0x0000;  // reset timer register
  T3R       = 1;         // start timer

  // Connect EXINT2 to the PEC to read out T3 value on falling edges of the
  // incoming serial data

  FEI2IC     = 0x003F;  // Connect to PEC channel 7
  PECC7      = 0x0200;  // Increment destination pointer

                        // Copy value from T3 register to memory
  SRCP7 =  SOF( &T3 );
  PECSN7=  ( ( ulong )( &auwTime[ 0 ]  ) >> 8 ) & 0xFF00;


  ///  ----------------------------------------------------------------------
  ///  End of Autobaud initialization
  ///  ----------------------------------------------------------------------

  ABS0CON       |=  0x0001;       // enable autobaud detection

} //  End of function ASC_vInit
```

```
//****************************************************************************
// @Function      void ASC_vABRestart(void)
//
//----------------------------------------------------------------------------
// @Description   Restarts the autobauding after a succesful detection
//
//----------------------------------------------------------------------------
// @Returnvalue   None
//
//----------------------------------------------------------------------------
// @Parameters    None
//
//----------------------------------------------------------------------------
// @Date          01.06.01
//
//****************************************************************************

void ASC_vABRestart(void)
{

  ///  ------------------------------------------------------------------------
  ///  Restart autobaud detection
  ///
  ///  NOTE that the following sequence has to be retained to guarantee proper
  ///  operation under all circumstances
  ///  ------------------------------------------------------------------------

  S0FDV         =  0;         // load ASC fractional divider register
  ABS0CON       =  0xFFFF;    // load ASC autobaud control register

  S0FDV         =  0x009D;    // load ASC fractional divider register
  ABS0CON       =  0x001A     // load ASC autobaud control register
                |  0x0001;    // enable autobaud detection

  ABS0CON       |=  0x0001;   // enable autobaud detection

}

//****************************************************************************
// @Function      void ASC_viTx(void)
//
//----------------------------------------------------------------------------
// @Description   This is the transmit interrupt service routine for the ASC.
//                It is called when the sending of data is terminated (S0TIR
//                is set).
//                Please note that you have to add application specific code
//                to this function.
//
//----------------------------------------------------------------------------
// @Returnvalue   None
//
//----------------------------------------------------------------------------
// @Parameters    None
//
//----------------------------------------------------------------------------
// @Date          01.06.01
//
//****************************************************************************
```

```
#ifdef KEIL
void ASC_viTx(void) interrupt S0TINT_TRAP_NUMBER
#endif // #ifdef KEIL
#ifdef TASKING
void interrupt( S0TINT_TRAP_NUMBER )
   ASC_viTx(void)
#endif // #ifdef TASKING
{

} //  End of function ASC_viTx


//****************************************************************************
// @Function      void ASC_viRx(void)
//
//----------------------------------------------------------------------------
// @Description   This is the receive interrupt service routine for the ASC.
//                It is called if a byte has been received via ASC (S0RIR is
//                set).
//                Please note that you have to add application specific code
//                to this function.
//
//----------------------------------------------------------------------------
// @Returnvalue   None
//
//----------------------------------------------------------------------------
// @Parameters    None
//
//----------------------------------------------------------------------------
// @Date          01.06.01
//
//****************************************************************************

#ifdef KEIL
void ASC_viRx(void) interrupt S0RINT_TRAP_NUMBER
#endif // #ifdef KEIL
#ifdef TASKING
void interrupt( S0RINT_TRAP_NUMBER )
   ASC_viRx(void)
#endif // #ifdef TASKING
{

} //  End of function ASC_viRx


//****************************************************************************
// @Function      void ASC_viTxBuf(void)
//
//----------------------------------------------------------------------------
// @Description   This is the transmit buffer interrupt service routine for
//                the ASC. It is called if the content of the TX-buffer has
//                been loaded into the TX-shift register.
//                Please note that you have to add application specific code
//                to this function.
//
//----------------------------------------------------------------------------
// @Returnvalue   None
//
```

```
//---------------------------------------------------------------------------
// @Parameters    None
//
//---------------------------------------------------------------------------
// @Date          01.06.01
//
//***************************************************************************

#ifdef KEIL
void ASC_viTxBuf(void) interrupt S0TBINT_TRAP_NUMBER
#endif // #ifdef KEIL
#ifdef TASKING
void interrupt( S0TBINT_TRAP_NUMBER )
   ASC_viTxBuf(void)
#endif // #ifdef TASKING
{


} //  End of function ASC_viTxBuf


//***************************************************************************
// @Function      void ASC_viEXINT2(void)
//
//---------------------------------------------------------------------------
// @Description   This is the interrupt handler for fast external interrupt 2,
//                which we occur when the PEC has transfered the two time-
//                values from timer 3 in reaction to the falling edges of the
//                second autobauding character on the serial receive pin.
//
//---------------------------------------------------------------------------
// @Returnvalue   None
//
//---------------------------------------------------------------------------
// @Parameters    None
//
//---------------------------------------------------------------------------
// @Date          01.06.01
//
//***************************************************************************

#ifdef KEIL
void ASC_viEXINT2( void ) interrupt EXINT2_TRAP_NUMBER
#endif // #ifdef KEIL
#ifdef TASKING
void interrupt( EXINT2_TRAP_NUMBER )
   ASC_viEXINT2(void)
#endif // #ifdef TASKING
{
  // Disable the interrupt, since no more values have to be transferred
    FEI2IE = 0;
}


//***************************************************************************
// @Function      void ASC_viAbEnd(void)
//
//---------------------------------------------------------------------------
// @Description   This is the end of autobaud interrupt service routine for
```

```
//                the ASC. The ABDETIR autobaud detected interrupt is always
//                generated after recognition of the second character of the
//                two-byte frame, this means after a successful autobaud
//                detection. If ABCON_FCDETEN is set the ABDETIR autobaud
//                detected interrupt is also generated after the recognition
//                of the first charecter of the two-byte frame.
//                Please note that you have to add application specific code
//                to this function.
//
//------------------------------------------------------------------------------
// @Returnvalue   None
//
//------------------------------------------------------------------------------
// @Parameters    None
//
//------------------------------------------------------------------------------
// @Date          01.06.01
//
//******************************************************************************

#ifdef KEIL
void ASC_viAbEnd(void) interrupt ABENDINT_TRAP_NUMBER
#endif // #ifdef KEIL
#ifdef TASKING
void interrupt( ABENDINT_TRAP_NUMBER )
    ASC_viAbEnd(void)
#endif // #ifdef TASKING
{

  static double flBaudRate;
  ubyte ubMode;

  static uword uwABCount = 0;

  // check autobaud status in ABSTAT - second char detected?
  if ( !( ABSTAT & 0x000C ) )
  {
     // Detection of first character, so init PEC for baudrate detection on second
character

     PECC7 |= 0x0002;                // Do 2 transfers after this the EX2INT handler
will be executed
     DSTP7 = SOF( &auwTime[ 0 ] );  // Reset destination pointer

     FEI2IR    = 0;                    // reset interrupt request
     FEI2IE    = 1;                    // enable interrupt

  }
  else
  {
     // Autobaud detection done
     uwABCount++;

     // Calculate baudrate
     flBaudRate = 4 * 9000000 / ( float )( auwTime[ 1 ] - auwTime[ 0 ]);

     S0TIR = 1;
     printf( "CONNECT%6.0f/",  flBaudRate );
```

```
      ubMode = S0CON & 0x0007;

      switch ( ubMode )
      {
      case 1: // 8 bit asynchronous
         printf( "8N1\n" );
         break;
      case 3: // 7 bit asynchronous + parity
         if ( S0CON & 0x1000 )
        {
            printf( "7O1\n" );
        }
         else
        {
            printf( "7E1\n" );
        }
         break;
      case 7: // 8 bit data asynchronous + parity
         if ( S0CON & 0x1000 )
        {
            printf( "8O1\n" );
        }
         else
        {
            printf( "8E1\n" );
        }
         break;
      }

      // Toggle user LED on P3.3 (Infineon evaluation system only)
      LED_TOGGLE();

      // Restart autobaud detection
      ASC_vABRestart();

   }

} //  End of function ASC_viAbEnd
```

Infineon goes for Business Excellence

“Business excellence means intelligent approaches and clearly defined processes, which are both constantly under review and ultimately lead to good operating results.
Better operating results and business excellence mean less idleness and wastefulness for all of us, more professional success, more accurate information, a better overview and, thereby, less frustration and more satisfaction.”

Dr. Ulrich Schumacher

http://www.infineon.com

Published by Infineon Technologies AG