9 Planung und Implementierung von DMC-Systeme

In den vorausgegangenen Kapiteln wurden die Grundlagen vermittelt, um eine permanente Bauteilkennzeichnung mittels DMC und anschließender erfolgreicher Dekodierung durchführen zu können. In den nun folgenden Unterkapiteln werden die wichtigsten Fragen für die einzelnen Anwendungsfälle zusammengestellt, um eine effiziente Hilfestellung bei der Planung und Implementierung eines DMC-Komplettsystems zu erhalten. Sollten Fragen hierzu nicht sofort zu beantworten sein, ist es notwendig diese in geeignetster Weise zu recherchieren und gegebenenfalls einen kompetenten Partner aus dem Bereich der maschinellen Identifikation heranzuziehen.

9.1 Bauteilkennzeichnung

Die Bauteilkennzeichnung selbst ist grundlegend in allen anderen Anwendungsbereichen enthalten und ist der erste Schritt der Identifikationsaufgabe. Hierbei ist zu entscheiden welche Art der Dekodierung aufgebracht werden soll. Da es in diesem Buch jedoch nur um den DMC und nicht um andere Dekodierungen geht wird vorausgesetzt, dass man sich bereits für diese Art der Beschriftung entschieden hat. Aus diesem Grund werden im nachstehenden Fragekatalog keine Fragen zur Auswahl der Kodierung gestellt.

- Auf welchem Material soll beschriftet werden ?
- Welche Markierungsverfahren können hier verwendet werden ?
- Wo soll auf dem Bauteil beschriftet werden ?
- Wie groß darf der Code maximal sein ?
- Welche Datenmenge soll im Code vorhanden sein ?
- Welcher Zeichensatz soll dekodiert werden ?
- Welche Störungen können auf das Bauteil beim Leseprozess einwirken ?
- Ist die Markierung für die möglichen Störungen geeignet ?
- Muss oder kann die Beschriftungsfläche vorbehandelt werden ?
- Wie lange muss die Markierung halten ?
- Welches Budget steht für die Markierung zur Verfügung ?

9.2 Dezentrale Datenbank

Wird die Markierung als dezentrale Datenbank verwendet ist die Codierung allein der Träger aller Informationen. Hierbei ist zu beachten, dass mit steigender Datenmenge hohe Ansprüche an die Markierungsqualität und deren Auflösung gestellt werden. Bei großen Datenmengen ist nicht mehr jedes Markierungsverfahren geeignet. Die Fragen der Bauteilkennzeichnung finden hier selbstverständlich genauso Anwendung und werden nicht mehr wiederholt.

- Welche Markierungsverfahren sind aufgrund der benötigten Auflösung noch möglich ?
- Muss die Kodierung im nachfolgenden Leseprozess auch mit Handlesegeräten oder nur mit stationären Geräten erfolgen ?
- Wann und im welchen Umfang werden Dekodiertests zur Auswahl von Beleuchtung und Lesegerät durchgeführt ?
- Wohin sollen die dekodierten Daten gesendet werden ?
- Welche Schnittstellen stehen hierfür zur Verfügung ?
- Welches Budget steht für die Implementierung zur Verfügung ?
- In welcher Zeit soll dass Projekt realisiert werden ?

9.3 Traceabilty

Bei der Bauteilrückverfolgbarkeit ist vor allem die Langlebigkeit der Bauteilbeschriftung ein wichtiges Merkmal. Man kann sich gut vorstellen, dass Beispielsweise ein eingebautes Bauteil in einem Kraftfahrzeug einem Verschleiß unterliegt, hinzu kommen Schmutz und weitere Abnutzungserscheinungen. Soll nun zu einem späteren Zeitpunkt, hierbei können auch mehrere Jahre vergehen, die Kennzeichnung wieder ausgelesen werden, um Rückschlüsse auf den Herstellungsprozess zu erhalten, muss die Kennzeichnung für ein entsprechendes Lesegerät noch erkennbar sein. Diese Aufgabenstellung ist von Anfang an mit zu bedenken.

- Hält die gewählte Markierung den Anforderungen an Langlebigkeit stand?
- Wo werden die Daten der Kodierung gespeichert ?
- Mit welchem Datenvolumen in der Datenbank ist zu rechnen ?
- Können die Daten jederzeit wieder aufgefunden werden ?
- Kann Datenverlust durch geeignete Datensicherungen verhindert werden?
- Welche Lesegeräte sollen verwendet werden, handgehalten und/oder stationär ?
- Wann und wo sollen die Daten dekodiert werden?
- Welche Einbausituation ist für stationäre Lesegeräte angedacht bzw. vorhanden ?

- Wann und im welchen Umfang werden Dekodiertests zur Auswahl von Beleuchtung und Lesegerät durchgeführt ?
- Wohin sollen die dekodierten Daten gesendet werden ?
- Welche Schnittstellen stehen hierfür zur Verfügung ?
- Welches Budget steht für die Implementierung zur Verfügung ?
- In welcher Zeit soll dass Projekt realisiert werden ?

9.4 Prozesssteuerung

Die Prozesssteuerung mittels DMC in Fertigungsstraßen setzt eine hohe Verfügbarkeit der Lesesysteme und eine extrem hohe Dekodierrate voraus. Vor allem hier werden Dekodierraten > 99,9% als Sollvorgabe für den späteren Prozess festgelegt. Jede Nichtlesung führt im schlechtesten Fall zum Stillstand der gesamten Anlage mit deren Verkettungen. Wie bei allen anderen Anwendungsgebieten sollte auch der Bauteilkennzeichnung ein hoher Stellenwert beigemessen werden, damit nicht bereits von Anfang an Störgrößen in den Prozess einfließen. Zusätzlich zu den Fragen der Bauteilkennzeichnung sollen die folgenden Fragen helfen den Prozess von Anfang an optimal zu planen.

- Ist die gewählte Markierungsart geeignet auftretende Störgrößen zu eliminieren ?
- Welche Prozessrate soll erreicht werden ?

- Was geschieht bei einer Nichtlesung ?
- Welches System trifft die Entscheidung der Arbeitsschritte ?
- Müssen die Daten des DMC für eine nachgeschaltete Bauteilrückverfolgbarkeit gespeichert werden ?
- Wie oft muss der DMC gelesen werden ?
- Werden Bauteile in der Fertigungsstraße händisch entnommen und wieder eingeschleust ?
- Welche Lesegeräte sollen verwendet werden, handgehalten und/oder stationär ?
- Welche Einbausituation ist für stationäre Lesegeräte angedacht bzw. vorhanden ?
- Wann und im welchen Umfang werden Dekodiertests zur Auswahl von Beleuchtung und Lesegerät durchgeführt ?
- Welche Schnittstellen sollen zwischen Lesegerät und Anlagentechnik realisiert werden ?
- Welches Budget steht für die Implementierung zur Verfügung ?
- In welcher Zeit soll dass Projekt realisiert werden ?

10 Ergebnisbewertung und Ausblick

Angelangt beim letzten Kapitel sollten wir nun in der Lage sein eine permanente Bauteilkennzeichnung mittels DMC für die genannten Anwendungsgebiete zu planen und auch erfolgreich im Unternehmen zu implementieren. Vor allem die Grundlagen des DMC und die Auswahl eines geeigneten Markierungsverfahren schaffen eine optimale Basis um auch ein späteres Dekodieren mit Prozessraten > 99,9 % zu bewerkstelligen.

Im Rückblick auf die Kapitel 1 bis 9 besitzt man nun das nötige Fachwissen, um auch komplexe Identifikations-aufgaben mittels DMC zu lösen. Wie jedoch auch mehrfach erwähnt, ist die Erfahrung kompetenter Berater nicht weg zu diskutieren und helfen bei einer effizienten und optimalen Umsetzung. Auf dem Markt wird viel versprochen, aber nicht jeder Anbieter von DMC Systemen besitzt das notwendige know-how, um im direkt-markierten Anwendungsbereich ein Optimum an Beratung und geeigneten Geräten anbieten zu können. Fragen Sie deshalb immer nach Referenzen und vereinbaren Sie gegebenenfalls auch einen Besuchstermin gemeinsam mit Ihrem Berater beim Referenzkunden. Vor allem bei kostenintensiven und Erst- Projekten hilft es sehr, bei der eigenen Umsetzung ähnliche Lösungen in der Praxis zu begutachten.

Durch die vielen verschiedenen Aufgabenstellungen im DPM Bereich wird es auch in Zukunft immer mehr Lese- und Markierungssysteme auf dem Markt geben. Das vorliegende Buch beschäftigte sich mit dem „Jetzt“ und nicht mit dem was kommen mag. Jedoch ist vor allem der Fortschritt in der Miniaturisierung und der Leistungssteigerung in der Elektronikindustrie noch keine Grenzen gesetzt und somit werden auch noch in den nächsten Jahren eine Vielzahl neuer DMC Geräte auf dem Markt angeboten werden. Eine

intensive Internetrecherche vor dem Kauf von Produkten würde ich deshalb zu jeder Zeit empfehlen.

Bei Fragen und der Suche nach geeigneten Produkten rund um das Thema Identifikation mit DataMatrix, Strichcode und Bildverarbeitung wenden Sie sich am besten direkt an die FuWa Informationssysteme.

Sie finden unsere Kontaktdaten unter www.fuwa-it.de.

Literaturverzeichnis

AIM International Technical Specification

Internet-Quellen

http://www.lyngsoesystems.com/graphics/
SupplyChainIllustration – RGB.jpg
Abruf: 26.02.2011

http://www.neffwashere.com/Tutorial/images/
background_asciichart.gif
Abruf: 02.03.2011

http://www.borries.com

http://www.unglaube.de

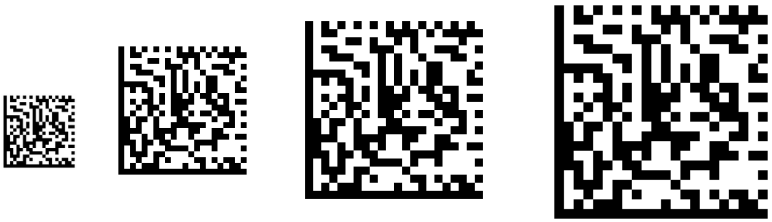
http://www.kba-metronic.com

DMC ECC200 Beispielcodes

ABC

01234567890

ABCDEFGHIJKLMNOPQRSTUVWXYZ



ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrs
tuvwxyz

Anmerkung:
Die physikalische Größe eines DMC ist allein von der Auflösung des Erzeugungsverfahren abhängig. Die Anzahl der Zeilen und Spalten ist abhängig vom kodierten Datenvolumen.

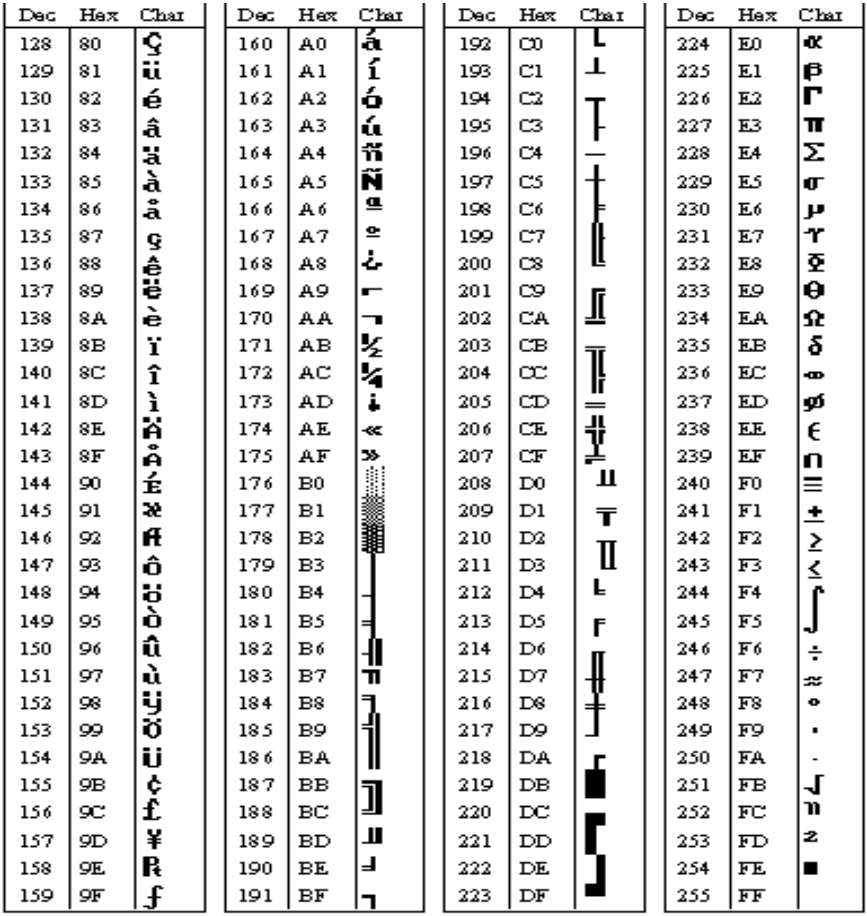
Ctrl	Dec	Hex	Char	Code
^@	0	00		NUL
^A	1	01	☺	SOH
^B	2	02	☻	SIX
^C	3	03	♥	EIX
^D	4	04	♦	EOI
^E	5	05	♣	ENQ
^F	6	06	♠	ACK
^G	7	07	•	BEL
^H	8	08	◘	BS
^I	9	09	○	HI
^J	10	0A	◙	LF
^K	11	0B	♂	VI
^L	12	0C	♀	FF
^M	13	0D	♪	CR
^N	14	0E	♫	SO
^O	15	0F	☼	SI
^P	16	10	►	SLE
^Q	17	11	◄	CS1
^R	18	12	↕	DC2
^S	19	13	‼	DC3
^I	20	14	¶	DC4
^U	21	15	§	NAK
^V	22	16	▬	SYN
^W	23	17	↨	EIB
^X	24	18	↑	CAN
^Y	25	19	↓	EM
^Z	26	1A	→	SIB
^[	27	1B	←	ESC
^\	28	1C	∟	FS
^]	29	1D	↔	GS
^^	30	1E	▲	RS
^_	31	1F	▼	US

Dec	Hex	Char
32	20	sp
33	21	!
34	22	"
35	23	#
36	24	$
37	25	%
38	26	&
39	27	'
40	28	(
41	29	)
42	2A	*
43	2B	+
44	2C	,
45	2D	-
46	2E	.
47	2F	/
48	30	0
49	31	1
50	32	2
51	33	3
52	34	4
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9
58	3A	:
59	3B	;
60	3C	<
61	3D	=
62	3E	>
63	3F	?

Dec	Hex	Char
64	40	@
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
70	46	F
71	47	G
72	48	H
73	49	I
74	4A	J
75	4B	K
76	4C	L
77	4D	M
78	4E	N
79	4F	O
80	50	P
81	51	Q
82	52	R
83	53	S
84	54	T
85	55	U
86	56	V
87	57	W
88	58	X
89	59	Y
90	5A	Z
91	5B	[
92	5C	\
93	5D	]
94	5E	^
95	5F	_

Dec	Hex	Char
96	60	`
97	61	a
98	62	b
99	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j
107	6B	k
108	6C	l
109	6D	m
110	6E	n
111	6F	o
112	70	p
113	71	q
114	72	r
115	73	s
116	74	t
117	75	u
118	76	v
119	77	w
120	78	x
121	79	y
122	7A	z
123	7B	{
124	7C	!
125	7D	}
126	7E	~
127	7F	⌂

Full ASCII Chart Teil 1



Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
128	80	Ç	160	A0	á	192	C0	└	224	E0	α
129	81	ü	161	A1	í	193	C1	┴	225	E1	ß
130	82	é	162	A2	ó	194	C2	┬	226	E2	Γ
131	83	â	163	A3	ú	195	C3	├	227	E3	π
132	84	ä	164	A4	ñ	196	C4	─	228	E4	Σ
133	85	à	165	A5	Ñ	197	C5	┼	229	E5	σ
134	86	å	166	A6	ª	198	C6	╞	230	E6	µ
135	87	ç	167	A7	º	199	C7	╟	231	E7	τ
136	88	ê	168	A8	¿	200	C8	╚	232	E8	Φ
137	89	ë	169	A9	⌐	201	C9	╔	233	E9	Θ
138	8A	è	170	AA	¬	202	CA	╩	234	EA	Ω
139	8B	ï	171	AB	½	203	CB	╦	235	EB	δ
140	8C	î	172	AC	¼	204	CC	╠	236	EC	∞
141	8D	ì	173	AD	¡	205	CD	═	237	ED	φ
142	8E	Ä	174	AE	«	206	CE	╬	238	EE	ε
143	8F	Å	175	AF	»	207	CF	╧	239	EF	∩
144	90	É	176	B0	▒	208	D0	╨	240	F0	≡
145	91	æ	177	B1	▒	209	D1	╤	241	F1	±
146	92	Æ	178	B2	▒	210	D2	╥	242	F2	≥
147	93	ô	179	B3	▒	211	D3	╙	243	F3	≤
148	94	ö	180	B4	▒	212	D4	╘	244	F4	⌠
149	95	ò	181	B5	▒	213	D5	╒	245	F5	⌡
150	96	û	182	B6	▒	214	D6	╓	246	F6	÷
151	97	ù	183	B7	▒	215	D7	╫	247	F7	≈
152	98	ÿ	184	B8	▒	216	D8	╪	248	F8	°
153	99	Ö	185	B9	▒	217	D9	┘	249	F9	∙
154	9A	Ü	186	BA	▒	218	DA	┌	250	FA	·
155	9B	¢	187	BB	▒	219	DB	▒	251	FB	√
156	9C	£	188	BC	▒	220	DC	▒	252	FC	ⁿ
157	9D	¥	189	BD	▒	221	DD	▒	253	FD	²
158	9E	₧	190	BE	▒	222	DE	▒	254	FE	■
159	9F	ƒ	191	BF	▒	223	DF	▒	255	FF	

Full ASCII Chart Teil 2

Quelle:
http://www.neffwashere.com/Tutorial/images/background_asciichart.gif

Listing C# Codewortverteilung ECC200

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Collections;

namespace ECC200
{
   class ECC200
   {
      static int nrow, ncol;
      static int[] module_array;

      static void Main(string[] args)
      {
         int x, y, z;
         if (args.Length < 2)
         {
            Console.Write("Command line: #_of_Data_Rows #_of_Data_Columns\n");
         }
         else
         {
            nrow = ncol = 0;
            nrow = Convert.ToInt16(args[0]); ncol = Convert.ToInt16(args[1]);
            if ((nrow >= 6) && Convert.ToBoolean(~nrow & 0x01) && (ncol >= 6)
                           && Convert.ToBoolean(~ncol & 0x01))
            {
               module_array = new int[sizeof(int) * nrow * ncol];
            }
            ecc200();
            for (x = 0; x < nrow; x++)
            {
               for (y = 0; y < ncol; y++)
               {
                  z = module_array[x * ncol + y];
                  if (z == 0) Console.Write("WHI".PadLeft(5));
                  else if (z == 1) Console.Write("BLK".PadLeft(5));
                  else Console.Write((
                  Convert.ToString((z / 10) + (z % 10)/10.0).PadLeft(5)).Replace(",","."));
               }
               Console.Write("\n");
            }
         }
         Console.ReadKey();
      }

      static void module(int row, int col, int chr, int bit)
      {
                  if(row < 0){row += nrow; col += 4 - ((nrow + 4) % 8);}
                  if(col < 0){col += ncol; row += 4 - ((ncol + 4) % 8);}
                  module_array[row * ncol + col] = 10 * chr + bit;
      }
```

```
static void utah(int row, int col, int chr)
{
        module(row-2,col-2,chr,1);
        module(row-2,col-1,chr,2);
        module(row-1,col-2,chr,3);
        module(row-1,col-1,chr,4);
        module(row-1,col,chr,5);
        module(row,col-2,chr,6);
        module(row,col-1,chr,7);
        module(row,col,chr,8);
}

static void corner1(int chr)
{
        module(nrow-1,0,chr,1);
        module(nrow-1,1,chr,2);
        module(nrow-1,2,chr,3);
        module(0,ncol-2,chr,4);
        module(0,ncol-1,chr,5);
        module(1,ncol-1,chr,6);
        module(2,ncol-1,chr,7);
        module(3,ncol-1,chr,8);
}

static void corner2(int chr)
{
        module(nrow-3,0,chr,1);
        module(nrow-2,0,chr,2);
        module(nrow-1,0,chr,3);
        module(0,ncol-4,chr,4);
        module(0,ncol-3,chr,5);
        module(0,ncol-2,chr,6);
        module(0,ncol-1,chr,7);
        module(1,ncol-1,chr,8);
}

static void corner3(int chr)
{
        module(nrow-3,0,chr,1);
        module(nrow-2,0,chr,2);
        module(nrow-1,0,chr,3);
        module(0,ncol-2,chr,4);
        module(0,ncol-1,chr,5);
        module(1,ncol-1,chr,6);
        module(2,ncol-1,chr,7);
        module(3,ncol-1,chr,8);
}

static void corner4(int chr)
{
        module(nrow-1,0,chr,1);
        module(nrow-1,ncol-1,chr,2);
        module(0,ncol-3,chr,3);
        module(0,ncol-2,chr,4);
```

```
                module(0,ncol-1,chr,5);
                module(1,ncol-3,chr,6);
                module(2,ncol-2,chr,7);
                module(3,ncol-1,chr,8);
        }


        static private void ecc200()
        {
                int row, col, chr;
                for(row = 0; row < nrow; row++)
                {for (col = 0; col < ncol; col++){module_array[row * ncol + col] = 0;} }
                chr = 1; row = 4; col =0;
                do
                {
                        if((row == nrow) && (col == 0)) corner1(chr++);
                        if((row == nrow - 2) && (col == 0)
                                        && Convert.ToBoolean(ncol % 4)) corner2(chr++);
                        if((row == nrow - 2) && (col == 0) && (ncol % 8 == 4))
                              corner3(chr++);
                        if((row == nrow + 4) && (col == 2)
                              && (!Convert.ToBoolean(ncol % 8))) corner4(chr++);

                        do
                        {
                              if((row < nrow) && (col >= 0)
                                 && (!Convert.ToBoolean(module_array[row * ncol + col])))
                                   utah(row,col,chr++);
                              row -= 2; col += 2;
                        }while((row >= 0) && (col < ncol));
                        row += 1; col += 3;

                        do
                        {
                               if(( row >= 0) && (col < ncol)
                                 && (!Convert.ToBoolean(module_array[row * ncol + col])))
                                    utah(row,col,chr++);
                               row += 2; col -= 2;
                        }while ((row < nrow) && (col >= 0));
                        row += 3; col +=1;

                }while ((row < nrow) || (col < ncol));

                if(!Convert.ToBoolean(module_array[nrow * ncol - 1]))
                {
                     module_array[nrow * ncol - 1] = module_array[nrow * ncol - ncol -1] = 1;
                }
        }
    }
}
```

Quelle: AIM International Technical Specification (von ANSI C auf C# portiert)