Technical Reference

Special Features, product specific

The functions to connect the MACS4-DC6 controller are available starting with firmware version 6.6.34 and following versions:

- Power-down save functions (6.6.34)
- Supply voltage readout (6.6.65)
- Power down trigger voltage level (6.6.65)
- Permanent data save (serial number) (6.7.20)

Power-down data saving (MACS4-DC6 only)

The power down save on MACS4-DC6 stops the axes and saves the encoder positions and 10 user defined parameters (if desired) when the power falls below 17 V.

This function is deactivated by default and must be activated by the user after every power-up of the controller.

Some changes have been made to the implementation to save code size (6.7.20). Hence, the way a checksum is calculated has changed. As a result, data saved with older firmware will no longer be detected as valid after a firmware update.

SETSYS and GETSYS Commands

A power-down save can be forced by calling **SETSYS 17 : 20 0**

SETSYS 16 value — This is used to activate the power-down save.

If value is ==	0, the power-down function is disabled (default)
If value is ==	1, the last power-down entry is erased in flash, and the power-down save is activated.
If value is > 1	additionally the stop wait time is set (default is 20,000 us = 20 ms). The stop wait time is the wait time between issuing the stop command and when the encoder positions are latched and saved. For example:

```
SETSYS 16 25001 // calling SETSYS 16
        // erases the current saved dataset
        // enables the power-down function (if given value > 0)
        // if power-down function is enabled and power fails later on, …
        // … "motor off" command is processed (i.e. position control is off)
        // … speed control / PWM command is set to 0
        // … a wait 25 ms (according to given value) is processed
        // … position value of all axis are saved
```

SETSYS 17: index sysvar — This is used to setup user defined SYSVARs to be saved at power-down. It must be called with an index value of 10 through 19, which equals the index that must be called with GETSYS 16 to read the saved value.

GETSYS 16: index — Read out a saved value or the data valid flag.

The data valid flag shows if the saved dataset is valid (the last save action succeeded)

Index -1	read data valid flag (1 = data is valid, 0 = data is not valid)
Index 0-5	read saved encoder positions (not axis- but encoder number-related)
Index 10-19	read saved SYSVAR

Supply Voltage Readout

The Supply Voltage on the MACS4-DC6 can be read out using

SYSVAR[0x01270001 + ((AxNr-1)*0x200) + 23]. The unit is mV.

Power Down Trigger Voltage Level

The voltage at which the power-down mechanism is triggered can be set using SETSYS 42.

The unit is mV. The default is 17V and presumably does not need to be altered.

Product identification by SDO (all MACS4)

63 long values are saved permanently on a MACS4. The SDO 0x2215 has been implemented for this.

The values can be read out by reading the subindexes 1-64. If the checksum of the saved data is invalid, a ERR_COP_RestoreParamWrongSignature error will be returned to the SDO request, and the value of the SDO will be returned as 0.

The permanent data is located in the very last sector of the flash. Power-down data is written at the beginning of the last kilobyte of the flash (4th-last sector).

See SDO Object Dictionary

DCP commands (MACS4 and RS485)

Starting with firmware 6.7.73, there are DCP commands for MACS4 RS485 interface:

INITDCP option

Option value

1 = init rs485 telegram handler
0 = stop rs485 telegram handler
-1 = reset telegram handler

DCPREADCOMM

This command checks for an incoming "high level" telegram and will return its user data if a telegram has arrived.

result = DCPREADCOMM array

array represents the Array which is used to fill in the received bytes, 1 byte per array element (lowest byte).

The result has the following meaning:

OK	x (>0)	TG has arrived with x bytes user data
ACTIVE	0	No TG arrived
TIMEOUT	-1	
BADFRAME	-2	The received frame is not valid.
OVERFLOW	-4	Received more than the receive buffer can take.

DCPWRITECOMM

The data which the user has filled into the array will be sent as it is (no telegram control bytes).
After sending, the RS485 will again be in receiving state.

result	=	DCPWRITECOMM length array
length	=	number of bytes to send (if set to 0 no telegram will be sent, but can be used to get communication state)
array	=	array which contains the data to be sent, 1 byte per element, lowest byte result has the following meaning

SENT	1	Telegram will be sent and previous telegram has been sent successfully
OK	0	Telegram will be sent
TIMEOUT	-1	The previous transmission has been timed out (new telegram will be sent anyway).
BUSY	-3	There is still another transmission ongoing and not timed out yet, you must retry later.

SETSYS 20 parity baudrate

Sets up the serial interface for the RS485 communication:

parity: 0 = no parity, 1 = odd parity, 2 = even parity

baud rate: serial baud rate, e.g. 19200

Example:
SETSYS 20: 2 19200 sets up the interface for 19200 baud and even parity

Array Structure of CAM Profiles

This section documents data structures and compiler details which are only required in exceptional cases by the user. For example, if an automatically generated programming is to be modified like a curve profile.

Header

The header contains general information like

- Identification for curve array
- Version number for curve structure
- Type of curve
- Name of curve
- Index to curve information section
- Index to start/stop point section
- Index to fixed point section
- Index to interpolation point section
- Index to start/stop point indices (in interpolation section)
- Index to start/stop velocities (times 100000)
- Index to start path interpolation points
- Index to stop path interpolation points

Curve information section

This section of the array contains all information about the type of curve like

- Length of curve (master)
- Length of curve (slave)
- Number of fix points
- Number of Interpolation points (this gives the resolution)
- Type of interpolation
- Slave stop point, point where slave is positioned, when synchronization is stopped
- Correction start point (only valid for marker synchronization)
- Correction end point (only valid for marker synchronization)
- Maximum correction which is allowed (only valid for marker synchronization)
- Maximum start/stop path length (Size of start/stop path area)(min. 2)
- No of start/stop point pairs
- Maximum number of cycles per minute (Application information)

Curve start/stop point section

This section contains the start/stop points. Because the use of this point is up to the user, we just speak of a path, which can be a start or a stop sequence. Every path consists of 2 points. If we are moving forward, the path starts (start or stop) with the a-point and ends with the b-point. If we are moving backward, the path starts with the b-point and ends with the a-point. So the user is able to tell us in the program, which pair of points to use for starting or stopping, when he uses a STARTCURVE or STOPCURVE command.

- Path 1 (a – point)
- Path 1 (b – point)
- Path 2 (a – point)
- Path 2 (b – point)
- …

These points have to lie on interpolation points, so possibly the PC software has to adjust them according to the interpolation resolution. This should not be a real restriction, because the interpolation points are normally very dense. So for example if we have rotating master which makes one revolution per cycle and we choose a cycle length of 3600 MU (1 MU = 1/10 degree). Let us further assume, that we choose the number of interpolation points as 1200, than you have a resolution of 3 MU = 3/10 degree for defining your start and stop points.

Fixed point section

This section contains the fix points, which were the basis for the interpolation calculation. These points always consist of the following triple

- Master coordinate
- Slave coordinate
- Type of point (tangent, curve)

These points are defined by the user in MU units (see internal description). If you want to avoid, that the real interpolation curve misses your fix points, you have to choose them in such a manner that they lay on an interpolation point (see above). This can be forced through a snap function within the PC software.

Interpolation point section

This section contains a list of slave coordinates. They belong to master coordinates which are of equal distance, given by the interpolation resolution.

Indices of start/stop points

Here we have the indices of the start/stop points (see above) within the interpolation array. These are necessary for the ease of start and stop recognition. We are waiting until start index for example equals the actual index and direction of movement is correct. If both is true, we start synchronization. The same is true for stopping.

Start Stop Velocities

To be able to calculate an appropriate starting or stopping path, we need the velocity we have to reach at end (start) or we will have at the beginning (stop) in UU/MU units (Slave units per Master units).

Start / Stop paths

This is the place for the interpolation points of the actual start and stop path. These points are calculated when a SYNCCSTART or SYNCCSTOP command is executed, but we have to reserve the room right now.

CAM Array Definition

Part	Index	Name	Unit	Value	Description
General	1	Identification	(dec)	999.000.001	Number to identify array
	2	VersioNumber	(dec)	100	Version as decimal (1.00 = 100)
	3	CurveType	(dec)	0	0 = symmetrical, 1 = compatible
	4	CurveName 1	(4char)	Nona	Name of curve total 16 char.
	5	CurveName 2	(4char)	meCu	default is:
	6	CurveName 3	(4char)	rve0	NonameCurve00001
	7	CurveName 4	(4char)	0001	
	8	IndexCIF	(dec)	16	Index to Curve Information Part
	9	IndexSTP	(dec)	27	Index to Start/Stop Point Part
	10	IndexFIP	(dec)	IndexSTP + STPno*2	Index to Fix Point Part
	11	IndexINP	(dec)	IndexFIP + FixPointNo * 3	Index to Interpolation Point Part
	12	IndexSTPInd	(dec)	IndexINP + InterpolPointNo	Index to StartStop Interpolation Indices
	13	IndexSTPVel	(dec)	IndexSTPInd +STPno*2	Index to StartStop Velocities
	14	IndexSTIP	(dec)	IndexSTPVel +STPno*2	Index to Startpath interpolation points
	15	IndexSTPIP	(dec)	IndexSTIP + MaxStartStopLen	Index to Stoppath interpolation points
Curve Information	1	MasterCycleLen	MU	-	Length of Curve in CurveMaster units
	2	SlaveCycleLen	UU	-	Slave max. travel distance in CurveSlave units
	3	FixPointNo	(dec)	4	Number of fix points (minimum 4)
	4	InterpolPointNo	(dec)	-	Number of interpolation points (including first and last, which correspond to the same location)
	5	InterpolType	(dec)	0	0 = cubic spline 1 = periodic cubic spline
	6	SlaveStopPosition	UU	0	Position, where slave stands after stopping
	7	CorrectionStartPoint	MU	0	Position, where Correction may start
	8	CorrectionStopPoint	MU	MasterCycleLen	Position, where Correction has to be finished
	9	MaximumCorrection	UU	-	Maximum Correction which is allowed in one cycle
	10	MaxStartStopLen	(dec)	0	Maximum length of start/stop path (no of int. points)
	11	StartStopNo	(dec)	0	Number of start stop point pairs (n) (see below)
	12	MMaxCycles	(dec)	0	Max. number of cycles per minute (application info)
	13	MMarkerPos	CM	0	Master Marker Position in curve
	14	SMarkerPos	CS	0	Slave Marker Position in curve
Start/Stop Point	1	STPoint_1.a	MU	0	Start (forward) / Stop (backward) Point no. 1
	2	STPoint_1.b	MU	0	Stop (forward) / Start (backward) Point no. 1
	3	STPoint_2.a	MU	0	Start (forward) / Stop (backward) Point no. 2
	4	STPoint_2.b	MU	0	Stop (forward) / Start (backward) Point no. 2
	5	...	MU	0	
	6	...	MU	0	
	2*n-1	STPoint_n.a	MU	0	Start (forward) / Stop (backward) Point no. n
	2*n	STPoint_n.b	MU	0	Stop (forward) / Start (backward) Point no. n

Part	Index	Name	Unit	Value	Description
Fix Point	1	FixPoint_1.master	MU	0	Fix point no. 1 - master coordinate
	2	FixPoint_1.slave	UU	-	Fix point no. 1 - slave coordinate
	3	FixPoint_1.type	(dec)	C	Fix point no. 1 - type of point (C = Curve Point, T = Tangent Point
	4	...			
	5	...			
	6	...			
	3*n-2	FixPoint_n.master	MU	MasterCycleLen	Fix point no. n - master coordinate
	3*n-1	FixPoint_n.slave	UU	-	Fix point no. n - slave coordinate
	3*n	FixPoint_n.type	(dec)	C	Fix point no. n - type of point (C = Curve Point, T = Tangent Point)
Interpolation Point	1	IntPoint_1	UU	0	Interpolation Point no. 1 - slave coordinate
	...				
	n	IntPoint_n	UU	-	Interpolation Point no. n - slave coordinate
StartStop Indices	1	STPoint_1.a-index	(dec)	0	Index in Interpolation Array, corresponding to startpoint
	2	STPoint_1.b-index	(dec)	0	Index in Interpolation Array, corresponding to startpoint
	3	..			
StartStop Velocities	1	STPoint_1.a-veloc.	(dec)	(*100000)	Velocity (UU/MU * 100000) in startpoint
	2	STPoint_1.b-veloc.	(dec)	(*100000)	Velocity (UU/MU * 100000) in startpoint
	...				
StartPath Interpolation Points	1	StartPoint_1	UU	0	Interpolation Point no. 1 - for start path
	...				
	n				
StopPath Interpolation Points	1	StopPoint_1	UU	0	Interpolation Point no. 1 - for stop path
	...				
	n				

Curve Arrays and Curve Types

Starting with firmware > 6.6.0 no interpolation points are used anymore. So only the fix points are relevant for the curve. When a SETCURVE is executed (or when the curve is really started), the coefficients for the corresponding polynomials are calculated. Then, when the curve is running, the polynomials are calculated on the fly, while driving.

This procedure allows the user to modify a curve on the fly within the application program. This can be done by overwriting some of the values within the curve array. (See description of curve array in Illustrations.) After that a SETCURVE must be executed to activate the modified curve. This curve is then started as soon as the active curve ends, or immediately by replacing the active curve, if the new curve is of type INTRPT_GRAD (see array description).

!!! The curve array is used by the internal SYNCC procedures as long as the curve is running. So you should never modify the curve array of a running curve. To solve this problem you must have two arrays which are used alternatively. That means while one curve is active, the next one can be prepared and started. As soon as the new one is active (PFG_CWRAP changes), the old curve can be modified again.

Type of Fix points

Starting with version 6.6.40 there are new types of fix points. The way tangent points are used is changed. Now the first fix point always tells what type of segment is following. So the last point is always of same type as the first point.

There are new points like Poly3 and Trapezoidal which allow special segments within the curve to be used.

For new curves only the **bold** point types should be used.

#define CU_CPOINT	**1**	**// Curve point (next segment is 3'rd or 5'th order polynomial).**
#define CU_T1POINT	2	// Tangent start point (replaced by CU_TPOINT).
#define CU_T2POINT	3	// Tangent end point (replaced by CU_CPOINT).
#define CU_TPOINT	**4**	**// Next segment is a tangent segment.**
#define CU_ZPOINT	**5**	**// Next segment is a trapezoidal segment.**
#define CU_3POINT	**6**	**// Next segment is a 3'rd order polynomial.**

So you can, for example, create sequences like .. 4 – 5 – 4 – 5 which means that you will have two straight lines (tangents) which are connected by two parabolas. Those two parabolas meet each other in the middle between the fix points and at all three points (start, middle, end) of the segments, the velocity is the same as the adjacent segment.

The following new curve types can be used by changing the array:

New Curve type 3 – CU_GRAD

Starting with version 6.5.05 another type of curve (3) is supported. This curve consists of only 2 fix points and is calculated as a polynomial of 5th order. Therefore, the following values were added at the end of the fix point area in the curve array (G_CFPIdx is the index of the fix point area):

```
RestartCurve[3] =                  3          // Curve type
RestartCurve[G_CFPIdx+0] =   0          // Master start coordinate
RestartCurve[G_CFPIdx+1] =   0          // Slave start coordinate
RestartCurve[G_CFPIdx+2] =   1          // Fix Point Type = curvepoint
RestartCurve[G_CFPIdx+3] = G_ShingleDistance * 4    // Master end coordinate
RestartCurve[G_CFPIdx+4] = G_ShingleDistance * 2    // Slave end coordinate
RestartCurve[G_CFPIdx+5] =   1          // Fix Point Type = curvepoint
RestartCurve[G_CFPIdx+6] =   0          // Velocity v0
RestartCurve[G_CFPIdx+7] =   1          // Divisor v0
RestartCurve[G_CFPIdx+8] =   0          // Acceleration a0
RestartCurve[G_CFPIdx+9] =   1          // Divisor a0
RestartCurve[G_CFPIdx+10] =  0          // Jerk j0
RestartCurve[G_CFPIdx+11] =  1          // Divisor j0
RestartCurve[G_CFPIdx+12] =  1          // Velocity v1
RestartCurve[G_CFPIdx+13] =  1          // Divisor v1
RestartCurve[G_CFPIdx+14] =  0          // Acceleration a1
RestartCurve[G_CFPIdx+15] =  1          // Divisor a1
RestartCurve[G_CFPIdx+16] =  0          // Jerk j1
RestartCurve[G_CFPIdx+17] =  1          // Divisor j1
```

That means we have the possibility to define start and end gradients and acceleration for the polynomial. (Jerk is ignored at the moment. Designed for future use.)

In this array, the start coordinates are only for display purposes because they are replaced by the actual values when the curve is started. (see below)

As a result of this behaviour of type 3 curves (predefined end values and calculated start values), they normally cannot be continued when they reach the end (since typically the velocities do not match). Therefore, they are normally continued by another standard curve. If for any reason they are not continued by another curve, they will just try to continue with the actual velocity. This is done with a poly5 looking more or less like a straight line.

!!! This continuation overwrites the original curve array with this continuation curve.

Continuation did not work for curves with more than 2 fix points prior to 6.7.11. In those versions, there was also an error when CU_GRAD curves with more than 2 fix points where used as a continuation curve.

New Curve type 4 CU_GRAD_INTRPT

Type 4 is available starting with version 6.5.05. This type is nearly identical to type 3 (CU_GRAD).

The big difference with curves of type CU_GRAD_INTRPT is that they are started immediately when the SETCURVE is executed. When this is done, the actual values for velocity and acceleration are used for the calculation. The actual values of the MCPOS and CURVEPOS are subtracted from the end coordinates of the curve before it is calculated (curves must always internally start at 0,0). This guarantees, that the original end coordinates are absolute to the start of the interrupted curve.

For example, assume that a curve is running which starts at 0,0 and ends at 2000,2000 (master,slave). Now we define a curve of type CU_GRAD_INTRPT, which starts anywhere and ends at 4000,4000. If this curve is now set by SETCURVE at the moment when the original curve passes 1500,1800, for example, then the new curve is calculated in such a manner that

it starts at this point (1500,1800) and ends at 4000,4000. To realise this, it uses the velocity and acceleration in the actual point, sets MCPOS and CURVEPOS to 0 and reduces the end coordinates to (4000-1500, 4000-1800) = (2500,2200). It will have the defined velocity (v1) and acceleration (a1) defined in the curve array.

These types of curve are used for processes where the standard curve looks more or less like a straight line (SYNCP / SYNCM behaviour) and where the poly5 curves are used to align start or stop or restart processes to defined points.

!!! The responsibility for the correctness of poly5 curve lies with the user / application. The firmware does not do any plausibility test.

To allow readability by CAM-Editor, the CurveVersion (index 2) should be 102. Otherwise, the CAM editor will not accept those new curves.

Curve type 3 CU_GRAD with SYNCCSTART

Such curves (Poly 5) can now also be started with SYNCCSTART. This means the curve starts immediately and does not wait for next marker. (In previous versions, it was only possible to start such a curve with SYNCMSTART 2001.)

If such a curve is started now with SYNCCSTART, the user is responsible for the correct setting of the endpoints. Startpoint is 0 which will be internally set to the actual command position. Hence, the curve must be defined from 0 .. endpoint.

Example:

```
G_CFPIdx = StartCurve[10]
  // Array index, where fix point definition starts
  // 1. Fix Point has to be 0/0,
  // because curve always has to start at this position
StartCurve[G_CFPIdx+0]  = 0    // Master start coordinate
StartCurve[G_CFPIdx+1]  = 0    // Slave start coordinate
StartCurve[G_CFPIdx+2]  = 1    // Fix Point Type = curvepoint
StartCurve[G_CFPIdx+3]  = G_ShingleDistance * 4
   // Master end coordinate
StartCurve[G_CFPIdx+4]  = G_ ShingleDistance * 2
   // Slave end coordinate
StartCurve[G_CFPIdx+5]  = 1    // Fix Point Type = curvepoint
StartCurve[G_CFPIdx+6]  = 0    // Velocity v0
StartCurve[G_CFPIdx+7]  = 1    // Divisor v0
StartCurve[G_CFPIdx+8]  = 0    // Acceleration a0
StartCurve[G_CFPIdx+9]  = 1    // Divisor a0
StartCurve[G_CFPIdx+10] = 0    // Jerk j0
StartCurve[G_CFPIdx+11] = 1    // Divisor j0
StartCurve[G_CFPIdx+12] = 1    // Velocity v1
StartCurve[G_CFPIdx+13] = 1    // Divisor v1
StartCurve[G_CFPIdx+14] = 0    // Acceleration a1
StartCurve[G_CFPIdx+15] = 1    // Divisor a1
StartCurve[G_CFPIdx+16] = 0    // Jerk j1
StartCurve[G_CFPIdx+17] = 1    // Divisor j1

SETCURVE StartCurve
….
SYNCC 0
SYNCCSTART 0
```

In such a case, when SYNCCSTART is executed, the 0 point of the curve is mapped onto the actual command position. Also, the actual velocity is calculated (and start acceleration is set to zero).

Curve type 3 CU_GRAD with SYNCCSTART and DEFSYNCORIGIN

In addition to the above mentioned possibility, you have the option to define the endpoints of the curve with absolute values. So the start of such a curve could look like

```
mendpos = MIPOS + mdistqc
  // apos must be in qc
sendpos = (sdistqc % G_SlaveQcProProdukt) * G_SlaveQcProProdukt +
SYSVAR[4098]
defsyncorigin mendpos sendpos
  // define target position for master and slave (in qc)
SYNCC 0
SYNCCSTART 0
```

In this case, the curve is again started immediately and the actual command position will be the Curve zero position. However, the end position of master and slave are calculated in such a way that the curve will end at the given absolute positions.

Final velocity is given by the curve (1 in our example above) and start velocity is taken from actual values.

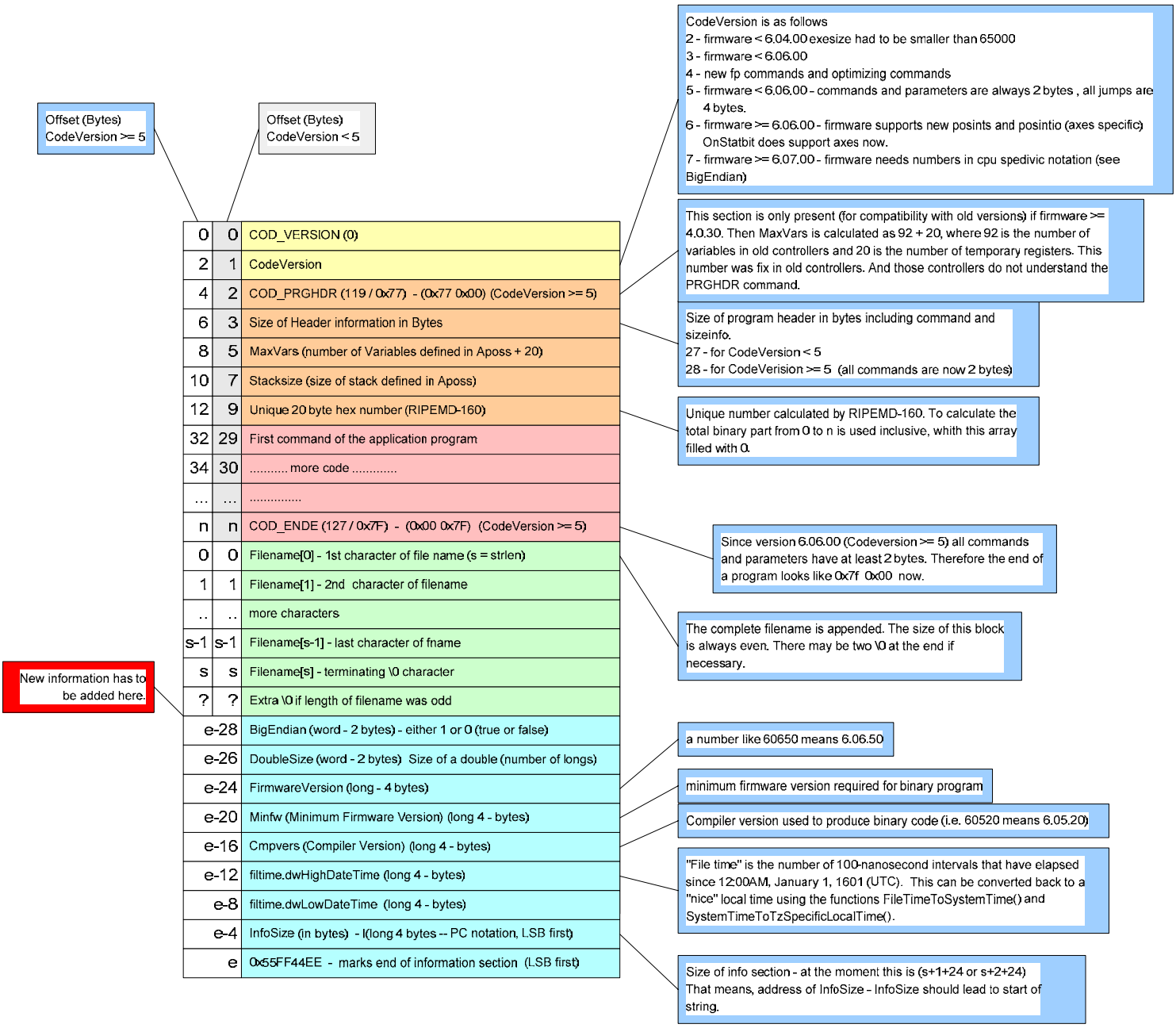
Curve types CU_GRAD with stability check

Whenever the new curve types are calculated, a check for extremes within a poly5 is also done if possible. (This only works if start acceleration is 0.) If an extreme within the interval is found, the curve error flag PG_FLAG_CURVE_ERR is set and the error number is stored in the PFG_G_LastError which allows the user to detect this situation. At the same time, the PFG_G_CPOLYMAXVEL [4288] and PFG_G_CPOLYMINVEL [4289] (SU/MU) are stored. Writing to PFG_G_LastError [4258] clears the flag PG_FLAG_CURVE_ERR (Bit 64 <<24) in the STAT.

So a sample could look like.

```
IF((STAT x(1))&(64<<24)) THEN // error bit set
   switch(SYSVAR[4258])
      case 5 :  PRINT " Minimum in interval ",SYSVAR[4289] * 100 % 128
                break
      case 6 :  PRINT " Maximum in interval ",SYSVAR[4288] * 100 % 128
                break
      case 7 :  PRINT " Minimum in interval ",SYSVAR[4289] * 100 % 128
                PRINT " Maximum in interval ",SYSVAR[4288] * 100 % 128
                break
      default : PRINT " Other curve error ",SYSVAR[4258]
   endswitch
ENDIF
```

Bin File Map (Compiler >= 6.5.36)

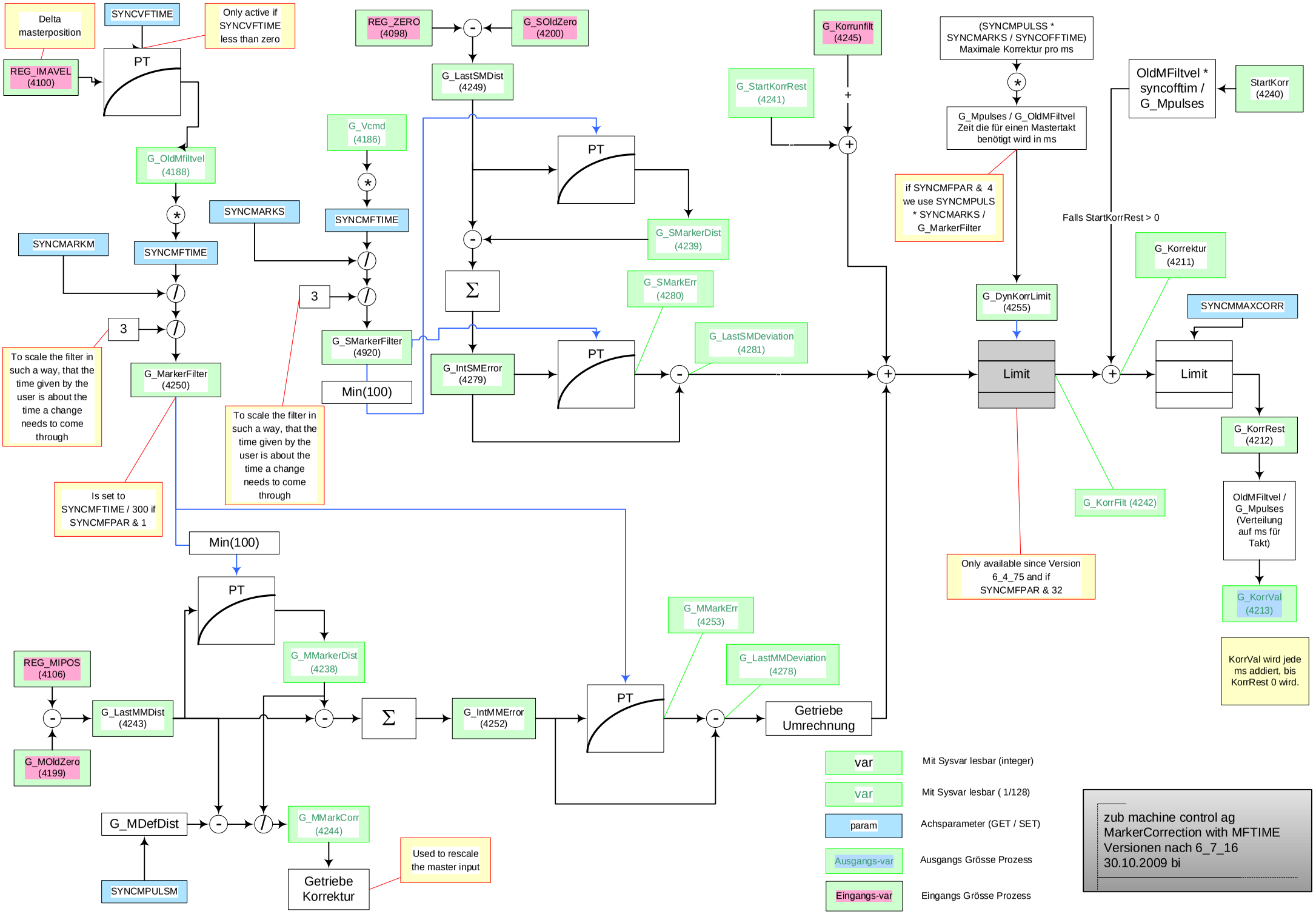


0	0	COD_VERSION (0)
2	1	CodeVersion
4	2	COD_PRGHDR (119 / 0x77) - (0x77 0x00) (CodeVersion >= 5)
6	3	Size of Header information in Bytes
8	5	MaxVars (number of Variables defined in Aposs + 20)
10	7	Stacksize (size of stack defined in Aposs)
12	9	Unique 20 byte hex number (RIPEMD-160)
32	29	First command of the application program
34	30	..... more code .....
...	...	.....
n	n	COD_ENDE (127 / 0x7F) - (0x00 0x7F) (CodeVersion >= 5)
0	0	Filename[0] - 1st character of file name (s = strlen)
1	1	Filename[1] - 2nd character of filename
..	..	more characters
s-1	s-1	Filename[s-1] - last character of fname
s	s	Filename[s] - terminating \0 character
?	?	Extra \0 if length of filename was odd
e-28		BigEndian (word - 2 bytes) - either 1 or 0 (true or false)
e-26		DoubleSize (word - 2 bytes) Size of a double (number of longs)
e-24		FirmwareVersion (long - 4 bytes)
e-20		Minfw (Minimum Firmware Version) (long 4 - bytes)
e-16		Cmpvers (Compiler Version) (long 4 - bytes)
e-12		filtime.dwHighDateTime (long 4 - bytes)
e-8		filtime.dwLowDateTime (long 4 - bytes)
e-4		InfoSize (in bytes) - l(long 4 bytes -- PC notation, LSB first)
e		0x55FF44EE - marks end of information section (LSB first)

All longs in the info section are in PC notation which means LSB first.
This section has to be read from the end backward.

All longs in other sections are MSB first, for Codeversion < 7 which means firmware < 6.7.0
For firmware >= 6.7.01 Codeversion 7 is used and the Integers are ordered corresponding to the CPU.
That means MSB first for Motorola and LSB first for DSP.

BinFileMap_6_5_36.vsd
2010-08-13-bi


Delta masterposition
SYNCVFTIME
Only active if SYNCVFTIME less than zero
REG_IMAVEL (4100)
PT
G_OldMfiltvel (4188)
SYNCMARKM
SYNCMFTIME
SYNCMARKS
3
To scale the filter in such a way, that the time given by the user is about the time a change needs to come through
G_MarkerFilter (4250)
Is set to SYNCMFTIME / 300 if SYNCMFPAR & 1
G_Vcmd (4186)
SYNCMFTIME
3
G_SMarkerFilter (4920)
Min(100)
To scale the filter in such a way, that the time given by the user is about the time a change needs to come through
REG_ZERO (4098)
G_SOldZero (4200)
G_LastSMDist (4249)
PT
G_SMarkerDist (4239)
Σ
G_IntSMError (4279)
PT
G_SMarkErr (4280)
G_LastSMDeviation (4281)
G_Korrunfilt (4245)
G_StartKorrRest (4241)
(SYNCMPULSS * SYNCMARKS / SYNCOFFTIME) Maximale Korrektur pro ms
G_Mpulses / G_OldMfiltvel Zeit die für einen Mastertakt benötigt wird in ms
if SYNCMFPAR & 4 we use SYNCMPULS * SYNCMARKS / G_MarkerFilter
OldMFiltvel * syncofftim / G_Mpulses
StartKorr (4240)
Falls StartKorrRest > 0
G_DynKorrLimit (4255)
Limit
G_Korrektur (4211)
SYNCMMAXCORR
Limit
G_KorrRest (4212)
G_KorrFilt (4242)
OldMFiltvel / G_Mpulses (Verteilung auf ms für Takt)
G_KorrVal (4213)
KorrVal wird jede ms addiert, bis KorrRest 0 wird.
Only available since Version 6_4_75 and if SYNCMFPAR & 32
Min(100)
PT
G_MMarkerDist (4238)
REG_MIPOS (4106)
G_LastMMDist (4243)
G_MOldZero (4199)
Σ
G_IntMMError (4252)
PT
G_MMarkErr (4253)
G_LastMMDeviation (4278)
Getriebe Umrechnung
G_MDefDist
SYNCMPULSM
G_MMarkCorr (4244)
Getriebe Korrektur
Used to rescale the master input
var
Mit Sysvar lesbar (integer)
var
Mit Sysvar lesbar (1/128)
param
Achsparameter (GET / SET)
Ausgangs-var
Ausgangs Grösse Prozess
Eingangs-var
Eingangs Grösse Prozess
zub machine control ag
MarkerCorrection with MFTIME
Versionen nach 6_7_16
30.10.2009 bi

CurveArray (Long orientiert)

1	1	Identification (999.000.001)	
2	2	Version (101, 102)	
3	3	CurveType	Range is 0 .. 4
4	4	CurveName1 (4char)	
5	5	CurveName2 (4char)	
6	6	CurveName3 (4char)	
7	7	CurveName4 (4char)	
8	8	IndexCIF - CurveInformation -	Default = 17
9	9	Index STP - Start-Stop Points	Default = 31 (32 for Version 102)
10	10	Index FIP - Fixpoint Part	Default STP+STPno*2
11	11	Index INP - Interpolation Part	Default FIP + FixPointNo * 3
12	12	Index STPI - StartStopInerpInd	Default INP + InterPolPointNo
13	13	Index STPV - StartStopVel	Default STP + STPno *2
14	14	Index STIP - StartPathInterpol	Default STPV + STPno*2
15	15	Index STPIP - StopPathInterpol	Default STIP + MaxStartStopLen
16	16	Extra Info Index (CU_GRAD /	LBLINF - Optional Label Info)
17	1	MasterCycleLen (MU)	Length of Curve in Master Units
18	2	SlaveCycleLen (UU) -	Max. Slave travel dist in UU
19	3	FixPoint number	Number of fix points (minimum 2)
20	4	InterpolPointNo	Number of interpol points
21	5	Interpolation Type	(0 = open, 1 = periodic, 2+3 spec)
22	6	SlaveStopPosition	position slave has to be after stop
23	7	CorrectionStartPoint	Pos. where correction may start
24	8	CorrectionStopPoint	Pos. where correction has to stop
25	9	Maximum Correction	maxmal allowed correction
26	10	MaxStartStopLen	max length of start stop path
27	11	StartStopNo	Number of start stop point pairs
28	12	MMaxCycles	Max cycles per minute (info only)
29	13	MMarkerPos	Master Marker Pos. in curve
30	14	SMarkerPos	Slave Marker Pos in curve (cmd)
31	15	ExtraDataSize	Size of extra Data (Version>=102)
32	1	STPoint_1a	Start Point Pair 1 - Point A
		STPoint_1b	Start Point Pair 1 - Point B
		.....	
		FixPoint_1.master	Master Coordinate
		FixPoint_1.slave	Slave Coordinate
		FixPoint_1.type	Type (C = curve, T = Tangent)
		.....	
		FixPoint_n.master	Master Coordinate
		FixPoint_n.slave	Slave Coordinate
		FixPoint_n.type	Type (C = curve, T = Tangent)
		StartVelocityNum	used for Poly5 and Splines
		StartVelocityDen	
		StartAccelerationNum	(not used for Splines)
		StartAccelerationDen	(not used for Splines)
		StartJerkNum	(not used at the moment)
		StartJerkDen	(not used at the moment)
		EndVelocityNum	used for Poly5 and Splines
		EndVelocityDen	
		EndAccelerationNum	(not used for Splines)
		EndAccelerationDen	(not used for Splines)
		EndJerkNum	(not used at the moment)
		EndJerkDen	(not used at the moment)
		Interpolatio nSection (Interpolation Points, Start Stop Indices, StartStop Velocities, StartPath Interpolation Points, StopPath Interpolation Points)	

defines handling of start / end velocities.
0 = start and end velocity is average,
1 = end velocity is set to start velocity (Hauser compatible).
2 = start and end gradients are set to 0 (5th order).
3 = start and end gradients are user defined (CU_GRAD)
4 = start and end gradients user defined (CU_GRAD_INTRPT) (starts immediately)

not used any more, because curves are calculated on the fly now

optional Label Info, only used if Interpolation Type (see 21) is >= 2
Since Version 102 used if CurveType(see 3) is CU_GRAD (3) or CU_GRAD_INTRPT(4). In that case the extra info contains start and stop gradients.

internally overwritten by FixPoint_n.master - FixPoint_1.master

internally overwritten by (FixPoint_n.slave- FixPoint_1.slave)

maximum is 100 at the moment

not used any more, because curves are calculated on the fly now

Interpolation Types are interpreted as follows:
0 = open (only relevant for CAM - editor)
1 = periodic (only relevant for CAM - editor)
2 = Labeling - old version with precalculation
3 = Labeling - actual development version

wird im Moment nicht übergeben !! Stop ohne CURVESTOP wird wohl nicht funktionieren

next 3 only relevant for marker correction

not used any more, because curves are calculated on the fly now

next 2 only relevant for marker correction

At the moment length is 12

values are given in MU. Point A is where the stopping begins and B is where it has to be finished. If driving backward, it is vice versa.

Fixpoints are give in MasterUnits (master coordinate) or UserUnits (slave coordinate). The type can be either
C = CurvePoint
T = TangentPoint
More then two curvepoints are interpolated by cubic spline functions. That means, that a 3rd order Polynom connects two curvepoints in that way, that the position, velocity and acceleration are identical in intermediate Curvepoints.
If there is only one CurvePoint followed by an TangentPoint ore a TangentPoint followed by only one CurvePoint (endpoint) or two tangents following each other, then we use 5th order polynoms to connect these points.

Velocity is used, if the CurveType is 3 or 4 (CU_GRAD or CU_GRAD_INTRPT) and curve does not start with a tangent. If curve starts with a tangent, all other values are ignored too.

Acceleration is used, if the CurveType is 3 or 4 (CU_GRAD or CU_GRAD_INTRPT) and curve starts with a Poly5.

Velocity is used, if the CurveType is 3 or 4 (CU_GRAD or CU_GRAD_INTRPT) and curve does not end with a tangent. If curve ends with a tangent, all other values are ignored too.

Acceleration is used, if the CurveType is 3 or 4 (CU_GRAD or CU_GRAD_INTRPT) and curve ends with a Poly5.

not used any more, because curves are calculated on the fly now

CurveArray_6_6_x.vsd
2008-05-24 bi