A2lPrep Special comments guide

1 Overview

This guide will help you place special comments in your source code so that your parameters and and signals can be easily tuned and displayed in HANtune. A2lPrep automatically generates ASAP2 files based on your C-source special comments.

2 Special Comment Syntax

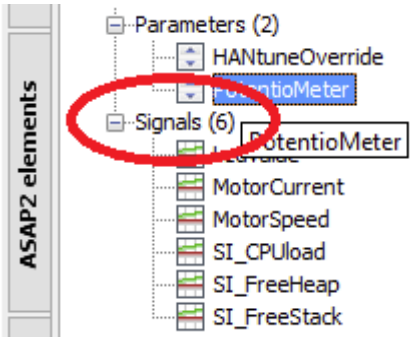
Every Special Comment starts with '`/*A2lPrep`' (no spaces and case sensitive) and ends with '`*/`'. Everything between these tokens will be interpreted as a special comment by the A2lPrep converter. It is interpreted as a comment by your C-compiler, therefore it will not influence the functionality of your code.

Example:

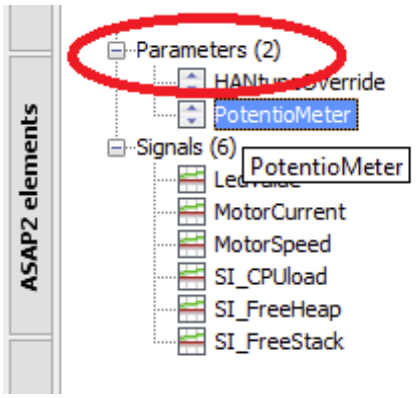
```
/*A2lPrep
  @Signal TempSensor
  @PrimitiveType U_BYTE
  @Description: "Example description"
*/
```

Special comments consists of the following 3 types of comments:

- Project specification: There should be exactly 1 project specification within your target C-source files passed to A2lPrep. It specifies some global values to be used within HANtune. Like CPU type, project name etc. You can place the project specification anywhere in your source code. It can be a `.c` or a `.h` file.
- Signal specification: A signal special comment specifies a ' MEASUREMENT' in an ASAP2 file after A2lPrep conversion. You can use as many signal specifications as you like. All the signals you specify will appear in the `Signals` section of the `ASAP2elements` sidebar in HANtune:



- Parameter specification: A parameter special comment specifies a 'CHARACTERISTIC' in an ASAP2 file. You can specify as many parameters as you like in your source code. Most likely you will place a parameter specification near the declaration of that parameter in your C-code. In HANtune your parameters will appear in the `Parameters` section of the `ASAP2elements`:



Project specification syntax

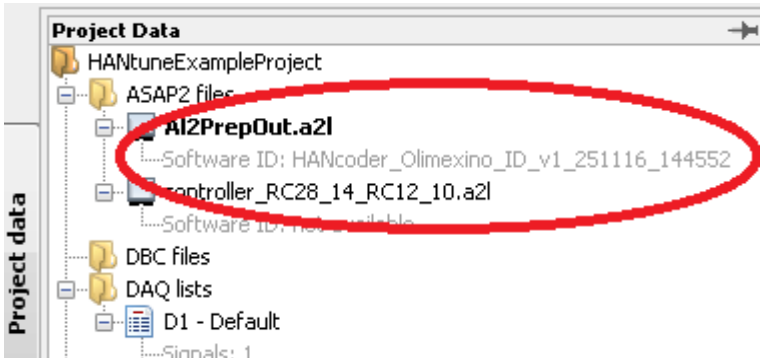
For project specification the following keywords can be used:

- `@ProjectId "<name>"`: Required; `<name>` holds a unique string containing target project name and unique version number (preferably including a build timestamp).
- `@AddressProjectId <address>`: Specification is required only when flashing a target is desired. `<address>` holds location where ProjectId string is stored in target memory. It can be a value holding a direct address or a string specifying the name of the variable holding the ProjectId. In this case the name should be present in the address map file with a valid address.
  When a direct address is specified, it can be a decimal value or hexadecimal. The latter must start with '0x'.
- `@CpuType "<cputype>"`: Optional. `<cputype>` holds a string that identifies the CPU type of the target.
- `@EcuType "<ecutype>"` : Optional. `<ecutype>` holds a string that describes the control unit. When Bodas Flash tool is used, `<ecutype>` must hold one of the hardware codes in the BODAS-service database to identify for reprogramming. For example: "RC28-14/30".
- `@Description <string>`: Optional; Free string for further clarification.

When the following project specification is used:

```
/*A2lPrep
  @ProjectId          "HANcoder_Olimexino_ID_v1_251116_144552"
  @AddressProjectId   XcpStationId
  @CpuType             "STM32F103"
  @EcuType             "Olimexino"
  @Description         "MOD PAR TestComment"
*/
```

It will show up like this in HANtune:



Signal specification syntax

For a signal specification, the following syntax should be used:

- `@Signal <name>`: Required. `<name>` should be a valid C variable and must be present in address file (.MAP).
- `@PrimitiveType <ItemPrimitiveType>`: Required. Specifies the primitive data type of the target for the signal. May only be one of the following:

PrimtiveType	Storage size	Value range	C data type
U_BYTE	1 byte	0..255	unsigned char
S_BYTE	1 byte	-128..127	signed char
U_WORD16	2 bytes	0..65535	unsigned short
S_WORD16	2 bytes	-32768 .. 32767	short
U_DWORD32	4 bytes	0 .. 4294967295	unsigned long
S_DWORD32	4 bytes	-2147483648 .. 2147483647	long
U_QWORD64	8 bytes (not yet supported by HANtune)	-2^{63} .. $2^{63}-1$	long long
S_QWORD64	8 bytes (not yet supported by HANtune)	0.. $2^{64}-1$	unsigned long long
FLOAT32_IEEE	4 bytes	±1,2E-38 .. ±3,4E+38	float
FLOAT64_IEEE	8 bytes (not yet supported by HANtune)	±2,25E-308 .. ±1,8E308	double

- `@StepSize <step>`: Optional, default=1.0. `<step>` and `<offset>` convert target value into physical display value. The following rule applies:

  `TargetValue = (DispVal - <offset>) / <step>`

  or

  `DispVal = Targetvalue * <step> + <offset>`

- `@Offset <offset>`: Optional, default=0.0. See also StepSize.
- `@DisplayFormat <display>`: Optional. default=DEC. Should be one of:

  `DEC|FLOAT(1|2|3|4)|HEX|<formatString>`

  formatString uses format `%5w.p` w holds width of floating point number. p holds number of digits after decimal point.

  Examples: `FLOAT3` or `HEX` or `"%4.3"`
- `@UnitText "<unit>"` : Optional, default="". Specifies a unit text when displayed in

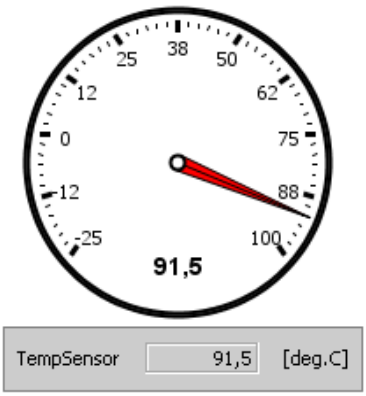
HANtune.

- `@LowerLimit <value>` : Optional, default=minimum value current PrimitiveType. Limit value must be specified in Target units
- `@UpperLimit <value>`: Optional, default=maximum value of current PrimitiveType. Limit value must be specified in Target units
- `@Description "<string>"` // Optional, default=""; Example: "Free text".

Example Signal specification (all optional fields present):

```
/*A2lPrep
  @Signal TempSensor
  @PrimitiveType U_BYTE
  @StepSize 0.5
  @Offset -25.0
  @DisplayFormat FLOAT1
  @UnitText "deg.C"
  @LowerLimit 0
  @UpperLimit 250
  @Description: "sensor 0..250 stored in byte (251..255 not
   used) corresponds to -25..100 degr. Centigrade."
*/
unsigned char TempSensor;
```

This will yield the following signal in HANtune:



Parameter Specification syntax

For a parameter specification in your special comments, the following syntax should be used:

- `@Parameter <name>`: Required. `<name>` should hold a valid C variable name and must be present in address file (*.map).
- `@Variety <ItemVariety>`: Required. `<ItemVariety>` must be one of the following:

VALUE	Represents a single parameter for tuning
CURVE	Represents a 1-dimensional lookup table. Needs to be accompanied by 1 index table (AXIS_PTS)
MAP	Represents a 2-dimensional lookup table. Requires 2 index tables (AXIS_PTS) specified
AXIS_PTS	Represents an index table (used by CURVE and MAP)

- `@PrimitiveType <ItemPrimitiveType>`: Required. Specifies the data type for the parameter. Same definition used as Signal Specification (see above).
- `@NumberOfAxisPts <value>`: For AXIS_PTS only, Required. Specifies the number of elements in a table. Must be greater than 0. Must be equal to the size of the corresponding table in C-code.
- `@StepSize <step>` : Optional, default=1.0; Same rules apply as for Signal.
- `@Offset <offset>` : Optional default=0.0; Same rules apply as for Signal.
- `@DisplayFormat <display>` : DEC|FLOAT(1|2|3|4)|HEX|<formatString>; Optional Same rules apply as for Signal. FormatString can be “%x.y”
- `@UnitText "<unit>"` : Optional, default="". Specifies a unit text when displayed in HANtune.
- `@LowerLimit <value>` : Optional, default=minimum value current PrimitiveType. Limit value must be specified in Target units
- `@UpperLimit <value>`: Optional, default=maximum value of current PrimitiveType. Limit value must be specified in Target units
- `@Description "<string>"` Optional, default=""; Example: "Free text"
- `@IndexTable <name>` : Required when variety is CURVE; <name> must hold name of an AXIS_PTS table
- `@IndexTableRowsY <name>` : Required when variety is MAP; <name> must hold name of AXIS_PTS table for Y-axis
- `@IndexTableColsX <name>` : Required when variety is MAP; <name> must hold name of AXIS_PTS table for X-axis.

Example VALUE parameter with full specification (all optional keywords present):

```
// 0..65536 limited to 50..50000 corresponds to
//-187.50..12300.00 volt displayed in physical units
unsigned short VoltageThreshold;
/*A2lPrep
  @Parameter  VoltageThreshold
  @Variety VALUE
  @PrimitiveType U_WORD16
  @StepSize 0.25
  @Offset -200
  @DisplayFormat  "%3.2"
  @UnitText "Volt"
  @LowerLimit 50   // phys = 50 * 0.25 - 200 => -187.50
```

```
  @UpperLimit 50000 // phys = 50000 * 0.25 - 200 => 12300.00
  @Description "Step: 0.25; Min: -187.50; Max: 12300.00 Volt"
*/
```

The above comment will show up in HANtune like this:

Stepsize will be 0.25 volt. Minimum value will be -187.50 and max will be set to 12300.00 Volt.

Example VALUE parameter (single value) with minimal specification (only required keywords used):

```
float AirPressureThreshold;
/*A2lPrep
  @Parameter  AirPressureThreshold
  @Variety VALUE
  @PrimitiveType FLOAT32_IEEE
*/
```

The above comment will show up in HANtune like this:

The stepsize will be (default) 1, minimum and maximum values: -3.4028e+38 and 3.4028e+38.

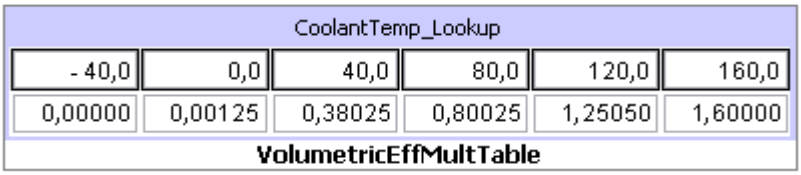
Example CURVE parameter (1-D table) with full specification (all optional keywords present). 2 special comments are required. A CURVE specification and an AXIS_PTS specification:

```
unsigned short VolumetricEffMultTable[6];
/*A2lPrep
  @Parameter VolumetricEffMultTable
  @Variety CURVE
  @PrimitiveType U_WORD16
  @StepSize 2.5e-5 // = 1/40000
  @Offset   0
  @DisplayFormat "%6.5"
  @UnitText "Mult"
  @LowerLimit 0
  @UpperLimit 64000
  @Description "Step: 0.000025; Min: 0.00000; Max: 1.60000"
  @IndexTable CoolantTemp_Lookup
*/

unsigned char CoolantTemp_Lookup[6];
/*A2lPrep
  @Parameter CoolantTemp_Lookup
```

```
  @Variety AXIS_PTS
  @PrimitiveType U_BYTE
  @NumberOfAxisPts 6
  @StepSize 0.8
  @Offset   -40
  @DisplayFormat FLOAT1
  @UnitText "Degr.C"
  @LowerLimit 0
  @UpperLimit 250
  @Description "Step: 0.8; Min: -40.0; Max: 160.0"
*/
```

The above will be displayed in HANtune like this:



The minimum and maximum values for lookup table: -40.0 and 160.0 Degr.C

Minimum and maximum for VolumetricEffMultTable: 0.00000 and 1.60000

Example MAP parameter (2-D table) with full specification (all optional keywords present). 3 special comments are needed. A MAP specification and 2 AXIS_PTS specifications:

```
unsigned short IgnitionAngleTable[3][5];
/*A2lPrep
  @Parameter IgnitionAngleTable
  @Variety MAP
  @PrimitiveType S_WORD16
  @StepSize 0.00390625
  @Offset 0.0
  @DisplayFormat "%6.4"
  @UnitText "degr"
  @LowerLimit -30000
  @UpperLimit 30000
  @Description "Step: 0.00390625; Min: -117.1875;
   Max:    117.1875 degr"
  @IndexTableColsY ManifoldAirPress_Lookup
  @IndexTableRowsX ThrottlePosSens_Lookup
*/

unsigned short ManifoldAirPress_Lookup[5];
/*A2lPrep
  @Parameter ManifoldAirPress_Lookup
  @Variety AXIS_PTS
```

```
  @PrimitiveType S_WORD16
  @NumberOfAxisPts 5
  @StepSize 0.00390625
  @Offset 0.0
  @DisplayFormat FLOAT3
  @UnitText "kPa"
  @LowerLimit 100
  @UpperLimit 32000
  @Description "Step: 0.00390625; Min: 0.390625;
   Max: 125.000 kPa"
*/

unsigned char ThrottlePosSens_Lookup[3];
/*A2lPrep
  @Parameter ThrottlePosSens_Lookup
  @Variety AXIS_PTS
  @PrimitiveType U_BYTE
  @NumberOfAxisPts 3
  @StepSize 0.00390625
  @Offset 0.0
  @DisplayFormat FLOAT1
  @UnitText "%"
  @LowerLimit 0
  @UpperLimit 25600
  @Description "Step: 0.00390625; Min: 0.000; Max: 100.0 %"
*/
```

The above will be displayed in HANtune:

ManifoldAirPress_Lookup

ThrottlePosSens...	0,3906	1	20	30	44
5	-117,1094	-110,5	0	50	100
40,5	20,3281	30	44	55,5	2,25
100	20,5	30,25	50,5	60,5	117,1875

IgnitionAngleTable

The minimum and maximum values for 5 x 3 lookup table: -117,1875 and 117,1875 Degr.